Machine Learning

Bert Kappen
SNN Radboud University, Nijmegen
Gatsby Unit, UCL London

October 23, 2016
Course setup

- 6 ec course, until mid January 2017

- Examination based on weekly exercises (25 %) and computer exercises (75 %)

- Instructors: Hans Ruiz, Giel van Bergen

- All course materials (slides, exercises) and schedule via [http://www.snn.ru.nl/~bertk/machinelearning/](http://www.snn.ru.nl/~bertk/machinelearning/)
Introduction: Probability, Entropy and Inference
Exercise 2.4

Urn with $B$ black balls and $W$ white balls and $K = W + W$.

Draw $N$ times a ball from the urn with replacement. What is the probability to draw $N_B$ black balls?

Define $f = B/K$. Then

$$p(N_B = 0) = (1 - f)^N$$
$$p(N_B = 1) = f(1 - f)^{N-1}N$$

$$p(N_B) = \left( \begin{array}{c} N \\ N_B \end{array} \right) f^{N_B}(1 - f)^{N-N_B}$$

Expected value and variance:

$$\sum_{N_B=0}^N p(N_B)N_B = \ldots = Nf$$
$$\sum_{N_B=0}^N p(N_B)(N_B - Nf)^2 = \ldots = Nf(1 - f)$$
Exercise 2.6

11 urns \( u = 0, \ldots, 10 \) each with 10 balls. Urn \( u \) has \( u \) black balls and \( 10 - u \) white balls. Select one urn at random, and draw \( N \) times with replacement from that urn. Given that after \( N = 10 \) draws there are \( N_B = 3 \) black balls, what is the probability that urn \( u \) was selected?

\[
p(u, N_B | N) = p(u) p(N_B | u, N)
\]

\[
p(u | N_B, N) = \frac{p(u) p(N_B | u, N)}{p(N_B | N)}
\]

\[
p(N_B | u, N) = \binom{N}{N_B} f_u^{N_B} (1 - f_u)^{N - N_B}
\]

\[
f_u = \frac{u}{10} \quad p(u) = \frac{1}{11}
\]

\[
p(N_B | N) = \sum_{u=0}^{10} p(u) p(N_B | u, N) = \frac{1}{11} \binom{N}{N_B} \sum_{u=0}^{10} f_u^{N_B} (1 - f_u)^{N - N_B}
\]

\[
p(u | N_B, N) = \frac{f_u^{N_B} (1 - f_u)^{N - N_B}}{\sum_{u'=0}^{10} f_{u'}^{N_B} (1 - f_{u'})^{N - N_B}}
\]
Left: Joint probability $p(u, N_B | N)$. Right: conditional probability $p(u | N_B, N)$. 
Exercise 2.7. The bent coin

Bent coin has probability $f$ to come up head. We do not know $f$. We toss $N$ times and get $N_H$ times head. What is $f$?

Assume a prior over $f$: $p(f)$. NB $f$ is not a random variable. $p(f)$ is our (subjective) prior belief in the value of $f$.

Given $f$, we know the likelihood of the observation:

$$p(N_H|f, N) = \binom{N}{N_H} f^{N_H} (1-f)^{N-N_H}$$

The posterior is

$$p(f|N_H, N) = \frac{p(N_H|f, N)p(f)}{p(N_H|N)}$$

When $p(f) = 1$ (maximal prior ignorance),

$$p(N_H|N) = \int_0^1 df p(f)p(N_H|f, N) = \binom{N}{N_H} \int_0^1 df f^{N_H} (1-f)^{N-N_H} = \binom{N}{N_H} \frac{N_H!(N-N_H)!}{(N+1)!} = \frac{1}{N+1}$$
Part IV: Probabilities and Inference

Inference: "How to compute marginal probabilities in a probability model" Examples:

- Graphical models
- Learning

Methods:

- Exact (enumeration ch. 21, marginalisation ch. 24)
- Deterministic approximations (max. likelihood, ch 22, Laplace ch. 27, variational ch. 33)
- Monte Carlo methods (ch. 29, 30)
- Ising model (ch. 31)

Part IV deals with these problems in the context of clustering.
Graphical models

Probabilistic approach to reasoning:

\[ p(A, S, T, L, B, E, X, D) = p(A)p(S)p(T|A) \]
\[ p(L|S)p(B|S)p(E|T, L)p(X|E)p(D|E, B) \]

What are the node probabilities given evidence?
Clustering/mixture of Gaussians as examples of learning

• Clustering (ch. 20)

• Learning a single Gaussian: Bayesian posterior (Ch 21.2) and maximum likelihood (Ch 22.1)

• Learning a mixture of Gaussians: Bayesian posterior (Ch 21.2) and maximum likelihood (Ch 22.2)

• Gradient descent methods and the EM algorithm
Ch. 20: Clustering

Clustering: group things that are similar

- "brown things that run away" $\rightarrow$ "animals"
- "green things that don't run away" $\rightarrow$ "plants"

Clustering is good for prediction and communication

One type of clustering is called vector quantisation:

- find $k(\vec{x})$ that encode each $\vec{x}$ in a label $k = 1, \ldots, K$
- find $k \rightarrow \vec{m}^k$ for decoding

$$D = \sum_{\vec{x}} P(\vec{x}) \frac{1}{2} \left( \vec{m}^{k(\vec{x})} - \vec{x} \right)^2$$
**K-means clustering**

**Initialization.** Set $K$ means $\{m^{(k)}\}$ to random values.

**Assignment step.** Each data point $n$ is assigned to the nearest mean. We denote our guess for the cluster $k^{(n)}$ that the point $x^{(n)}$ belongs to by $\hat{k}^{(n)}$.

$$\hat{k}^{(n)} = \arg\min_k \{d(m^{(k)}, x^{(n)})\}. \quad (20.3)$$

An alternative, equivalent representation of this assignment of points to clusters is given by ‘responsibilities’, which are indicator variables $r_k^{(n)}$. In the assignment step, we set $r_k^{(n)}$ to one if mean $k$ is the closest mean to datapoint $x^{(n)}$; otherwise $r_k^{(n)}$ is zero.

$$r_k^{(n)} = \begin{cases} 1 & \text{if } \hat{k}^{(n)} = k \\ 0 & \text{if } \hat{k}^{(n)} \neq k. \end{cases} \quad (20.4)$$

What about ties? – We don’t expect two means to be exactly the same distance from a data point, but if a tie does happen, $\hat{k}^{(n)}$ is set to the smallest of the winning $\{k\}$.

**Update step.** The model parameters, the means, are adjusted to match the sample means of the data points that they are responsible for.

$$m^{(k)} = \frac{\sum_n r_k^{(n)} x^{(n)}}{R^{(k)}} \quad (20.5)$$

where $R^{(k)}$ is the total responsibility of mean $k$,

$$R^{(k)} = \sum_n r_k^{(n)}. \quad (20.6)$$

What about means with no responsibilities? – If $R^{(k)} = 0$, then we leave the mean $m^{(k)}$ where it is.

Repeat the assignment step and update step until the assignments do not change.
$K$-means clustering

Two clusters:

Data:

Assignment  Update  Assignment  Update  Assignment  Update
$K$-means clustering

Results depend on initial condition

Run 1: convergence after 5 iterations; 4 equally sized clusters
Run 2: convergence after 6 iterations; 1 large cluster, 3 small ones
$K$-means clustering

Ad-hoc features:

- which distance to use?
- which value of $K$?
- $K$-means does not take breadth or weight of cluster into account.

(a) 75 data points from two gaussians. Large cluster has 5 times more data than small cluster. (b) Clustering ($K = 2$) yields wrong assignments of data points and displaced means.
**K-means clustering**

Ad-hoc features:

- which distance to use?
- which value of $K$?
- $K$-means does not take breadth or weight of cluster into account.

(a) two elongated clusters are clustered wrongly ($K = 2$ b).
Soft $K$-means clustering

**Assignment step.** Each data point $x^{(n)}$ is given a soft ‘degree of assignment’ to each of the means. We call the degree to which $x^{(n)}$ is assigned to cluster $k$ the responsibility $r_k^{(n)}$ (the responsibility of cluster $k$ for point $n$).

$$r_k^{(n)} = \frac{\exp(-\beta d(m^{(k)}, x^{(n)}))}{\sum_{k'} \exp(-\beta d(m^{(k')}, x^{(n)}))}.$$  \hspace{1cm} (20.7)

The sum of the $K$ responsibilities for the $n$th point is 1.

**Update step.** The model parameters, the means, are adjusted to match the sample means of the data points that they are responsible for.

$$m^{(k)} = \frac{\sum_n r_k^{(n)} x^{(n)}}{R^{(k)}}$$  \hspace{1cm} (20.8)

where $R^{(k)}$ is the total responsibility of mean $k$,

$$R^{(k)} = \sum_n r_k^{(n)}. \hspace{1cm} (20.9)$$

The update is identical. The only difference is that the responsibilities take values between 0 and 1.
Soft $K$-means clustering

Large $\sigma$ …

Circles indicate length scale $\sigma = 1/\sqrt{\beta}$. Bifurcation as a function of $\beta$. 

… small $\sigma$
Chapter 21.2: Exact inference for single Gaussian

Hypothesis space for a Gaussian distribution in 1 dimension:

\[ p(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(\frac{-(x - \mu)^2}{2\sigma^2}\right) \]
Chapter 21.2: Exact inference for single Gaussian

Give some data, we can evaluate $p(\{x_n\}_{n=1}^N | \mu, \sigma)$
Chapter 22: Maximum likelihood for Gaussian

We return to the Gaussian for our first examples. Assume we have data \( \{x_n\}_{n=1}^{N} \). The log likelihood is:

\[
\ln P(\{x_n\}_{n=1}^{N} | \mu, \sigma) = -N \ln(\sqrt{2\pi}\sigma) - \sum_n (x_n - \mu)^2/(2\sigma^2). \tag{22.1}
\]

The likelihood can be expressed in terms of two functions of the data, the sample mean

\[
\bar{x} \equiv \sum_{n=1}^{N} x_n/N, \tag{22.2}
\]

and the sum of square deviations

\[
S \equiv \sum_n (x_n - \bar{x})^2: \tag{22.3}
\]

\[
\ln P(\{x_n\}_{n=1}^{N} | \mu, \sigma) = -N \ln(\sqrt{2\pi}\sigma) - [N(\mu - \bar{x})^2 + S]/(2\sigma^2). \tag{22.4}
\]

Because the likelihood depends on the data only through \( \bar{x} \) and \( S \), these two quantities are known as **sufficient statistics**.
Maximum likelihood for Gaussian

For given $\sigma$, posterior solution for $\mu$ is Gaussian with mean $\mu = \bar{x}$ and variance $\sigma^\mu = \frac{\sigma}{\sqrt{N}}$. 
Maximum likelihood for Gaussian

Example 22.3. Find the maximum likelihood standard deviation \( \sigma \) of a Gaussian, whose mean is known to be \( \mu \), in the light of data \( \{x_n\}_{n=1}^N \). Find the second derivative of the log likelihood with respect to \( \ln \sigma \), and error bars on \( \ln \sigma \).

Solution. The likelihood’s dependence on \( \sigma \) is

\[
\ln P(\{x_n\}_{n=1}^N | \mu, \sigma) = -N \ln(\sqrt{2\pi\sigma}) - \frac{S_{\text{tot}}}{(2\sigma^2)},
\]

(22.9)

where \( S_{\text{tot}} = \sum_n (x_n - \mu)^2 \). To find the maximum of the likelihood, we can differentiate with respect to \( \ln \sigma \). [It’s often most hygienic to differentiate with respect to \( \ln u \) rather than \( u \), when \( u \) is a scale variable; we use \( du^n / d(\ln u) = nu^n \).]

\[
\frac{\partial \ln P(\{x_n\}_{n=1}^N | \mu, \sigma)}{\partial \ln \sigma} = -N + \frac{S_{\text{tot}}}{\sigma^2}
\]

(22.10)

This derivative is zero when

\[
\sigma^2 = \frac{S_{\text{tot}}}{N},
\]

(22.11)

i.e.,

\[
\sigma = \sqrt{\frac{\sum_{n=1}^N (x_n - \mu)^2}{N}}.
\]

(22.12)

The second derivative is

\[
\frac{\partial^2 \ln P(\{x_n\}_{n=1}^N | \mu, \sigma)}{\partial (\ln \sigma)^2} = -2\frac{S_{\text{tot}}}{\sigma^2},
\]

(22.13)

and at the maximum-likelihood value of \( \sigma^2 \), this equals \(-2N\). So error bars on \( \ln \sigma \) are

\[
\sigma_{\ln \sigma} = \frac{1}{\sqrt{2N}}.
\]

(22.14)
Exact inference for Gaussian mixture

\[
p(x|\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1, \sigma_2) = \frac{\pi_1}{\sqrt{2\pi}\sigma_1} \exp\left(-\frac{(x - \mu_1)^2}{2\sigma_1^2}\right) + \frac{\pi_2}{\sqrt{2\pi}\sigma_2} \exp\left(-\frac{(x - \mu_2)^2}{2\sigma_2^2}\right)
\]

Weight of the mixture components is \(\pi_1, \pi_2 = 0.6, 0.4\) in the top half and \(0.8, 0.2\) in the bottom half. Means \(\mu_1\) and \(\mu_2\) vary horizontally, and standard deviations \(\sigma_1\) and \(\sigma_2\) vary vertically.
Exact inference for Gaussian mixture

\[ p(x|\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1, \sigma_2) = \frac{\pi_1}{\sqrt{2\pi\sigma_1}} \exp\left(\frac{-(x - \mu_1)^2}{2\sigma_1^2}\right) + \frac{\pi_2}{\sqrt{2\pi\sigma_2}} \exp\left(\frac{-(x - \mu_2)^2}{2\sigma_2^2}\right) \]

Bayesian posterior is distribution over 5 dimensional continuous space.
Maximum likelihood for mixture of two Gaussians

\[ p(x | \vec{\mu}, \sigma) = \sum_{k=1}^{2} p_k \mathcal{N}(x | \mu_k, \sigma) \]

\( \mathcal{N}(x | \mu, \sigma) \) is the Normal distribution, \( p_1 = p_2 = \frac{1}{2} \), \( \vec{\mu} = (\mu_1, \mu_2) \).

Introduce hidden variable \( k = 1, 2 \) and define

\[ p(x, k) = p_k \mathcal{N}(x | \mu_k, \sigma) \]

then \( p(x) = \sum_k p(x, k) \).

Given \( \mu_1, \mu_2, \sigma \) we compute responsibilities for each data point:

\[ p_{k|n} = p(k | x_n) = \frac{p(x_n, k)}{p(x_n)} = \frac{\mathcal{N}(x_n | \mu_k, \sigma)}{\sum_{l=1}^{2} \mathcal{N}(x_n | \mu_l, \sigma)} \]

\( 0 \leq p_{k|n} \leq 1 \) and \( p_{1|n} + p_{2|n} = 1 \).
Maximum likelihood for mixture of two Gaussians

Define

\[ L = \sum_{n=1}^{N} \log p(x_n|\mu, \sigma) \]

Then

\[ \frac{\partial L}{\partial \mu_k} = \sum_n \frac{1}{p(x_n)} \frac{\partial p(x_n, k)}{\partial \mu_k} = \sum_n p_{k|n} \frac{x_n - \mu_k}{\sigma^2} \]

\[ H_{kl} = \frac{\partial^2 L}{\partial \mu_k \partial \mu_l} \approx - \sum_n p_{k|n} \frac{1}{\sigma^2} \delta_{kl} \]

Using Newtons method the update becomes:

\[ \mu_k' = \mu_k - \sum_l H_{kl}^{-1} \frac{\partial L}{\partial \mu_l} = \sum_n p_{k|n} x_n \]

\[ \sum_n p_{k|n} \]

This is identical to the soft clustering algorithm.
Gradient descent to minimize a function

Gradient descent: walk downwards in with small steps the direction of the negative gradient.

\( E \) is minimal when \( \nabla E(w) = 0 \), but not vice versa!

\[ \Rightarrow \text{gradient based methods find a local minimum, not necessary the global minimum.} \]
Gradient descent to minimize a function

Gradient descent algorithm:

1. Start with an initial value of $w$ and $\epsilon$ small.

2. While ”change in $w$ large”: Compute $w := w - \epsilon \nabla E$

Stop criterion is $\nabla E \approx 0$, which means that we stop in a local minimum of $E$.

Does this algorithm converge? Yes, if $\epsilon$ is ”sufficiently small” and $E$ bounded from below.

Proof: Denote $\Delta w = -\epsilon \nabla E$.

$$E(w + \Delta w) \approx E(w) + (\Delta w)^T \nabla E = E(w) - \epsilon \sum_i \left( \frac{\partial E}{\partial w_i} \right)^2 \leq E(w)$$

In each gradient descent step the value of $E$ is lowered. Since $E$ bounded from below, the procedure must converge asymptotically.
Newtons method

One can also use Hessian information for optimization. As an example, consider a quadratic approximation to $E$ around $w_0$:

$$E(w) = E(w_0) + b^T(w - w_0) + \frac{1}{2}(w - w_0)H(w - w_0)$$

$$b_i = \frac{\partial E(w_0)}{\partial w_i} \quad H_{ij} = \frac{\partial^2 E(w_0)}{\partial w_i \partial w_j}$$

$$\nabla E(w) = b + H(w - w_0)$$

We can solve $\nabla E(w) = 0$ and obtain

$$w = w_0 - H^{-1}\nabla E(w_0)$$

This is called Newtons method. Inversion of the Hessian may be computational costly. A number of methods, known as quasi-newton methods, are based on approximations of this procedure.
The original EM method was introduced in [Baum et al., 1970].

Consider a model \( p(x, y|\theta) \). The problem is to find \( \theta \) that maximizes the observed data likelihood \( L(\theta) = \sum_y \hat{q}(y) \log p(y|\theta) \) with \( \hat{q}(y) \) the empirical distribution of \( y \). Taking the gradient wrt \( \theta \) and setting to zero, we find

\[
\sum_y \hat{q}(y) \sum_x p(x, y|\theta) \frac{d \log p(x, y|\theta)}{d \theta} = 0
\]

which is not easy.

We repeat the estimation using the EM algorithm. For each \( y \), we write

\[
\log p(y|\theta) = \log \sum_x p(x, y|\theta) = \log \sum_x s_y(x) \frac{p(x, y|\theta)}{s_y(x)} \geq \sum_x s_y(x) \log \frac{p(x, y|\theta)}{s_y(x)} = L_y
\]
EM

The E step consists of maximizing $L_y$ wrt $s_y(x)$ for given $p(x, y|\theta_{old})$. When $s_y(x)$ is unparametrized and fully optimized, the result is simply Bayes rule:

\[
E\text{step} : \quad s_y(x) = \frac{p(x, y|\theta_{old})}{p(y|\theta_{old})} = p(x|y, \theta_{old}) \quad L_y = \sum_x p(x|y, \theta_{old}) \log \frac{p(x, y|\theta)}{p(x|y, \theta_{old})}
\]

The M step is to maximize $L_{EM} = \sum_y \hat{q}(y)L_y$ wrt $\theta$ for fixed $\theta_{old}$. The fixed point equation is

\[
M\text{step} : \quad \sum_y \hat{q}(y) \sum_x p(x|y, \theta_{old}) \frac{d \log p(x, y|\theta)}{d\theta} = 0
\]

Clustering: $y = x \in R^d$ is observed data; $x = k = 1, \ldots, K$ are the cluster labels. $\theta$ is $\{m^k, \sigma^2_k\}, k = 1, \ldots, K$ (sphetical Gaussians) $\theta$ is $\{m^k, (\sigma^2_i)^k\}, k = 1, \ldots, K$ and $i = 1, \ldots, d$ (axis aligned Gaussians)
Maximum likelihood via EM for mixture of $K$ spherical Gaussians

**Assignment step.** The responsibilities are

$$
\tau_k^{(n)} = \frac{\pi_k \frac{1}{(\sqrt{2\pi}\sigma_k)^I} \exp \left( -\frac{1}{\sigma_k^2} d(m^{(k)}, x^{(n)}) \right)}{\sum_{k'} \pi_k \frac{1}{(\sqrt{2\pi}\sigma_{k'})^I} \exp \left( -\frac{1}{\sigma_{k'}^2} d(m^{(k')}, x^{(n)}) \right)}
$$

where $I$ is the dimensionality of $x$.

**Update step.** Each cluster's parameters, $m^{(k)}$, $\pi_k$, and $\sigma_k^2$, are adjusted to match the data points that it is responsible for.

$$
m^{(k)} = \frac{\sum_n \tau_k^{(n)} x^{(n)}}{R^{(k)}}
$$

$$
\sigma_k^2 = \frac{\sum_n \tau_k^{(n)} (x^{(n)} - m^{(k)})^2}{IR^{(k)}}
$$

$$
\pi_k = \frac{R^{(k)}}{\sum_k R^{(k)}}
$$

where $R^{(k)}$ is the total responsibility of mean $k$,

$$
R^{(k)} = \sum_n \tau_k^{(n)}.
$$
Maximum likelihood for mixture of $K$ spherical Gaussians

Performance on same examples as before shows that size of cluster is properly found but requires more iterations.

$K = 2$ with soft K max algorithm with spherical Gaussians works well
Maximum likelihood via EM for mixture of $K$ axis-alligned Gaussians

\[
\begin{align*}
r_k^{(n)} &= \frac{\pi_k}{\prod_{i=1}^{I} \sqrt{2 \pi \sigma_i^{(k)}}} \exp \left( - \frac{1}{2} \sum_{i=1}^{I} \frac{(m_i^{(k)} - x_i^{(n)})^2}{\sigma_i^{(k)}^2} \right) \\
\quad \quad \quad \quad \text{(numerator, with } k' \text{ in place of } k) \\
\sigma_{i}^{2(k)} &= \frac{\sum_{k} r_k^{(n)} (x_i^{(n)} - m_i^{(k)})^2}{R^{(k)}} \\
\end{align*}
\]

(22.27) (22.28)
Maximum likelihood for mixture of $K$ axis-aligned Gaussians

Performance on same examples as before shows that size and shape of cluster is properly found.
Overfitting: When a cluster covers only 1 data point the likelihood can be made arbitrary large by shrinking the variance to zero.

The posterior density has infinite spikes, and its integrated probability is small.

Adding a prior and computing $\max_\theta p(Data|\theta)p(\theta)$ does not resolve the problem.
Chapter 24: Marginalisation

Given the Gaussian distribution

\[ p(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) \]

the log likelihood for \( N \) data points is

\[
\log p(\{x_n\}_{n=1}^N | \mu, \sigma) = -N \log (\sqrt{2\pi\sigma}) - \sum_n (x_n - \mu)^2 / 2\sigma^2
\]

\[
= -N \log (\sqrt{2\pi\sigma}) - \left[N(\mu - \bar{x})^2 + S\right] / 2\sigma^2
\]

with \( S = \sum_n (x_n - \bar{x})^2 \).

The posterior over \( \mu, \sigma \) assuming flat priors is Gaussian in \( \mu \) but not in \( \sigma \).

The maximum posterior likelihood is given by \( \{\mu, \sigma\}_{ML} = \{\bar{x}, \sigma_N = \sqrt{S/N}\} \).

\( \mu_{ML} \) is unbiased: \( \mathbb{E}\mu_{ML} = \mathbb{E}x_n = \mu \)

\( \sigma_{ML}^2 \) is biased: \( \mathbb{E}\sigma_{ML}^2 = \frac{N-1}{N} \sigma^2 \) (show!)
Chapter 24: Marginalisation

Estimating $\mu$ is unbiased for each $\sigma$:

$$
p(\mu|\{x_n\}_{n=1}^N, \sigma) \propto \exp\left(-N(\mu - \bar{x})^2/2\sigma^2\right)
$$
Chapter 24: Marginalisation

Estimating $\sigma$ depends on $\mu$. 
Chapter 24: Marginalisation

Given the data, what is a good estimate for $\sigma$? This question requires marginalisation over $\mu$.

\[
p(\mu, \sigma|\{x_n\}_{n=1}^N) \propto (2\pi\sigma^2)^{-N/2} \exp \left( - \frac{N(\mu - \bar{x})^2 + S}{2\sigma^2} \right)
\]

\[
p(\sigma|\{x_n\}_{n=1}^N) = \int d\mu p(\mu, \sigma|\{x_n\}_{n=1}^N)
\]

\[
\int \exp \left( -N(\mu - \bar{x})^2 / 2\sigma^2 \right) = \sqrt{2\pi\sigma^2 / N}
\]

\[
p(\sigma|\{x_n\}_{n=1}^N) \propto (2\pi\sigma^2)^{-(N-1)/2} \exp \left( -S / 2\sigma^2 \right)
\]

The most likely value is $\sigma_{N-1} = \sqrt{S/(N-1)}$.

See bottom right figure previous page.
Chapter 27: Laplace method

When computing marginal:

\[ p(x_1) = \int dx_2 \ldots dx_N p(x_1, \ldots, x_N) \]

One can use the Laplace approximation.

From book pdf pg. 353.

NB: This does not solve the bias in the estimation of the parameters: the estimate from the Laplace approximation is the same as the maximum likelihood method.
Summary

Inference is parameter estimation can be done

• full Bayesian. Requires integration over model parameters
• or maximum likelihood approach. Requires optimization

Bayesian approach

• Explicit solution (Gaussian case, exponential models with conjugate priors)
• Monte Carlo sampling
• variational methods
• Laplace approximation (= similar to maximum likelihood)
Summary

Maximum likelihood approach (may be biased)

- Simple models explicit solution possible (Gaussian)
- Models with latent variables (clustering, but also, Boltzmann Machines, deep networks, etc) often use EM approach

EM maximized a lower bound on the likelihood via two operations

- E step computes a conditional distribution $p(x|y)$ on latent variable $x$ given observation $y$; use exact (clustering) MCMC or variational
- M step improves or maximizes (clustering) parameters
Chapter 29 and 30: Monte Carlo sampling

- Uniform sampling
- Importance and Rejection sampling
- Metropolis-Hasting method
- Detour on Markov processes, stationary distribution, ergodicity
- Gibbs sampling
- Hybrid Monte Carlo
The problems

1. generate samples \( \{x^r\}, r = 1, \ldots, R \) from \( p(x) \)

2. estimate

\[
\Phi = \langle \phi(x) \rangle = \int d^n x p(x) \phi(x)
\]

We focus on 1, since 1 solves 2:

\[
\hat{\Phi} = \frac{1}{R} \sum_r \phi(x^r)
\]

\[
\langle \hat{\Phi} \rangle = \Phi
\]

\[
\text{var}(\hat{\Phi}) = \frac{\sigma^2}{R}, \quad \sigma^2 = \int d^n x p(x)(\phi(x) - \Phi)^2
\]

when \( \{x^r\} \) independent.
Uniform sampling:

\[ \{x^r\}, \, r = 1, \ldots, R \]

requires \( a \) samples per dimension \( \rightarrow a^n \) samples.

For learning or inference, number of parameters

\[ n = 100 - 1000. \]
Consider the Ising model

\[ p(s|w) = \frac{\exp\left(\frac{1}{2} \sum_{ij} s_i s_j w_{ij}\right)}{Z} \]

\( s_i = \pm 1, \ i = 1, \ldots, n. \) This distribution is intractable to compute, due to the normalisation

\[ Z = \sum_{s_1} \ldots \sum_{s_n} \exp\left(\frac{1}{2} \sum_{ij} s_i s_j w_{ij}\right) \]

Total number of states is \( 2^n \), but most probability is concentrated in the so-called typical set \( T \).
**Typical set**

Consider $X$ from alphabet $A_X = \{a_1, \ldots, a_M\}$. Draw long string of $N$ values. Value $a_i$ occurs approximately $p_i N$ times.

Probability

\[
P(X_1, \ldots, X_N) \approx p_1^{p_1 N} \cdots p_M^{p_M N} = 2^N \sum_{j=1}^M p_j \log_2 p_j = 2^{-NH}
\]

with $H = - \sum_i p_i \log p_i$ the entropy of the distribution.

For $N = 1$, $P(X) \approx 2^{-H}$ and thus typical size $|T| \approx 2^H$.

Thus, in the case $X = (X_1, \ldots, X_n)$ with $X_i \pm 1$, the probability to hit the typical set is

\[
p = \frac{2^H}{2^n}
\]

If one draws $R$ samples uniform, the expected number of hits to the typical set is

\[
R_{\text{hit}} = R \frac{2^H}{2^n}
\]
To ensure that $R_{hit} \gg 1$ one thus finds

$$R \gg 2^{n-H}$$

What is $H$?

- For high temperature (noise) $H \approx n$. Uniform sampling feasible
- For low temperature $H \ll n \Rightarrow R = O(2^n)$

Uniform sampling only works for uniform distributions.
Sample from another distribution \( q(x) \) Often one can propose a sample density that is 1) better than uniform and 2) easy to sample from. For instance, a (spherical) Gaussian:

\[
q(x) \propto \exp\left(- \sum_i x_i^2/2\right)
\]
Consider simple 1-d sampling problem. Given $p(x)$, compute

$$a = \text{Prob}(x < 0) = \int_{-\infty}^{\infty} I(x)p(x)dx$$

with $I(x) = 0, 1$ if $x > 0, x < 0$, respectively.

Naive method: generate $N$ samples $X_i \sim p$

$$\hat{a} = \frac{1}{N} \sum_{i=1}^{N} I(X_i)$$
Importance sampling

Consider another distribution \( q(x) \). Then

\[
a = \text{Prob}(x < 0) = \int_{-\infty}^{\infty} I(x) \frac{p(x)}{q(x)} q(x) dx
\]

Importance sampling: generate \( N \) samples \( X_i \sim q \)

\[
\hat{a} = \frac{1}{N} \sum_{i=1}^{N} I(X_i) \frac{p(X_i)}{q(X_i)}
\]

Unbiased (= correct) for any \( q \)!
The distribution

\[ q^*(x) = \frac{p(x)I(x)}{a} \]

is the optimal importance sampler.

One sample \( X \sim q^* \) is sufficient to estimate \( a \):

\[ \hat{a} = I(X) \frac{p(X)}{q^*(X)} = a \]
Choose sufficiently broad importance sampler

Toy 1-d amino acid sampling problem, showing effect of poor importance sampler. Left: Gaussian suggests convergence after 500 iterations. Right: Cauchy is broader distribution and shows more robust behaviour.
Normalization

Typically, $p(x)$ can only be easily computed, up to a constant:

$$p(x) = \frac{p^*(x)}{Z} \quad Z = \int dx p^*(x)$$

For instance

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)} \quad p(D) = \int dw p(D|w)p(w)$$

Estimate both numerator and denominator by sampling.

$$\Phi = \int dx p(x) \phi(x) = \frac{\int dx p^*(x) \phi(x)}{\int dx p^*(x)} = \frac{\int dx q(x) \frac{p^*(x)}{q(x)} \phi(x)}{\int dx q(x) \frac{p^*(x)}{q(x)}}$$

Sample $\{x^r\}$ from $q(x)$ and compute

$$w_r = \frac{p^*(x^r)}{q(x^r)} \quad \hat{\Phi} = \frac{\sum_r w_r \phi(x^r)}{\sum_r w_r}$$
Importance sampling

The estimate is biased.

\[ \hat{\Phi} = \mathbb{E} \left( \frac{\sum_{r=1}^{N} w_r \phi(x^r)}{\mathbb{E} \sum_{r=1}^{N} w_r} \right) \neq \frac{\mathbb{E} \sum_{r=1}^{N} w_r \phi(x^r)}{\mathbb{E} \sum_{r=1}^{N} w_r} = \frac{N \int dx p^*(x) \phi(x)}{N \int dx p^*(x)} = \Phi \]

However, for large \( N \):

\[ \sum_{r=1}^{N} w_r = N \int dx p^*(x) + O(\sqrt{N}) \]

\[ \mathbb{E} \hat{\Phi} \approx \mathbb{E} \left( \frac{\sum_{r=1}^{N} w_r \phi(x^r)}{N \int dx p^*(x)} \right) = \frac{\mathbb{E} \sum_{r=1}^{N} w_r \phi(x^r)}{N \int dx p^*(x)} = \frac{N \int dx p^*(x) \phi(x)}{N \int dx p^*(x)} = \Phi \]

The estimator \( \hat{\Phi} \) is asymptotically unbiased.
Rejection sampling

Choose, \( c \) such that for all \( x \): \( cQ^*(x) > P^*(x) \)

- generate \( x \) from \( Q^*(x) \)
- generate \( u \) uniform from \([0, cQ^*(x)]\)
- if \( u > P^*(x) \) reject \( x \), otherwise accept \( x \)

This procedure samples \( P^*(x) \) because \((x, u)\) uniform from light grey area.

\[
\hat{\Phi} = \sum_r \phi(x^r) \rightarrow \int dx \phi(x)p(x)
\]
Let $p(x)$ and $q(x)$ be spherical Gaussians in $n$ dimensions with mean 0 and $\sigma_q = 1.01\sigma_p$. 
Since

\[
q(0) = \left(\frac{1}{\sqrt{2\pi\sigma_q^2}}\right)^n \quad p(0) = \left(\frac{1}{\sqrt{2\pi\sigma_p^2}}\right)^n
\]

then

\[
c = \frac{p(0)}{q(0)} = \left(\frac{\sigma_q}{\sigma_p}\right)^n = 1.01^n
\]

With \( n = 1000 \) we find \( c=20.000 \).

Acceptance rate = \( \frac{\text{volume}}{\text{volume} \cdot cq} = \frac{p}{c} = \frac{1}{c} \)

Thus rejection sampling is inefficient in high dimensions.

A similar argument holds for importance sampling.
The Metropolis algorithm (1956) considers a sampling density which depends on the current sample value: \( q(x|x^r) \).

Initialize in some random state \( x^1 \)
At iteration $r$, sample $x'$ from $q(x'|x^r)$ and compute

$$a = \frac{p^*(x')q(x^r|x')}{p^*(x^r)q(x'|x^r)}$$

If $a \geq 1$, accept $x'$ as the new state: $x^{r+1} = x'$

Else, accept $x'$ as the new state with probability $a$

If accept: $x^{r+1} = x'$, else $x^{r+1} = x^r$
Markov processes

Given a Markov process, when is the stationary distribution unique? What are the
dominant relaxation times? To answer these questions we make a brief digression.

Let $S$ denote the set of all state vectors $s$. $s \in S$ is a binary vector of length $n$ and
thus $s$ can take on $2^n$ different values. Therefore, $p_t(s)$ is a vector of length $2^n$ and
$T(s'|s)$ is a $2^n \times 2^n$ matrix.

Since $p_t(s)$ denotes a probability vector, it must satisfy $\sum_s p_t(s) = 1$. In addition,
$T(s'|s)$ is a probability vector in $s'$ for each value of $s$ and therefore each column
must add up to one:

$$\sum_{s'} T(s'|s) = 1.$$ 

Matrices with this property are called stochastic matrices.
Let us denote the eigenvalues and left and right eigenvectors of $T$ by $\lambda_\alpha, l_\alpha, r_\alpha, \alpha = 1, \ldots, 2^n$, respectively. In matrix notation we have

$$Tr_\alpha = \lambda_\alpha r_\alpha$$
$$l_\alpha^\dagger T = \lambda_\alpha l_\alpha^\dagger$$

Since $T$ is a non-symmetric matrix, the left and right eigenvectors are different, non-orthogonal and complex valued. $^\dagger$ denotes complex conjugation and transpose. The eigenvalues are complex valued. Under rather general conditions each set of eigenvectors spans a non-orthogonal basis of $C^{2n}$. These two bases are dual in the sense that:

$$l_\alpha^\dagger r_\beta = \delta_{\alpha\beta}.$$

We can therefore expand $T$ on the basis of its eigenvectors:

$$T = \sum_{\alpha=1}^{2^n} \lambda_\alpha r_\alpha l_\alpha^\dagger$$

\[1\] In general, the number of eigenvalues of $T$ can be less than $2^n$. However, for our purposes we can ignore this case.
At $t = 0$ the network is in a state $s^0$:

$$p_{t=0}(s) = \delta_{s,s^0}$$

At some later time:

$$p_t = T^t p_0 = \sum_{\alpha} \lambda^t_\alpha r_\alpha (l^\dagger_\alpha p_0)$$

Stationarity:

$$T p_\infty = p_\infty.$$  

Thus, the stationary distribution is a right eigenvector of $T$ with eigenvalue 1.

The time to approach stationarity is also given by the eigenvalues of $T$. In particular, each eigenvalue whose norm $|\lambda_\alpha| < 1$ corresponds to a transient mode with relaxation time

$$\tau_\alpha = \frac{-1}{\log \lambda_\alpha}.$$  

We will now study under which conditions $p_\infty$ is unique.
Some properties

A Markov process is called \textit{irreducible}, or \textit{ergodic}, on a subset of states $C \subset S$ if for any state $s \in C$ there is a finite probability to visit any other state $s' \in C$:

$$s = s^0, s^1, \ldots, s^k = s'$$

with $T(s^i|s^{i-1}) > 0$, $i = 1, \ldots, k$.

A subset of states $C \subset S$ is called \textit{closed} when the Markov process can never escape from $C$, once entered:

$$T(s'|s) = 0 \quad \text{for all} \quad s \in C, s' \not\in C.$$

In general, we can decompose the state space $S$ uniquely into closed irreducible subsets

$$S = \mathcal{T} \cup C_1 \cup C_2 \ldots,$$

where $\mathcal{T}$ is a set of \textit{transient states} and the $C_i$ are closed irreducible sets.
Perron-Frobenius theorem

An irreducible Markov process $T$ of periodicity $d$ has $d$ eigenvalues given by

$$\lambda_m = \exp(2\pi im/d), \quad m = 0, \ldots, d - 1,$$

and all remaining eigenvalues of $T$ are inside the unit circle in the complex plane: $|\lambda_\alpha| < 1$.

In particular, $T$ has exactly one eigenvalue 1. Its corresponding right eigenvector is equal to the (unique) stationary distribution (ergodicity).

The left eigenvector with eigenvalue 1 is $\propto (1, \ldots, 1)$. 
Non-ergodic behavior

A non-irreducible or non-ergodic Markov process has more than one eigenvalue 1 and therefore more than one left and right eigenvector with eigenvalue 1. Let us denote these eigenvectors by $l_1, \ldots, l_k$ and $r_1, \ldots, r_k$, respectively. Any linear combination of the right eigenvectors

$$p_\infty = \sum_{\alpha=1}^{k} \rho_\alpha r_\alpha$$

is therefore a stationary distribution, assuming $p_\infty(s) \geq 0$ for all $s$ and proper normalization: $\sum_s p_\infty(s) = 1$. Thus, there exists a manifold of dimension $k - 1$ of stationary distributions.
The $k$ left eigenvectors with eigenvalue 1 encode *invariants* of the dynamics. Define

$$L_\alpha(p_t) = l^\dagger_\alpha p_t, \alpha = 1, \ldots, k.$$  

Then $L_\alpha$ is invariant under the Markov dynamics:

$$L_\alpha(p_{t+1}) = l^\dagger_\alpha p_{t+1} = l^\dagger_\alpha T p_t = l^\dagger_\alpha p_t = L_\alpha(p_t).$$

- $k = 1$ ensures invariance of normalisation: $l_1 \propto (1, \ldots, 1)$

- the remaining $k - 1$ invariants are determined by the initial distribution $p_0$. They parametrize the stationary manifold and determine uniquely the stationary distribution.
Stationary distribution depends on initial state

From

\[ p_{\infty} = \sum_{\alpha=1}^{k} \rho_{\alpha} r_{\alpha} \]

\[ l_{\alpha}^\dagger r_{\beta} = \delta_{\alpha\beta} \]

we obtain

\[ L_{\alpha} = l_{\alpha}^\dagger p_0 = l_{\alpha}^\dagger p_{\infty} = \rho_{\alpha} \]

Thus, the stationary state depends on the initial state as:

\[ p_{\infty} = \sum_{\alpha=1}^{k} (l_{\alpha}^\dagger p_0) r_{\alpha} \cdot \]

Note, that in the ergodic case \((k = 1)\) the dependence on the initial state disappears, as it should, since \(l_1^\dagger p_0 = 1\) for any initial distribution \(p_0\)
Detailed balance

Stationarity:

\[ \sum_{s'} T(s|s')p(s') = p(s) = \sum_{s'} T(s'|s)p(s) \]

There always exists a stationary solution \( p(s) \).

The Markov process \( T \) satisfies detailed balance if \( \exists p \) such that

\[ T(s|s')p(s') = T(s'|s)p(s) \quad \text{for all} \quad s, s'. \]

Detailed balance is a property of the Markov dynamics \( T \). It is not true in general.

DB holds for thermodynamic systems and Metropolis Hastings, not for chaotic systems.

If DB \( p(s) \) is a stationary distribution of \( T \). The reverse is not true.
**Summary**

1. The transition matrix can be analysed in terms of its eigenvalues and eigenvectors. Eigenvalues close to 1, imply long convergence times.

2. Ergodic systems have unique stationary distribution.

3. Non-ergodic systems have multiple stationary distributions. Their asymptotic behaviour depends on the initial distribution.
Convergence of Metropolis algorithm

Metropolis algorithm is example of Markov process. Given two states $x$ and $x'$. Define

$$a_{x'x} = \frac{p^*(x')q(x|x')}{p^*(x)q(x'|x)}, \quad a_{xx'} = \frac{1}{a_{x'x}}$$

Suppose $a_{x'x} \geq 1$. Then

**Given** $x$, the probability to accept $x'$ is

$$T(x'|x) = q(x'|x)$$

**Given** $x'$, the probability to accept $x$ is

$$T(x|x') = q(x|x')a_{xx'}$$

$$\frac{T(x'|x)}{T(x|x')} = a_{x'x} \frac{q(x'|x)}{q(x|x')} = \frac{p^*(x')}{p^*(x)} = \frac{p(x')}{p(x)},$$

i.e. detailed balance. This implies that the process $T(x'|x)$ converges to $p(x)$. 
Metropolis Hasting for the Ising model

Use MH to sample from the Ising model

\[
p(x) = \frac{1}{Z} \exp(-E(x)) \quad E(x) = -\frac{1}{2} \sum_{i \neq j} w_{ij}x_i x_j = - \sum_{(ij)} w_{ij} x_i x_j
\]

Consider \(q(x'|x)\) to implement single spin flips. Define \(F_i x\) is the vector of spins obtained by flipping bit \(i\). Then

\[
q(F_i x|x) = \frac{1}{n} \quad q(x|x) = 0
\]

The MH ratio for the transition from state \(x\) to state \(F_i x\) is

\[
a_{x,F_i x} = \frac{p^*(F_i x)}{p^*(x)} = \exp(-\Delta E_{x,F_i x})
\]

\[
\Delta E_{x,F_i x} = E(F_i x) - E(x) = 2x_i \sum_{j \neq i} w_{ij} x_j
\]
Example: 2 spins

\[ E = -w x_1 x_2. \]

There are four states \( \alpha = 1 : 4 \) with energies:

\[ E_{1:4} = \{ E_{(++)}, E_{(+-)}, E_{(-+)}, E_{(--)} \} = \{ -w, w, w, -w \} \]

We define \( \Delta E_{\alpha\beta} = E_{\beta} - E_{\alpha} \):

\[
\Delta E = \begin{pmatrix}
0 & 2w & 2w & 0 \\
-2w & 0 & 0 & -2w \\
-2w & 0 & 0 & -2w \\
0 & 2w & 2w & 0
\end{pmatrix}
\]

We define \( q_{\alpha\beta} = q(\beta|\alpha) \):

\[
q = \frac{1}{2} \begin{pmatrix}
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0
\end{pmatrix}
\]
Example: 2 spins

Assume $w > 0$. We have for $\alpha \neq \beta$ $a_{\alpha,\beta} = \min(\exp(-2\Delta E_{\alpha,\beta}), 1)$:

$$a = \begin{pmatrix}
* & e^{-2w} & e^{-2w} & 1 \\
1 & * & 1 & 1 \\
1 & 1 & * & 1 \\
1 & e^{-2w} & e^{-2w} & *
\end{pmatrix}$$

For $\alpha \neq \beta$: $T_{\alpha,\beta} = T(\beta|\alpha) = a_{\alpha,\beta}q_{\alpha,\beta}$:

$$T = \begin{pmatrix}
1 - e^{-2w} & \frac{1}{2}e^{-2w} & \frac{1}{2}e^{-2w} & 0 \\
\frac{1}{2} & 0 & 0 & \frac{1}{2} \\
\frac{1}{2} & 0 & 0 & \frac{1}{2} \\
0 & \frac{1}{2}e^{-2w} & \frac{1}{2}e^{-2w} & 1 - e^{-2w}
\end{pmatrix}$$

Diagonal terms follow from $\sum_{\beta} T_{\alpha,\beta} = 1$.

Interpretation of $T$: from high energy states 2,3 ($(+, -)$ and $(-, +)$ spins transit with equal probability to one of the two low energy states 1,4 ($(+, +)$ and $(-, -)$). From low energy states 1,4 their is finite probability to transit to high energy states (decreasing with $w$).
Example: 2 spins

Eigenvalues of $T$ versus $w$. For low $w$ there is one eigenvalue 1 and is ergodic. For high $w$ there are two eigenvalues 1, system is non-ergodic.
Correlations slow down sampling

When \( q(x'|x) \) is Gaussian centered on \( x \), \( \frac{q(x'|x)}{q(x|x')} \) independent of \( x, x' \):

\[
a_{x'x} = \frac{p^*(x')}{p^*(x)}
\]

\( \epsilon \) large:
Acceptance rate \( a_{x'x} \) small.

\( \epsilon \) small:
Strong dependence on starting value. Many samples needed to sample.
Correlations slow down sampling

With step $\epsilon$ random, the particle moves a distance $L \approx \sqrt{T\epsilon}$ in $T$ iterations.

If largest length scale is $L$ then

$$T \approx \left(\frac{L}{\epsilon}\right)^2$$
Gibbs sampling

- Consider only change of one element of \((x_1, \ldots, x_n)\) at the time. Choose \(i\) random and update component \(i\) according to

\[
q(x'|x) = p(x'_i|x_{-i}) \delta_{x_i,x'_i}
\]

- Accept: \(x^{r+1} = x'\)

The one dimensional sampling can be done using for instance Rejection sampling.
Gibbs sampling

Each Gibbs sampling is a MH step with acceptance \( a = 1 \):

\[
q(x'|x) = p(x'_i|x_{-i}) \delta_{x_i,x'_i}
\]

\[
a = \frac{p(x')q(x|x')}{p(x)q(x'|x)} = \frac{p(x'_i|x_{-i})p(x'_{-i})p(x_i|x_{-i})}{p(x_i|x_{-i})p(x_{-i})p(x'_i|x_{-i})} = \frac{p(x'_i|x_{-i})p(x_i|x_{-i})}{p(x_i|x_{-i})p(x'_i|x_{-i})} = 1
\]
Gibbs sampling

Convergence slow when variables correlated: When two variables have marginal width $L$ and conditional width $\epsilon$, the number of iterations scales as $T \approx L^2 / \epsilon^2$. 
The Hybrid Monte Carlo Method

Let

\[ P(x) = \frac{e^{-E(x)}}{Z} \]

with \( E \) and its gradient \( \frac{\partial E}{\partial q_i} \) easy to compute.

Gradient information reduces random walk behaviour in Metropolis method.

Double the state space by introducing for each \( x_i \) a momentum \( p_i \). Define the Hamiltonian

\[ H(p, x) = E(x) + \frac{\alpha}{2} \sum_i p_i^2 \]

\[ P_H(p, x) = \frac{1}{Z_H} \exp \left( -E(x) - \frac{\alpha}{2} \sum_i p_i^2 \right) \]

The marginal is \( p(x) = \int dp P_H(p, x) \).
Sample from $P_H$ using the Hamiltonian dynamics and Metropolis-Hasting:

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial x_i}, \quad \frac{dx_i}{dt} = \frac{\partial H}{\partial p_i}$$

leaves $H$ invariant.

$$\frac{dH}{dt} = \sum_i \frac{dH}{dp_i} \frac{dp_i}{dt} + \frac{dH}{dx_i} \frac{dx_i}{dt} = 0$$
Pseudo code

Choose initial $x^1$.

For $t = 1 : T$:

1. choose $p^t$ from $\mathcal{N}(0, \alpha^{-1})$, giving $(x^t, p^t)$

2. run Hamilton dynamics, giving $(x', p')$

3. Metropolis step: accept $(x^{t+1}, p^{t+1}) = (x', p')$ as new state with probability

$$a = \frac{p^*(x')}{p^*(x)} = \frac{e^{-H(x', p')}}{e^{-H(x, p)}} \min (1, a)$$

4. On rejection, $(x^{t+1}, p^{t+1}) = (x^t, p^t)$
Pseudo code

g = gradE (x); \hspace{1cm} # set gradient using initial x
E = findE (x); \hspace{1cm} # set objective function too

for \( l = 1:L \) \hspace{1cm} # loop \( L \) times
\hspace{1.5cm} p = randn (size(x)); \hspace{1cm} # initial momentum is Normal(0,1)
\hspace{1.5cm} H = p' \ast p / 2 + E; \hspace{1cm} # evaluate \( H(x,p) \)

xnew = x; gnew = g;
for \( \tau = 1:\text{Tau} \) \hspace{1cm} # make \( \text{Tau} \) 'leapfrog' steps
\hspace{2.5cm} p = p - \epsilon \ast gnew / 2; \hspace{1cm} # make half-step in \( p \)
\hspace{2.5cm} xnew = xnew + \epsilon \ast p; \hspace{1cm} # make step in \( x \)
\hspace{2.5cm} gnew = gradE (xnew); \hspace{1cm} # find new gradient
\hspace{2.5cm} p = p - \epsilon \ast gnew / 2; \hspace{1cm} # make half-step in \( p \)
endfor

Enew = findE (xnew); \hspace{1cm} # find new value of \( H \)
Hnew = p' \ast p / 2 + Enew;
dH = Hnew - H; \hspace{1cm} # Decide whether to accept

if (dH < 0) \hspace{1cm} accept = 1;
elseif (rand() < \exp(-dH)) \hspace{1cm} accept = 1;
else \hspace{1cm} accept = 0;
endif

if (accept)
g = gnew; \hspace{1cm} x = xnew; \hspace{1cm} E = Enew;
endif
endfor
Consider the double well cost \( E(x) = (x^2 - 1)^2 \).

The Hamiltonian \( H(x, p) = E(x) + \frac{1}{2}p^2 \) has the form.

\( \alpha \) large yields large \( p \) values and dynamical trajectories to large distances.
Example 2

\[ H(x, p) = \frac{1}{2} \beta_1 x_1^2 + \frac{1}{2} \beta_2 x_2^2 + \frac{1}{2} (p_1^2 + p_2^2) \]

The dynamics is

\[
\begin{align*}
\dot{x}_i &= \frac{\partial H(x, p)}{\partial p_i} = p_i \\
\dot{p}_i &= -\frac{\partial H(x, p)}{\partial x_i} = -\beta_i x_i \\
\ddot{x}_i &= -\beta_i x_i
\end{align*}
\]

The solution is of the form \( x_i(t) = \sin\left(\sqrt{\beta_i} t\right) \).

If \( \beta_1 \gg \beta_2 \), oscillations in \( x_1 \) much faster than in \( x_2 \).
Comparison of HMC and Metropolis

Two dimensional elongated Gaussian distribution. a-b) Hybrid Monte Carlo method c-d) Metropolis method.
Perceptron

\[ p(t = 1 | x, w) = \sigma(\vec{w} \cdot \vec{x}) \]

\[ \sigma(x) = \frac{1}{1 + \exp(-2x)} \]

\[ \vec{w} \cdot \vec{x} = w_0 + w_1 x_1 + w_2 x_2 \]

\[ \mathbf{w} = (0, 2) \]
Perceptron

\[ w = (-2, 3) \]
\[ w = (1, 4) \]
\[ w = (5, 4) \]
\[ w = (0, 2) \]
\[ w = (2, 2) \]
\[ w = (5, 1) \]
\[ w = (1, 0) \]
\[ w = (3, 0) \]
\[ w = (2, -2) \]
Ch. 39

Perceptron

\[ p(t = 1|x, w) = \sigma(\vec{w} \cdot \vec{x}) \]

\[ G(w) = -\sum_{\mu} \log(p(t^\mu|x^\mu, w)) \]

\[ M(w) = G(w) + \alpha E_w(w) \]

\[ E_w(w) = \sum_i w_i^2 \]

\[ p(w|D) = \frac{p(D|w)p(w)}{p(D)} \propto \exp(-M(w)) \]
The maximum posterior solution

Minimizing the cost function $M(w)$ yields smoother solutions for larger $\alpha$. 
Learning as inference

Data set: \( \{x^\mu, t^\mu\}, \mu = 1, \ldots, P \)

Probability of data point under the model: \( p(t^\mu|x^\mu, w) \)

Likelihood:

\[
p(D|w) = \prod_{\mu} p(t^\mu|x^\mu, w) = \exp(-G(w)) \quad G(w) = - \sum_{\mu} \log(p(t^\mu|x^\mu, w))
\]

Prior:

\[
p(w) = \frac{\exp(-\alpha E_w(w))}{Z_w(\alpha)} \quad E_w(w) = \sum_i w_i^2
\]

makes solutions with small weights more probable.

Posterior:

\[
p(w|D) = \frac{p(D|w)p(w)}{p(D)} \propto \exp(-M(w))
\]

\[
M(w) = G(w) + \alpha E_w(w)
\]
ML versus Bayesian

Standard in neural network learning is to compute the maximum likelihood or maximum posterior solution. For new test point $a$

$$D \rightarrow w_{ml}$$

$$p(t|a) = p(t|a, w_{ml})$$

Bayesian approach requires integration over multiple solutions:

$$D \rightarrow p(w|D)$$

$$p(t|a) = \int \! dw p(w|D)p(t|a, w) = \langle p(t|a, w) \rangle_{p(w|D)}$$
The full Bayesian solution

<table>
<thead>
<tr>
<th>Data set</th>
<th>Likelihood</th>
<th>Probability of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N = 0$</td>
<td>(constant)</td>
<td><img src="image1" alt="Likelihood" /></td>
</tr>
<tr>
<td>$N = 2$</td>
<td><img src="image2" alt="Likelihood" /></td>
<td><img src="image3" alt="Probability" /></td>
</tr>
<tr>
<td>$N = 4$</td>
<td><img src="image4" alt="Likelihood" /></td>
<td><img src="image5" alt="Probability" /></td>
</tr>
<tr>
<td>$N = 6$</td>
<td><img src="image6" alt="Likelihood" /></td>
<td><img src="image7" alt="Probability" /></td>
</tr>
</tbody>
</table>
The full Bayesian solution

\[ \alpha = 0.01, \quad q(w'|w) = \mathcal{N}(w'|w, \sigma), \quad \sigma = 0.1 \]

\[ p(t|x) = \int dw p(t|x, w)p(w|D) \approx \frac{1}{R} \sum_r p(t|x, w^r) \]
Ch. 31 Ising model

Ising model is a probability distribution over vectors of binary variables. \( x = (x_1, \ldots, x_N) \) with \( x_i = \pm 1 \).

\[
p(x) = \frac{1}{Z} \exp(-\beta E(x)) \quad E(x) = -\frac{1}{2} \sum_{i \neq j} J_{ij} x_i x_j - \sum_i h_i x_i
\]

\[
Z = \sum_{x} \exp(-\beta E(x)) = \sum_{x_1} \ldots \sum_{x_N} \exp(-\beta E(x))
\]

\( \beta = 1/kT \) is the inverse temperature, can be absorbed in the coupling.\(^2\)

Why study the Ising model:
- phase transition: qualitative change when going from high temperature to low temperature
- relation to attractor neural networks (hopfield models) and Boltzmann Machines

\(^2\)It is sometimes convenient to write \( \frac{1}{2} \sum_{i \neq j} J_{ij} x_i x_j = \sum_{(ij)} J_{ij} x_i x_j \).
Some basic relations

The mean energy of the spin system \( \langle E \rangle = \sum_x E(x)p(x) \) can be computed from \( Z \):

\[
\frac{\partial \log Z}{\partial \beta} = \frac{1}{Z} \frac{\partial Z}{\partial \beta} = \frac{1}{Z} \sum_x -E(x) \exp(-\beta E(x)) = -\langle E \rangle
\]

The energy fluctuations are computed similarly:

\[
\frac{\partial^2 \log Z}{\partial \beta^2} = \frac{\partial}{\partial \beta} \left( \frac{1}{Z} \frac{\partial Z}{\partial \beta} \right) = -\frac{1}{Z^2} \left( \frac{\partial Z}{\partial \beta} \right)^2 + \frac{1}{Z} \frac{\partial^2 Z}{\partial \beta^2}
\]

\[
= -\langle E \rangle^2 + \frac{1}{Z} \sum_x E(x)^2 \exp(-\beta E(x)) = \langle E \rangle^2 + \langle E^2 \rangle = \text{var}(E)
\]

One calls \( F = -\frac{1}{\beta} \log Z \) the free energy, or a log partition sum.

\[
F = \langle E \rangle - \frac{1}{\beta} H \quad \text{ex.}31.1
\]
Metropolis Hasting for the Ising model

Use MH to sample from the Ising model

\[ p(x) = \frac{1}{Z} \exp(-E(x)) \]
\[ E(x) = -\sum_{(i,j)} w_{ij} x_i x_j \]

Consider \( q(x'|x) \) to implement single spin flips. Define \( F_{i} x \) is the vector of spins obtained by flipping bit \( i \). Then

\[ q(F_{i} x| x) = \frac{1}{n} \quad q(x|x) = 0 \]

The MH ratio for the transition from state \( x \) to state \( F_{i} x \) is

\[ a_{x, F_{i} x} = \frac{p^{*}(F_{i} x)}{p^{*}(x)} = \exp(-\Delta E_{x, F_{i} x}) \]
\[ \Delta E_{x, F_{i} x} = E(F_{i} x) - E(x) = 2x_i \sum_{j \neq i} w_{ij} x_j \]
Ferro-magnet

Rectangular grid with $J = 1$ and $h_i = 0$ Ferromagnet, periodic boundary.

Run system at different temperatures $\beta = 0.03 \to 10 \to 0.03$ (check for hysteresis)
Run $T$ iterations at each temperature, discard first 1/3 for 'burn in'.
record $\langle E \rangle, \text{var}(E), \langle m^2 \rangle$ as time averages:

$$\langle E \rangle = \frac{1}{T} \sum_{t=1}^{T} E(x_t)$$

$$\langle m^2 \rangle = \frac{1}{T} \sum_{t=1}^{T} m^2(x_t) \quad m(x_t) = \frac{1}{N} \sum_{i=1}^{N} x_i$$
Ferro-magnet

Left: Mean energy. Each spin has 4 links, thus $E$ has $2N$ terms.
- at low temperature $\langle E \rangle / N = -2$ (for one of the two ground states).
- At high temperature $\langle E \rangle / N = 0$ and $\text{var}(E/N) \propto 1/N$.
- at intermediate temperature $\text{var}(E/N)$ increases with $N$

Right: Mean squared magnetisation
- is zero for high temperature
- and 'breaks' to $\pm 1$ for low temperature.
Ferro-magnet

Energy fluctuations increase around the critical temperature
Anti-ferromagnet

Anti-ferromagnet has $J = -1$ between neighbors on the grid.

Ground states of the anti-ferromagnet are one-to-one with ground states in ferromagnet.

We can divide the spins into two groups $B, W$ according to the checker board.

$$E_{\text{anti}}(x_B, x_W) = - \sum_{i \in B} \sum_{j \in W} (J = -1)x_i x_j = - \sum_{i \in B} \sum_{j \in W} (J = 1)x_i (x'_j = -x_j)$$

$$= E_{\text{ferro}}(x_B, -x_W) = E_{\text{ferro}}(x_B, x_B)$$

The ferro and anti-ferro magnetic system have the same distribution of energies $E(x)$ by relabeling $x$. 
Anti-ferromagnet

When the number of spins per direction is odd, this is no longer true.

Rectangular grid of $5 \times 5$ spins with ferromagnetic (left) and anti-ferromagnetic (right) interaction.

Not all bonds $J_{x_i x_j}$ can be minimized. This effect decreases with system size.
Frustration: Anti-ferromagnetic triangular Ising model

When $J = -1$, $E(x)$ cannot be minimized by minimizing all link energies $-Jx_ix_j = x_ix_j$.

Of the eight possible configurations of three spins, six have energy $-1$ and two have energy $+3$.

As a result: no 'clear cut' unique minimal energy state(s) but many states with similar low energy (Spin glass).
Frustration: Anti-ferromagnetic triangular Ising model

Energy (top), fluctuations (middle), heat capacity (bottom) for ferromagnet $J = 1$ (left) and anti-ferromagnet ($J = -1$ on triangular lattice).

Note: different low energies due to frustration, absence of peak in $\text{var}(E)$ indicates no phase transition to state with long range order.
Direct computation of partition function

The summation of the partition sum $Z$ can be performed in the 2-dimensional Ising model on a lattice of size $W \times C$.

$$Z = \sum_x \exp(-\beta E(x))$$

$$= \sum_{s_1} \sum_{s_2} \ldots \sum_{s_C} \exp \left( -\beta \sum_{c=1}^C E(s_c, s_{c+1}) \right)$$

$s_c$ is a binary vector of length $W$, $\sum_{s_c}$ contains $2^W$ terms. The energy is

$$E(s_c, s_{c+1}) = \sum_{i \in c, j \in c+1} J x_i x_j + \frac{1}{4} \sum_{i, j \in c} J x_i x_j + \frac{1}{4} \sum_{i, j \in c+1} J x_i x_j$$

$$Z = \sum_{s_1} \sum_{s_2} \ldots \sum_{s_C} M_{s_1, s_2} M_{s_2, s_3} \ldots M_{s_{C-1}, s_C} M_{s_C, s_1} = \sum_{s_1} (M^C)_{s_1, s_1}$$

$$= \text{Trace}(M^C) \quad M_{s_c, s_{c+1}} = \exp(-\beta E(s_c, s_{c+1}))$$
Direct computation of partition function

\[
\lim_{C \to \infty} Z = \lim_{C \to \infty} \text{Trace}(M^C) = \lim_{C \to \infty} \sum_{\alpha} \mu_{\alpha}^C = \mu_{\text{max}}^C
\]

with \( \mu_{\text{max}} \) the largest eigenvalue of \( M \).

Free energy per spin

\[
f = -\frac{1}{\beta W C} \log Z = -\frac{1}{\beta W} \log \mu_{\text{max}}
\]

1-3: \( F = \langle E \rangle - \frac{1}{\beta} H \), high temperature \( \langle E \rangle \approx 0 \) and \( f \approx -\frac{1}{\beta} \log 2 \). Entropy \( H = -\frac{\partial F}{\partial T} \), (Ex. 31.1) low temperature slope is zero for ferro and non-zero for triang. anti-ferro.

4: Mean energy of frustrated system at low temperature is higher.
Direct computation of partition function

\[ \text{var}(E) = \frac{\partial^2 \log Z}{\partial \beta^2} \] versus temperature shows signs of phase transition for rectangular and triangular ferromagnet, but not for the frustrated triangular anti-ferromagnet.
Simulated annealing

Another way to accelerate convergence is by simulated annealing. Consider

\[ p(x) = \frac{1}{Z} \exp(-E(x)) \quad \rightarrow \quad p_T(x) = \frac{1}{Z_T} \exp(-E(x)/T) \]

Initially use large \( T \) and decrease to \( T = 1 \) (‘cooling’ from high to low temperature).
Exercise Simulated annealing

Write your own SA algorithm to minimize \( E(x) = -\frac{1}{2} x' w x = -\sum_{ij} x_i x_j w_{ij} \).

If \( w_{ij} > 0 \) random, optimization is easy (ferromagnetic). There are two global minima.
If \( w_{ij} < 0 \) random, optimization is hard (spin glass). There are many minima of approx equal quality.

Initial temperature is set \( T_0 = \max(dE) \), where \( dE = E(x') - E(x) \) when \( x \to x' \). As a result,

\[
a = \frac{p^*(x')}{p^*(x)} = \exp \left( -\frac{dE}{T_0} \right) > \exp(-1)
\]

\( \beta_0 = 1/T_0 \). Use \( T = 1000 \) samples at each temperature.

while \( \text{std}(E) > 0 \) do
- generate \( x_{1:T} \). Compute \( E(x_{1:T}) \).
- Compute mean and std of \( E \).
- \( \beta = 1.05\beta \)
Variational methods

Recall the Kulback-Leibler divergence (aka relative entropy) between two distributions $p, q$ over the same domain $\mathcal{A}$:

$$KL(q|p) = \sum_{x \in \mathcal{A}} q(x) \log \frac{q(x)}{p(x)}$$

Properties:

$$KL(q|p) \neq KL(p|q) \quad KL(q|p) \geq 0 \quad KL(q|p) = 0 \iff p = q$$

Often, distributions are of the form $p(x) = \frac{p^*(x)}{Z}$, where $p^*(x)$ is easy to evaluate for any $x$, but $Z = \sum_x p^*(x)$ is intractable. \(^3\)

Efficient approximation of $Z$ and other ‘statistics’ of interest.

\(^3\)For instance, the Ising model:

$$p(x) = \frac{1}{Z} \exp(-\beta E(x)) \quad E(x) = -\frac{1}{2} \sum_{i \neq j} J_{ij} x_i x_j - \sum_i h_i x_i$$

$$Z = \sum_x \exp(-\beta E(x))$$
Jensen’s inequality

A function $f$ is convex if

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2)$$

This generalises to

$$f\left(\sum_{i=1}^{n} \lambda_i x_i\right) \leq \sum_{i=1}^{n} \lambda_i f(x_i) \quad \sum_{i} \lambda_i = 1$$

$$f\left(\sum_{x} q(x)x\right) \leq \sum_{x} q(x)f(x) \quad f(\mathbb{E}x) \leq \mathbb{E}f(x)$$
Variational methods

Since $- \log(x)$ is a convex function:

$$F = - \log Z = - \log \sum_x p^*(x) = - \log \sum_x q(x) \frac{p^*(x)}{q(x)} \leq - \sum_x q(x) \log \left( \frac{p^*(x)}{q(x)} \right)$$

$$= \sum_x q(x) \log \left( \frac{q(x)}{p^*(x)} \right) = \tilde{F}(q)$$

$F$ is the free energy and $\tilde{F}(q) \geq F$ is the variational free energy.

Note the following properties:

$$\tilde{F}(q) = \sum_x q(x) \log q(x) - \sum_x q(x) \log p^*(x) = \langle E \rangle_q - S(q) \quad p^*(x) = \exp(-E(x))$$

$$\tilde{F}(q) = \sum_x q(x) \log \left( \frac{q(x)}{p(x)} \right) - \sum_x q(x) \log Z = KL(q|p) + F$$

When $q = p$, $\tilde{F}(q) = F$. Approach is to minimise $\tilde{F}$ with $q$ 'simple' distributions.
Variational methods

In general $\tilde{F}(q)$ is as hard to compute as $F$, but simplifies
- for certain $q$: factorized, sparse, or Gaussian
- for certain $p$: if $p(x) \propto \prod_{\alpha} \psi_{\alpha}(x_{\alpha})$, $\alpha$ is small subset of variables
The (naive) mean field approximation for spin system

The simplest possible $q$ is factorized:

$$ q(x) = \prod_{i=1}^{n} q_i(x_i) \quad q_i(x) = \frac{\exp(a_i x)}{\exp(a_i x) + \exp(-a_i x)} = \sigma(a_i x) $$

The variational free energy $\tilde{F}(q) = \langle E \rangle_q - S$.

$$ \langle E \rangle = - \sum_{(i,j)} w_{ij} \langle x_i x_j \rangle - \sum_i h_i \langle x_i \rangle $$

$$ \langle x_i \rangle = q_i(1) - q_i(-1) = \tanh(a_i) \equiv m_i $$

$$ \langle x_i x_j \rangle = \langle x_i \rangle \langle x_j \rangle = m_i m_j $$

$$ \langle E \rangle = - \sum_{(i,j)} w_{ij} m_i m_j - \sum_i h_i m_i $$

We can express $q_i(x) = \frac{1}{2} \left( 1 + m_i x_i \right)$. 

Bert Kappen

ML 117
The (naive) mean field approximation for spin system

\[ S = - \sum_x q(x) \log q(x) = - \sum_x \frac{1}{n} \sum_{i=1}^n \log q_i(x_i) \]

\[ = - \sum_{i=1}^n \sum_x q(x) \log q_i(x_i) = - \sum_{i=1}^n \sum_{x_i} q_i(x_i) \log q_i(x_i) \]

\[ \sum_{x_i} q_i(x_i) \log q_i(x_i) = \frac{1}{2} \left( (1 + m_i) \log \frac{1}{2}(1 + m_i) + (1 - m_i) \log \frac{1}{2}(1 - m_i) \right) \]

\[ 4 \text{ Where we used } \sum_x q(x) h(x_i) = \sum_{x_i} \sum_{x_{-i}} q(x, x_{-i}) h(x_i) = \sum_{x_i} q(x_i) h(x_i) = \sum_{x_i} q_i(x_i) h(x_i) \]
The variational free energy is a function of $m_1:n$:

$$\tilde{F}(m) = -\sum_{(ij)} w_{ij}m_im_j - \sum_i h_im_i$$

$$+ \sum_i \frac{1}{2} \left( (1 + m_i) \log \frac{1}{2}(1 + m_i) + (1 - m_i) \log \frac{1}{2}(1 - m_i) \right)$$

We find the minimum by differentiation:

$$\frac{\partial \tilde{F}(m)}{\partial m_i} = -\sum_j w_{ij}m_j - h_i + \frac{1}{2} \log \frac{1 + m_i}{1 - m_i}$$

It is easy to show that $\frac{1}{2} \log \frac{1 + m_i}{1 - m_i} = a_i = \tanh^{-1}(m_i)$. Thus,

$$m_i = \tanh \left( \sum_j w_{ij}m_j + h_i \right)$$

are the mean field equations.
The (naive) mean field approximation for spin system

The MF equation are $n$ coupled non-linear equations with $n$ unknown

MF free energy $\tilde{F}(m)$ vs. $(m_1, m_2)$ for $E(x) = -x_1 x_2$. NB: $\tilde{F}(m_1, m_2)$ convex in $m_1$ for fixed $m_2$ and visa versa, but not convex in $(m_1, m_2)$.

There may be more than one solution (local minima)
The (naive) mean field approximation for spin system

Solve by coordinate ascent/sequential updating:
- choose $i$ (random or fixed order), compute new $m'_i = \tanh \left( \sum_j w_{ij}m_j + h_i \right)$.
- Then $\tilde{F}(m_{-i}, m'_i) \leq \tilde{F}(m_{-i}, m_i)$.
Guaranteed convergence to local minimum.

Fixed point iteration/parallel updating:
- for all $i$ compute new $m'_i = \tanh \left( \sum_j w_{ij}m_j + h_i \right)$.
Generally faster (in Matlab), but no guaranteed convergence.
The (naive) mean field approximation the Ising model

In the case of the 2d Ising model, all couplings $w_{ij} = w$ for neighbouring spins and zero otherwise and the external field $h_i = h$. The variational distribution has now only a single variable:

$$q(x|a) = \frac{1}{Z} \exp(\sum_i ax_i) \quad m = \tanh(a)$$

The mean field equation becomes

$$m = \tanh(Cwm + h)$$

with $C$ the number of neighbours in the lattice (4 for square lattice).

MF estimate of critical temperature is $T_c = 4$, exact value is $T_c = 2.27$.

For $T > T_c$ there $\tilde{F}(m)$ is convex and has one minimum, for $T < T_c$ $\tilde{F}(m)$ is not convex and has 2 minima.
The (naive) mean field approximation the Ising model

Comparison of MF approximation to exact solution for $8 \times 8$ ferromagnetic Ising model. MF free energy is higher than true free energy. Energy fluctuations are not well predicted.

Mean field approximation improves when number of connections increase, and becomes exact for the fully connected system in the large $N$ limit.
Variational methods in inference and data modelling

In Bayesian learning we consider the posterior:

\[
p(w|D) = \frac{p(D|w)p(w)}{p(D)}
\]

Posterior is used for evaluation on new data, or model comparison

\[
p(y|x, D) = \int dw p(y|x, w)p(w|D) \quad p(D) (= p(D|M_i)) = \int dw p(D|w)p(w)
\]

We define the variational approximation by introduction the variational distribution \( q(w) \) and minimize

\[
\tilde{F}(q) = \int dw q(w) \log \frac{q(w)}{p(D|w)p(w)} = \int dw q(w) \log \frac{q(w)}{p(w|D)} - \log p(D)
\]

\( \tilde{F} \) is minimized by \( q(w) = p(w|D) \) with value \(-\log p(D)\).
Example: the Gaussian likelihood

Given the Gaussian distribution

\[ p(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) \]

posterior for \( N \) data points \( D = \{x_n\}_{n=1}^N \) is

\[ p(\mu, \sigma|D) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{N(\mu - \bar{x})^2 + S}{2\sigma^2}\right) \frac{1}{\sigma} \]

\[ p(D) \]

with \( \bar{x} = \frac{1}{N} \sum_n x_n \), \( S = \sum_n (x_n - \bar{x})^2 \), flat prior on \( \mu \) and \( 1/\sigma \) 'non-informative prior' on \( \sigma \).
Gamma prior

The Gamma distribution is defined for positive quantities:

\[ \Gamma(x|c, s) = \frac{1}{Z} \left( \frac{x}{s} \right)^{c-1} \exp \left( -\frac{x}{s} \right) \quad Z = \Gamma(c)s \]

When \( y = \log x \) then

\[ p(y) = p(x) \left| \frac{dx}{dy} \right| = \frac{1}{\Gamma(c)} \left( \frac{x(y)}{s} \right)^{c} \exp \left( -\frac{x(y)}{s} \right) \]

\( \Gamma(x|c, s) \) may be peaked at \( x = 0 \) this peak is absent in the log basis.

When \( c \to 0 \) with \( sc = 1 \).

\[ \Gamma(x|c, s) = \frac{1}{Z} \left( \frac{x}{s} \right)^{c-1} \exp \left( -\frac{x}{s} \right) = \frac{1}{Z} c^{c-1} \left( \frac{x}{sc} \right)^{c-1} \exp \left( -\frac{x}{sc} \right) \propto x^{-1} \]

The non-informative prior has no scale \( s \) and is 'flat' in the log basis.
Variational methods in inference and data modelling

We approximate the posterior by a factorized variational distribution $q_\mu(\mu)q_\sigma(\sigma)$.

$$\tilde{F}(q) = \int d\mu q_\mu(\mu) \log q_\mu(\mu) + \int d\sigma q_\sigma(\sigma) \log q_\sigma(\sigma)$$

$$- \int d\mu d\sigma q_\mu(\mu)q_\sigma(\sigma) \log p(D|\mu, \sigma) \frac{1}{\sigma}$$

$$= \int d\mu q_\mu(\mu) \log q_\mu(\mu) + \int d\sigma q_\sigma(\sigma) \log q_\sigma(\sigma)$$

$$- \int d\mu d\sigma q_\mu(\mu)q_\sigma(\sigma) \left( \log \frac{1}{(2\pi \sigma^2)^{N/2}} \frac{1}{\sigma} - \frac{N(\mu - \bar{x})^2 + S}{2\sigma^2} \right)$$

$$= \int d\mu q_\mu(\mu) \log q_\mu(\mu) + \int d\sigma q_\sigma(\sigma) \log q_\sigma(\sigma)$$

$$+ \int d\sigma q_\sigma(\sigma) \frac{N + 1}{2} \log \sigma^2 + \int d\mu q_\mu(\mu) \frac{\bar{\beta}}{2} \left( (N\mu - \bar{x})^2 + S \right) + k$$

with $\bar{\beta} = \int d\sigma q_\sigma(\sigma) \frac{1}{\sigma^2}$. 
Minimize wrt $q_{\mu}$ with Lagrange multiplier $\lambda \left( \int d\mu q_{\mu}(\mu) - 1 \right)$:

$$\frac{\partial \tilde{F}}{\partial q_{\mu}(\mu)} = \log q_{\mu}(\mu) + 1 + \frac{\bar{\beta}}{2} \left( N(\mu - \bar{x})^2 + S \right) + \lambda$$

$$q_{\mu}(\mu) = \mathcal{N}\left( \mu | \bar{x}, \sigma^2_{\mu|D} = \frac{1}{N\bar{\beta}} \right)$$

Minimizing wrt $q_{\sigma}$ with Lagrange multiplier $\lambda \left( \int d\sigma q_{\sigma}(\sigma) - 1 \right)$:

$$\frac{\partial \tilde{F}}{\partial q_{\sigma}(\sigma)} = \log q_{\sigma}(\sigma) + 1 + (N + 1) \log \sigma + \int d\mu q_{\mu}(\mu) \frac{N(\mu - \bar{x})^2 + S}{2\sigma^2} + \lambda$$

$$= \log q_{\sigma}(\sigma) + 1 + (N + 1) \log \sigma + \frac{1}{2} \beta \left( N\sigma^2_{\mu|D} + S \right) + \lambda$$

Define $q_{\beta}(\beta) = q_{\sigma}(\sigma(\beta)) \left| \frac{d\sigma}{d\beta} \right|$. Then $q_{\sigma}(\sigma) = 2 \sqrt{2\beta^{3/2}} q_{\beta}(\beta)$:

$$= \log q_{\beta}(\beta) + \left( -\frac{N}{2} + 1 \right) \log \beta + \frac{1}{2} \beta \left( N\sigma^2_{\mu|D} + S \right) + \lambda + k$$
Thus,

$$
\log q_\beta(\beta) \propto (c' - 1) \log \beta - \frac{\beta}{b'}
$$

$$
c' = \frac{N}{2}, \quad \frac{1}{b'} = \frac{1}{2}\left(N\sigma_{\mu|D}^2 + S\right)
$$

$$
q_\beta(\beta) = \Gamma(\beta | b', c')
$$

Solid: $p(\mu, \sigma|D)$. Dashed: Iterative optimization of $q_\mu (b,d)$ and $q_\sigma (c,e)$ from initial (a) converges in 15 iterations (f)
The variational view on EM

Consider the joint distribution of a clustering problem

\[
p(x_{1:N}, k_{1:N}, \theta) = p(\theta) \prod_{n=1}^{N} (p(x_n|k_n, \theta)p(k_n|\theta))
\]

\(x_{1:N}\) are the observed data. \(k_{1:N}\) are the latent (missing) labels.

The posterior over \(\theta, k_{1:N}\) is proportional to the joint probability:

\[
p(k_{1:N}, \theta|x_{1:N}) \propto p(x_{1:N}, k_{1:N}, \theta)
\]

We define a variational free energy as usual:

\[
\tilde{F}(q_k, q_\theta) = \sum_{k_{1:N}} \int d\theta q_k(k_{1:N})q_\theta(\theta) \log \frac{q_k(k_{1:N})q_\theta(\theta)}{p(x_{1:N}, k_{1:N}, \theta)}
\]
By 'coordinate descent': minimize wrt \( q_k \) for fixed \( q_\theta \) and minimize wrt \( q_\theta \) for fixed \( q_k \).

We assume \( q_\theta(\theta) = \delta(\theta - \theta^*) \). Then

\[
\tilde{F}(q_k, \theta^*) = \sum_{k_{1:N}} q_k(k_{1:N}) \log \frac{q_k(k_{1:N})}{p(x_{1:N}, k_{1:N}, \theta^*)}
\]

The E step is given by

\[
\frac{\partial \tilde{F}}{\partial q_k(k_{1:N})} = \log \frac{q_k(k_{1:N})}{p(x_{1:N}, k_{1:N}, \theta^*)} + \lambda
\]

\[
q_k(k_{1:N}) \propto p(x_{1:N}, k_{1:N}, \theta^*) \propto \prod_{n=1}^{N} (p(x_n|k_n, \theta)p(k_n|\theta))
\]

The variational approximation factorizes by construction.

This is equivalent that for each data point \( n \) we compute the responsability \( r_n(k) \propto p(x_n|k, \theta)p(k|\theta) \) as a distribution over \( k \).
Since \( q_k = \prod_n r_n(k_n) \) is factorized we get \((p(\theta) = 1)\):

\[
\tilde{F}(q_k, \theta^*) = \sum_{k_1:N} q_k(k_1:N) \log \frac{q_k(k_1:N)}{p(x_1:N, k_1:N, \theta)} = \sum_n \sum_{k_n} r_n(k_n) \log \frac{r_n(k_n)}{p(x_n|k_n, \theta)p(k_n|\theta)}
\]

Assume \( p(x|k, \theta) = \mathcal{N}(x|\mu_k) \),

\[
\frac{\partial \tilde{F}}{\partial \mu_k} = - \sum_n \sum_{k_n} r_n(k_n)(x_n - \mu_{kn})\delta_{k,k_n} = - \sum_n r_n(k)(x_n - \mu_k) = 0
\]

yields

\[
\mu_k = \frac{\sum_n r_n(k)x_n}{\sum_n r_n(k)}
\]
In each time interval $\Delta t$ the neuron can either emit one spike or no spike ($y = 1, 0$). The probability of $y = 1$ for neuron $i$ is

$$p(y_i = 1, t+1|y(t)) = \sigma \left( \sum_{j \neq i} I_{ij} y_j(t) + \Theta_i \right), \quad \sigma(x) = \frac{1}{2}(1 + \tanh(x)).$$

$t$ labels the discretized time in units of $\Delta t$. The maximal firing frequency is $1/\Delta t$. 
Replace the binary variables $y_i = 0, 1$ by the variables $s_i = \pm 1$ using the relation $y_i = \frac{1}{2}(s_i + 1)$.

$$p(s'_i, t + 1|s, t) = \sigma(s'_i h_i(s(t))) \quad h_i(s) = \sum_{j \neq i} w_{ij} s_j + \theta_i$$

$$w_{ij} = \frac{1}{2}I_{ij} \quad \theta_i = \Theta_i + \frac{1}{2} \sum_{j \neq i} I_{ij}$$
Network dynamics

Neuron dynamics:

\[ p(s_i' | s) = \frac{1}{2} (1 + \tanh(h_i s_i')) \]

\[ h_i = \sum_j w_{ij} s_j + \theta_i \]

Neurons are updated sequentially.

Transitions are allowed from state \( s \) to any of its nearest neighbors \( s' = F_i s \)

\[ T(F_i s | s) = \frac{1}{n} p(s_i' | s) \quad T(s | s) = 1 - \sum_i^n T(F_i s | s) \]
Boltzmann-Gibbs distributions

The stationary distribution satisfies:

$$\sum_{s'} T(s|s') p(s') = p(s) = \sum_{s'} T(s'|s) p(s)$$

Detailed balance:

$$T(s|s') p(s') = T(s'|s) p(s) \text{ for all } s, s'.$$

If detailed balance holds, it implies that $p(s)$ is a stationary distribution of $T$. However, the reverse is not true.
For sequential dynamics $s' = F_i s$:

$$T(F_i s | s) = \frac{1}{n} p(s'_i | s) \Rightarrow \frac{T(s | F_i s)}{T(F_i s | s)} = \exp(2h_i s_i).$$

Consider the Boltzmann-Gibbs distribution

$$p(s) = \frac{1}{Z} \exp(-E(s))$$

$$-E(s) = \frac{1}{2} \sum_{ij} w_{ij} s_i s_j + \sum_i \theta_i s_i.$$ 

$$Z = \sum_s \exp(-E(s))$$

Depends only on symmetric part of $w_{ij}$.

$$\frac{p(s)}{p(F_i s)} = \exp(2(\sum_j w_{ij}^s s_j + \theta_i) s_i).$$

Thus, $p(s)$ satisfies detailed balance when $w_{ij}$ symmetric.
Intractability

From the Boltzmann distribution, one can compute simple statistics such as

\[ \langle s_i \rangle = \sum_s s_i p(s) \quad \text{or} \quad \langle s_i s_j \rangle = \sum_s s_i s_j p(s) \]

These sums involve \(2^n\) terms and can in general not be simplified unless eg. \(p(s) = \prod_j p_j(s_j)\):

\[ \langle s_i \rangle = \sum_s s_i \prod_j p_j(s_j) = \sum s_i p_i(s_i) \prod_{j \neq i} \sum s_j p_j(s_j) \]

Complexity is essentially in \(Z\):

\[ \langle s_i \rangle = \frac{1}{Z} \frac{\partial Z}{\partial \theta_i} \]
\[ \langle s_i s_j \rangle = \frac{1}{Z} \frac{\partial^2 Z}{\partial \theta_i \partial \theta_j} \]
Mean field theory

As for the Ising model, we choose for $q(s)$ a factorized distribution:

$$q(s) = \prod_i \frac{1}{2}(1 + m_i s_i), \quad m_i = \langle s_i \rangle_q$$

Miminizing $KL(q|p) = \sum_s q(s) \log \frac{q(s)}{p(s)}$ wrt $q$ yields the MF equations:

$$m_i = \tanh(\sum_{j=1}^{n} w_{ij} m_j + \theta_i)$$
Linear response correction

We can also compute the correlations in the mean field approximation. The crucial observation is that both the mean firing rates and the correlations can be computed as derivatives of the partition function:

\[
\langle s_i \rangle = \frac{\partial \log Z}{\partial \theta_i}
\]

\[
\chi_{ij} = \frac{\partial^2 \log Z}{\partial \theta_i \partial \theta_j}
\]

with the correlations \( \chi_{ij} = \langle s_i s_j \rangle - \langle s_i \rangle \langle s_j \rangle \). Thus,

\[
\chi_{ij} = \frac{\partial \langle s_i \rangle}{\partial \theta_j} \approx \frac{\partial m_i}{\partial \theta_j}
\]

Because the mean field equations give us an implicit relation between \( m_i \) and \( \theta_j \),
we can derive

\[
\delta m_i = \frac{1}{1 - m_i^2} \left( \sum_j w_{ij} \delta m_j + \delta \theta_i \right)
\]

\[
\frac{\partial \theta_i}{\partial m_j} = \frac{\delta_{ij}}{1 - m_i^2} - w_{ij} \equiv A_{ij} = \chi_{ij}^{-1}
\]

Thus,

\[
\langle s_i \rangle \approx m_i \quad \langle s_i s_j \rangle \approx m_i m_j + A_{ij}^{-1}
\]
Boltzmann Machines

The basic idea is to treat Boltzmann-Gibbs distribution as a statistical model \( p(s|w, \theta) \), and use standard statistical tools to estimate its parameters.

Simplest case: no hidden units.

Learning: given a set of \( P \) training patterns \( s^\mu = (s^\mu_1, \ldots, s^\mu_n) \) with \( \mu = 1, \ldots, P \), find \( w, \theta \) such that the \( p(s|w, \theta) \) 'best' describes these data.

\[
L(w, \theta) = \frac{1}{P} \sum_\mu \log p(s^\mu_1, \ldots, s^\mu_n|w, \theta)
\]

and maximize this function wrt to \( w \) and \( \theta \).
We absorbe the thresholds:

\[
p(s|w) = \frac{1}{Z} \exp \left(\frac{1}{2} \sum_{i=0}^{n} \sum_{j=0}^{n} w_{ij} s_i s_j \right)
\]

\[
= \frac{1}{Z} \exp \left(\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} s_i s_j + \sum_{i=1}^{n} \theta_i s_i \right)
\]

with \( w_{i0} = \theta_i \) and \( s_0 = 1 \).

\[
\frac{\partial \log p(s|w)}{\partial w_{ij}} = -\frac{\log Z}{\partial w_{ij}} + s_i s_j = -\langle s_i s_j \rangle + s_i s_j
\]

\[
\frac{\partial L}{\partial w_{ij}} = \frac{1}{P} \sum_{\mu} \frac{\partial \log p(s^\mu|w)}{\partial w_{ij}} = -\langle s_i s_j \rangle + \frac{1}{P} \sum_{\mu} s_i^\mu s_j^\mu = -\langle s_i s_j \rangle + \langle s_i s_j \rangle_c
\]
This maximization can be easily performed by computing the gradients of $L$ wrt $w_{ij}$ and $\theta_i$:

$$\frac{\partial L}{\partial \theta_i} = (\langle s_i \rangle_c - \langle s_i \rangle),$$

$$\frac{\partial L}{\partial w_{ij}} = (\langle s_is_j \rangle_c - \langle s_is_j \rangle) \quad i \neq j.$$

Free expectations:

$$\langle s_i \rangle = \sum_s s_ip(s), \quad \langle s_is_j \rangle = \sum_s s_is_jp(s)$$

Clamped expectations:

$$\langle s_i \rangle_c = \frac{1}{P} \sum_\mu s_i^\mu, \quad \langle s_is_j \rangle_c = \frac{1}{P} \sum_\mu s_i^\mu s_j^\mu$$
The simplest learning procedure is gradient ascent: start at $t = 0$ with a random initial value of all weights and thresholds and to iteratively change these values in the direction of their gradients:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\partial L}{\partial w_{ij}}$$

$$\theta_i(t+1) = \theta_i(t) + \eta \frac{\partial L}{\partial \theta_i}$$

which increases $L$ at each step (for sufficiently small $\eta$).

Learning terminates when the gradients are zero, i.e. at a local maximum. The first and second order statistics of the Boltzmann distribution $p$ and of the data are equal.
Hidden units

Denote visible units by $x$, hidden units by $h$ and total state $y = (x, h)$. The likelihood is given as:

$$L(w) = \frac{1}{P} \sum_{\mu} \log p(x^\mu | w) = \sum_{\mu} \log \sum_h p(x^\mu, h | w)$$

$$\frac{\partial L(w)}{\partial w_{ij}} = \frac{1}{P} \sum_{\mu} \frac{1}{p(x^\mu | w)} \sum_h \frac{\partial p(x^\mu, h | w)}{\partial w_{ij}} = \frac{1}{P} \sum_{\mu, h} \frac{p(x^\mu, h | w)}{p(x^\mu | w)} \frac{\partial \log p(x^\mu, h | w)}{\partial w_{ij}}$$

$$= \frac{1}{P} \sum_{\mu, h} p(h | x^\mu, w) \frac{\partial \log p(x^\mu, h | w)}{\partial w_{ij}} = \frac{1}{P} \sum_{\mu, h} p(h | x^\mu, w) \left( -\langle y_i y_j \rangle + y_i^\mu y_j^\mu \right)$$

$$= -\langle y_i y_j \rangle + \langle y_i y_j \rangle_c$$

with $y^\mu = (x^\mu, h)$. 
Hidden units

When \( i, j \in x \):

\[
\langle y_i y_j \rangle = \langle x_i x_j \rangle \quad \langle x_i x_j \rangle_c = \frac{1}{P} \sum_\mu x_i^\mu x_j^\mu
\]

When \( i, j \in h \):

\[
\langle y_i y_j \rangle = \langle h_i h_j \rangle \quad \langle y_i y_j \rangle_c = \frac{1}{P} \sum_\mu \langle h_i h_j \rangle_\mu \quad \langle h_i h_j \rangle_\mu = \sum_h p(h|x^\mu, w)h_i h_j
\]

When \( i \in x, j \in h \):

\[
\langle y_i y_j \rangle = \langle x_i h_j \rangle \quad \langle y_i y_j \rangle_c = \frac{1}{P} \sum_\mu x_i^\mu \langle h_j \rangle_\mu \quad \langle h_j \rangle_\mu = \sum_h p(h|x^\mu, w)h_j
\]

Bert Kappen ML 147
Mean field learning

One can use the mean field and linear response approach to approximate the learning rule.

In the absence of hidden units we can directly solve the fixed point equations. Define

\[ C_{ij} = \langle s_i s_j \rangle_c - \langle s_i \rangle_c \langle s_j \rangle_c \]

From \( \frac{\partial L}{\partial \theta_i} = 0 \) we compute \( m \):

\[ \langle s_i \rangle = \langle s_i \rangle_c \quad m_i \approx \langle s_i \rangle_c \]

From \( \frac{\partial L}{\partial w_{ij}} = 0 \) and LR approximation we compute \( w \):

\[ \langle s_i s_j \rangle = \langle s_i s_j \rangle_c \quad \chi_{ij} = C_{ij} \]

\[ \chi_{ij}^{-1} = C_{ij}^{-1} \quad w_{ij} = \frac{\delta_{ij}}{1 - m_i^2} - (C^{-1})_{ij} \]
From MF equations we compute $\theta$:

$$\theta_i = \tanh^{-1}(m_i) - \sum_{j=1}^{n} w_{ij} m_j$$
Classification of digits
Sample of 60000 training patterns and 10000 test patterns of the $28 \times 28$ handwritten digits of the U.S. Postal Service Office of Advanced Technology. Patterns are binary and 10% pixel noise is added.
Learning procedure

One Boltzmann Machine per digit. Per digit, we use approximately 6000 patterns for training. We thus obtain 10 Boltzmann distributions, each with its own parameters $W^\alpha = (w_{ij}^\alpha, \theta_i^\alpha), \alpha = 1, \ldots, 10$.

We classify each pattern $s$ to the model $\alpha$ with the highest probability:

$$\text{class}(s) = \operatorname*{argmax}_\alpha p_\alpha(s),$$

$$p_\alpha(s) = \frac{1}{Z(W^\alpha)} \exp\left(\frac{1}{2} \sum_{ij} w_{ij}^\alpha s_i s_j + \theta_i^\alpha s_i\right)$$

The normalization $Z(W^\alpha)$ is intractable and depends on $\alpha$ and therefore affects classification. We use its mean field approximation $\log Z \approx -F$.

Test the performance on 500 of the 10000 test patterns classifies 45 incorrect.

Compare with simple template matching on the mean image yields 123 errors.
The Perceptron

Relevant in history of pattern recognition and neural networks.

- Perceptron learning rule + convergence, Rosenblatt (1962)
- Perceptron critique (Minsky and Papert, 1969) → ”Dark ages of neural networks”
- Revival in the 80’s: Backpropagation
The Perceptron

\[ y(x) = \text{sign}(w^T \phi(x)) \]

where

\[ \text{sign}(a) = \begin{cases} 
  +1, & a \geq 0 \\
  -1, & a < 0.
\end{cases} \]

and \( \phi(x) \) is a feature vector (e.g. hard wired neural network).

---

Figure 3.10. The perceptron network used a fixed set of processing elements, denoted \( \phi_j \), followed by a layer of adaptive weights \( w_j \) and a threshold activation function \( g(\cdot) \). The processing elements \( \phi_j \) typically also had threshold activation functions, and took inputs from a randomly chosen subset of the pixels of the input image.
The Perceptron

Ignore $\phi$, ie. consider inputs $\xi^\mu$ and outputs $\zeta^\mu = \pm 1$
Define $w^T \xi = \sum_{j=1}^{n} w_j \xi_j + w_0$. Then, the learning condition becomes

$$\text{sign}(w^T \xi^\mu) = \zeta^\mu, \quad \mu = 1, \ldots, P$$

We have

$$\text{sign}(w^T \xi^\mu \zeta^\mu) = 1 \quad \text{or} \quad w^T x^\mu > 0$$

with $x_j^\mu = \xi_j^\mu \zeta^\mu$. 
Linear separation

Classification depends on sign of $w^T \xi$. Thus, decision boundary is hyper plane:

$$0 = w^T \xi = \sum_{j=1}^{n} w_j \xi_j + w_0$$

Perceptron can solve linearly separable problems.

AND problem is linearly separable.

XOR problem and linearly dependent inputs not linearly separable.
Perceptron learning rule

Learning successful when

\[ w^T x^\mu > 0, \text{ all patterns} \]

Learning rule is 'Hebbian':

\[ w_j^{\text{new}} = w_j^{\text{old}} + \Delta w_j \]
\[ \Delta w_j = \eta \Theta(-w^T x^\mu) \xi^\mu_j \zeta^\mu = \eta \Theta(-w^T x^\mu) x^\mu \]

\( \eta \) is the learning rate.
Depending on the data, there may be many or few solutions to the learning problem (or non at all).

The quality of the solution is determined by the worst pattern. Since the solution does not depend on the size of $w$:

$$D(w) = \frac{1}{|w|} \min_{\mu} w^T x^\mu$$

Acceptable solutions have $D(w) > 0$.

The best solution is given by $D_{\text{max}} = \max_w D(w)$. 
\[ D_{\text{max}} > 0 \] iff the problem is linearly separable.
Convergence of Perceptron rule

Assume that the problem is linearly separable, so that there is a solution $w^*$ with $D(w^*) > 0$.

At each iteration, $w$ is updated only if $w \cdot x^\mu < 0$. Let $M^\mu$ denote the number of times pattern $\mu$ has been used to update $w$. Thus,

$$w = \eta \sum_\mu M^\mu x^\mu$$

Consider the quantity

$$-1 < \frac{w \cdot w^*}{||w|| ||w^*||} < 1$$

We will show that

$$\frac{w \cdot w^*}{||w|| ||w^*||} \geq O(\sqrt{M})$$

with $M = \sum_\mu M^\mu$ the total number of iterations.

Therefore, $M$ can not grow indefinitely. Thus, the perceptron learning rule converges in a finite number of steps when the problem is linearly separable.
Proof:

\[
    w \cdot w^* = \eta \sum_{\mu} M^\mu x^\mu \cdot w^* \geq \eta M \min_{\mu} x^\mu \cdot w^*
\]

\[
    = \eta MD(w^*)|w^*|
\]

\[
    \Delta |w|^2 = |w + \eta x^\mu|^2 - |w|^2 = 2\eta w \cdot x^\mu + \eta^2 |x^\mu|^2
\]

\[
    \leq \eta^2 |x^\mu|^2 = \eta^2 N
\]

\[
    |w| \leq \eta \sqrt{NM}
\]

Thus,

\[
    1 \geq \frac{w \cdot w^*}{|w||w^*|} \geq \sqrt{M} \frac{D(w^*)}{\sqrt{N}}
\]

Number of weight updates:

\[
    M \leq \frac{N}{D^2(w^*)}
\]
Capacity of the Perceptron

Consider $P$ patterns in $N$ dimensions in general position:
- no subset of size less than $N$ is linearly dependent.
- general position is necessary for linear separability

Question: What is the probability that a problem of $P$ samples in $N$ dimensions is linearly separable?
Define $C(P, N)$ the number of linearly separable colorings on $P$ points in $N$ dimensions, with separability plane through the origin. Then (Cover 1966):

$$C(P, N) = 2 \sum_{i=0}^{N-1} \binom{P-1}{i}$$

When $P \leq N$ small, then $C(P, N) = 2 \sum_{i=0}^{P-1} \binom{P-1}{i} = 2(1 + 1)^{P-1} = 2^P$

When $P = 2N$, then 50 % is linearly separable: $C(P, N) = 2 \sum_{i=0}^{N-1} \binom{2N-1}{i} = \sum_{i=0}^{2N-1} \binom{2N-1}{i} = 2^{2N-1} = 2^{P-1}$
Proof by induction.

Add one point $X$. The set $C(P, N)$ consists of
- colorings with separator through $X$ (A)
- rest (B)

Thus,

\[ C(P + 1, N) = 2A + B = C(P, N) + A \]
\[ = C(P, N) + C(P, N - 1) \]

Yields

\[ C(P, N) = 2 \sum_{i=0}^{N-1} \binom{P-1}{i} \]
Feed-forward Network functions

We extend the previous regression model with fixed basis functions

\[
    y(x, w) = f \left( \sum_{j=1}^{M} w_j \phi_j(x) \right)
\]

to a model where \( \phi_j \) is adaptive:

\[
    \phi_j(x) = h \left( \sum_{i=0}^{D} w_{ji} x_i \right)
\]
Feed-forward Network functions

In the case of \( K \) outputs

\[
y_k(x, w) = h_2 \left( \sum_{j=1}^{M} w_{kj}^{(2)} h_1 \left( \sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) \right)
\]

\( h_2(x) \) is \( \sigma(x) \) or \( x \) depending on the problem. \( h_1(x) \) is \( \sigma(x) \) or \( \tanh(x) \).

Left) Two layer architecture. Right) general feed-forward network with skip-layer connections.

If \( h_1, h_2 \) linear, the model is linear. If \( M < D, K \) it computes principle components (Bishop section 12.4.2).
Feed-forward Network functions

Two layer NN with 3 'tanh' hidden units and linear output can approximate many functions. \( x \in [-1, 1] \), 50 equally spaced points. From left to right: \( f(x) = x^2, \sin(x), |x|, \Theta(x) \). Dashed lines are outputs of the 3 hidden units.

Two layer NN with two inputs and 2 'tanh' hidden units and sigmoid output for classification. Dashed lines are hidden unit activities.

Feed-forward neural networks have good approximation properties.
Weight space symmetries

For any solutions of the weights, there are many equivalent solutions due to symmetry:
- for any hidden unit \( j \) with \( \tanh \) activation function, change \( w_{ji} \rightarrow -w_{ji} \) and \( w_{kj} \rightarrow -w_{kj} \): \( 2^M \) solutions
- rename the hidden unit labels: \( M! \) solutions

Thus a total of \( M!2^M \) equivalent solutions, not only for \( \tanh \) activation functions.
Network training

Regression: $t_n$ continue valued, $h_2(x) = x$ and one usually minimizes the squared error (one output)

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2$$

$$= - \log \prod_{n=1}^{N} N(t_n | y(x_n, w), \beta^{-1}) + \ldots$$

Classification: $t_n = 0, 1$, $h_2(x) = \sigma(x)$, $y(x_n, w)$ is probability to belong to class 1.

$$E(w) = - \sum_{n=1}^{N} \{ t_n \log y(x_n, w) + (1 - t_n) \log(1 - y(x_n, w)) \}$$

$$= - \log \prod_{n=1}^{N} y(x_n, w)^{t_n}(1 - y(x_n, w))^{1-t_n}$$
5.2 Network training

More than two classes: consider network with $K$ outputs. $t_{nk} = 1$ if $x_n$ belongs to class $k$ and zero otherwise. $y_k(x_n, w)$ is the network output

\[
E(w) = - \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log p_k(x_n, w)
\]

\[
p_k(x, w) = \frac{\exp(y_k(x, w))}{\sum_{k'=1}^{K} \exp(y_{k'}(x, w))}
\]
Parameter optimization

\[ E(\mathbf{w}) \]

\[ \nabla E(\mathbf{w}) = 0 \]

\( E \) is minimal when \( \nabla E(\mathbf{w}) = 0 \), but not vice versa!

As a consequence, gradient based methods find a local minimum, not necessary the global minimum.
Gradient descent optimization

The simplest procedure to optimize $E$ is to start with a random $w$ and iterate

$$w^{\tau + 1} = w^\tau - \eta \nabla E(w^\tau)$$

This is called batch learning, where all training data are included in the computation of $\nabla E$.

Does this algorithm converge? Yes, if $\epsilon$ is "sufficiently small" and $E$ bounded from below.

Proof: Denote $\Delta w = -\eta \nabla E$.

$$E(w + \Delta w) \approx E(w) + (\Delta w)^T \nabla E = E(w) - \eta \sum_i \left( \frac{\partial E}{\partial w_i} \right)^2 \leq E(w)$$

In each gradient descent step the value of $E$ is lowered. Since $E$ bounded from below, the procedure must converge asymptotically.
Convergence of gradient descent in a quadratic well

\[
E(w) = \frac{1}{2} \sum_i \lambda_i w_i^2
\]

\[
\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = -\eta \lambda_i w_i
\]

\[
w_i^{\text{new}} = w_i^{\text{old}} + \Delta w_i = (1 - \eta \lambda_i) w_i
\]

Convergence when \(|1 - \eta \lambda_i| < 1\). Oscillations when \(1 - \eta \lambda_i < 0\).

Optimal learning parameter depends on curvature of each dimension.
Learning with momentum

One solution is adding momentum term:

\[ \Delta w_{t+1} = -\eta \nabla E(w_t) + \alpha \Delta w_t \]

\[ = -\eta \nabla E(w_t) + \alpha (-\eta \nabla E(w_{t-1}) + \alpha (-\eta \nabla E(w_{t-2}) + \ldots)) \]

\[ = -\eta \sum_{k=0}^{t} \alpha^k \nabla E(w_{t-k}) \]

Consider two extremes:

**No oscillations** all derivative are equal:

\[ \Delta w_{t+1} \approx -\eta \nabla E \sum_{k=0}^{t} \alpha^k = -\frac{\eta}{1 - \alpha \frac{\partial E}{\partial w}} \]

results in acceleration
Oscillations all derivatives are equal but have opposite sign:

$$\Delta w(t + 1) \approx -\eta \nabla E \sum_{k=0}^{t} (-\alpha)^k = -\frac{\eta}{1 + \alpha \partial w} \partial E$$

results in decceleration
Newton's method

One can also use Hessian information for optimization. As an example, consider a quadratic approximation to $E$ around $w_0$:

$$E(w) = E(w_0) + b^T(w - w_0) + \frac{1}{2}(w - w_0)H(w - w_0)$$

$$b_i = \frac{\partial E(w_0)}{\partial w_i} \quad H_{ij} = \frac{\partial^2 E(w_0)}{\partial w_i \partial w_j}$$

$$\nabla E(w) = b + H(w - w_0)$$

We can solve $\nabla E(w) = 0$ and obtain

$$w = w_0 - H^{-1}\nabla E(w_0)$$

This is called Newton's method.

Quadratic approximation is exact when $E$ is quadratic, so convergence in one step.
Line search

Another solution is line optimisation:

\[ w_1 = w_0 + \lambda d_0, \quad d_0 = \nabla E(w_0) \]

\( \lambda \) is found by a one dimensional optimisation

\[ 0 = \frac{\partial}{\partial \lambda} E(w_0 + \lambda d_0) = d_0 \cdot \nabla E(w_1) = d_0 \cdot d_1 \]

Therefore, subsequent search directions are orthogonal.
Conjugate gradient descent

We choose as new direction a combination of the gradient and the old direction

\[ d'_1 = \nabla E(w_1) + \beta d_0 \]

Line optimisation \( w_2 = w_1 + \lambda d'_1 \) yields \( \lambda \) such that \( d'_1 \cdot \nabla E(w_2) = 0 \).

The direction \( d'_1 \) is found by demanding that \( \nabla E(w_2) \approx 0 \) also in the 'old' direction \( d_0 \):

\[ 0 = d_0 \cdot \nabla E(w_2) \approx d_0 \cdot (\nabla E(w_1) + \lambda H(w_1)d'_1) \]

or

\[ d_0 H(w_1)d'_1 = 0 \]

\( d_0, d'_1 \) are said to be conjugate.
Polak-Ribiere rule

The conjugate directions can be computed without computing the Hessian matrix, for instance using the Polak-Ribiere rule:5

$$\beta = \frac{\left( \nabla E(w_1) - \nabla E(w_0) \right) \cdot \nabla E(w_1)}{\| \nabla E(w_0) \|^2}$$

It can be proven that this rule keeps the last $n$ directions all mutually conjugate [Press et al., 1996].

---

5 We need $0 = d_0^T H(w_1) d'_1$. We use $\nabla E(w_0) \approx \nabla E(w_1) + (w_0 - w_1)^T H(w_1) = \nabla E(w_1) - d_0^T H(w_1)$. 
Stochastic gradient descent

One can also consider on-line learning, where only one or a subset of training patterns is considered for computing $\nabla E$.

$$E(w) = \sum_{n} E_n(w) \quad w_{t+1} = w_t - \alpha_t \nabla E_n(w^T)$$

May be efficient for large data sets. This results in a stochastic dynamics in $w$ that can help to escape local minima.
Robbins Monro

Method of *stochastic approximation* originally due to Robbins and Monro 1951:
- Solve $M(x) = a$ with $M(x) = \langle N(x, \xi) \rangle$.
- Iterate $x_{t+1} = x_t + \alpha_t(a - N(x, \xi))$
- Convergence requires

\[
\sum_t \alpha_t = \infty \quad \sum_t \alpha_t^2 < \infty
\]

For instance $\alpha_t = 1/t$.

Application to stochastic gradient descent:
- $\nabla E(w) = 0$ with $\nabla E(w) = \sum_n \nabla E_n(w)$
- Iterate $w_{t+1} = w_t - \eta_t \nabla E_n(w)$

Extensions of SGD and comparisons see [Sohl-Dickstein et al., 2013].
Error backpropagation

Error is sum of error per pattern

\[ E(w) = \sum_n E^n(w) \quad E^n(w) = \frac{1}{2} \| y(x_n, w) - t_n \|^2 \]

\[ y_k(x, w) = h_2 \left( w_{k0} + \sum_{j=1}^M w_{kj} h_1 \left( w_{j0} + \sum_{i=1}^D w_{ji} x_i \right) \right) \]

\[ = h_2(a_k) \]

\[ a_k = w_{k0} + \sum_{j=1}^M w_{kj} h_1(a_j) = \sum_{j=0}^M w_{kj} h_1(a_j) \quad h_1(a_0) = 1 \]

\[ a_j = w_{j0} + \sum_{i=1}^D w_{ji} x_i = \sum_{i=0}^D w_{ji} x_i \quad x_0 = 1 \]

\[ i \text{ labels inputs, } j \text{ labels hiddens, } k \text{ labels outputs.} \]
Error backpropagation

We do each pattern separately, so we consider $E^n$

$$y_k(x^n, w) = h_2(a^n_k) = h_2\left(\sum_{j=0}^{M} w_{kj} h_1(a^n_j)\right) = h_2\left(\sum_{j=0}^{M} w_{kj} h_1\left(\sum_{i=0}^{D} w_{ji} x^n_i\right)\right)$$

$$\frac{\partial E^n}{\partial w_{kj}} = (y^n_k - t^n_k) \frac{\partial y^n_k}{\partial w_{kj}} = (y^n_k - t^n_k) h'_2(a^n_k) \frac{\partial a^n_k}{\partial w_{kj}} = (y^n_k - t^n_k) h'_2(a^n_k) h_1(a^n_j)$$

$$= \delta^n_k h_1(a^n_j) \delta^n_k = (y^n_k - t^n_k) h'_2(a^n_k)$$

$$\frac{\partial E^n}{\partial w_{ji}} = \sum_{k=1}^{K} (y^n_k - t^n_k) \frac{\partial y^n_k}{\partial w_{ji}} = \sum_{k=1}^{K} (y^n_k - t^n_k) h'_2(a^n_k) \frac{\partial a^n_k}{\partial w_{ji}}$$

$$= \sum_{k=1}^{K} \delta^n_k w_{kj} h'_1(a^n_j) \frac{\partial a^n_j}{\partial w_{ji}} = \sum_{k=1}^{K} \delta^n_k w_{kj} h'_1(a^n_j) x^n_i = \delta^n_j x^n_i$$

$$\delta^n_j = h'_1(a^n_j) \sum_{k=1}^{K} \delta^n_k w_{kj}$$
Error backpropagation

The back propagation extends to arbitrary layers:

1. $z^n_i = x^n_i$ forward propagation all activations $z^n_j = h_1(a^n_j)$ and $z^n_k = h_2(a^n_k)$, etc.

2. Compute the $\delta^n_k$ for the output units, and back-propagate the $\delta$ to obtain $\delta^n_j$ each hidden unit $j$

3. $\partial E^n / \partial w_{kj} = \delta^n_k z^n_j$ and $\partial E^n / \partial w_{ji} = \delta^n_j z^n_i$

4. for batch mode, $\partial E / \partial w_{ji} = \sum_n \partial E^n / \partial w_{ji}$

$E$ is a function of $O(|w|)$ variables. In general, the computation of $E$ requires $O(|w|)$ operations. The computation of $\nabla E$ would thus require $O(|w|^2)$ operations.

The backpropagation method allows to compute $\nabla E$ efficiently, in $O(|w|)$ operations.
Regularization

Complexity of neural network solution is controlled by number of hidden units

sum squared test error for different number of hidden units and different weight initializations. Error is also affected by local minima.
Part of the cause of local minima is the saturation of the sigmoid functions 
\( \tanh(\sum w_{ij}x_j) \). When \( w_{ij} \) becomes large, any change in its value hardly affects 
the output, implying \( \nabla_{ij}E = 0 \).

One can partly prevent this from happening by

- choosing \( \tanh \) instead of \( \sigma \) transfer functions and scaling of inputs and outputs 
  with mean zero and standard deviation one

- proper initialisation of \( w_{ij} \) with mean zero and standard deviation of order \( 1/\sqrt{n_1} \), 
  with \( n_1 \) the number of inputs to neuron \( i \).

- add regularizer such as \( \sum_i w_i^2 \) to cost keeps weights small

- dropouts, other transfer functions, adding noise, ....
MLPs are universal approximators

Consider $2^n$ binary patterns in $n$ dimensions and two classes:

$$x^\mu \rightarrow c^\mu = \pm 1, \quad x_i^\mu = \pm 1$$

Use $2^n$ hidden units, labeled $j = 0, \ldots, 2^n - 1$, $i$ labels input. Set

$$w_{ji} = b \quad \text{if } i\text{th digit in binary repr. of } j \text{ is 1}$$
$$w_{ji} = -b \quad \text{else}$$

<table>
<thead>
<tr>
<th>$j$</th>
<th>binary</th>
<th>$w_{j1}$</th>
<th>$w_{j2}$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\sum_i w_{0i}x_i$</th>
<th>$w_{1i}x_i$</th>
<th>$w_{2i}x_i$</th>
<th>$w_{3i}x_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>-b</td>
<td>-b</td>
<td>-1</td>
<td>-1</td>
<td>2b</td>
<td>0</td>
<td>0</td>
<td>-2b</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>-b</td>
<td>b</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>2b</td>
<td>-2b</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>b</td>
<td>-b</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>-2b</td>
<td>2b</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>b</td>
<td>b</td>
<td>1</td>
<td>1</td>
<td>-2b</td>
<td>0</td>
<td>0</td>
<td>2b</td>
</tr>
</tbody>
</table>
MLPs are universal approximators

Use threshold of \((n - 1)b\) at each hidden unit. \(z_j = \Theta[\sum_i w_{ji}x_i - (n - 1)b]\). The remaining problem has \(p = 2^n\) patterns in \(2^n\) dimensions and is linearly separable.

Define \(c = \text{sign}[\sum_{j=0}^{3} w_jz_j]\).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>z0</th>
<th>z1</th>
<th>z2</th>
<th>z3</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>\text{sign}[w_0]</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>\text{sign}[w_1]</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>\text{sign}[w_2]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>\text{sign}[w_3]</td>
</tr>
</tbody>
</table>

The combination of linear summation and non-linear functions can create many different functions.

- The MLP with a single hidden layer can map any continuous function \cite{Cybenko, 1989, Hornik et al., 1989}
- The MLP with multiple hidden layers may (or may not) be more efficient
Deep belief networks [Hinton and Salakhutdinov, 2006]

Finding important features in data helps to build better pattern recognition systems.

The problem can be viewed as a dimension reduction problem:
- PCA
- (non-linear) auto-encoders

Optimization is difficult with many layers. Idea:
- initialise by training RBMs for pairs of layers
- train with gradient descent/backpropagation
Restricted BM and contrastive divergence

A layer of visible neurons \( (v_i, \text{’pixels’}) \) and a layer of hidden neurons \( (h_j, \text{’features’}) \) with between-layer connections only.

\[
E(v, h) = - \sum_i b_i v_i - \sum_j b_j h_j - \sum_{i,j} w_{ij} v_i h_j \quad v_i, h_j \text{ binary}
\]

BM learning rule

\[
\Delta w_{ij} = \eta \left( \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{free}} \right)
\]

\[
\langle v_i h_j \rangle_{\text{data}} = \sum_{\mu \in \text{data}} v_i^{\mu} \langle h_j \rangle_{v, \mu} \quad \langle h_j \rangle_{v} = \sigma \left( \sum_{i'} v_{i'}^{\mu} w_{ij} + b_j \right)
\]

\[
\langle v_i h_j \rangle_{\text{free}} = \sum_{v,h} v_i h_j p(v, h)
\]

Contrastive divergence approximates \( \langle v_i h_j \rangle_{\text{free}} \) by sampling a few steps:

\[
v^{\mu} \rightarrow h^{\mu} \rightarrow (v^{\mu})' \rightarrow (h^{\mu})' \quad \langle v_i h_j \rangle_{\text{free}} \approx \sum_{\mu} (v^{\mu})'_i (h^{\mu})'_j
\]
DBN construction
Experiments: MNIST data

784 – 1000 – 500 – 250 – 30 autoencoder
60,000 training images and 10,000 test images

Top to bottom: test samples; DBN; logistic PCA with 30 components; PCA with 30 components.
The average squared errors for the last three rows are 3.00, 8.01, and 13.87.
Experiments: MNIST data

Fig. 3. (A) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (B) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (8).

DBN yields better visualisation than PCA
Experiments: documents

Each newswire story was represented as a 2000 dimensional vector of common word counts.

Training:
- 2000-500-250-125-2 autoencoder
- 402,207 training documents, 402,207 test documents

Mining for Structure

Massive increase in both computational power and the amount of data available from web, video cameras, laboratory measurements.

Images & Video
- flickr
- Google
- YouTube
- Product Recommendation
- Amazon
- eBay

Text & Language
- Relational Data/Social Network
- Facebook
- Twitter

Speech & Audio
- Gene Expression
- Climate Change
- Geological Data

- Develop statistical models that can discover underlying structure, cause, or statistical correlation from data in unsupervised or semi-supervised way.
- Multiple application domains.
Caption Generation with Visual Attention

A man riding a horse in a field.

Xu et al., ICML 2015
Challenges - I

Very different input representations

- Images – real-valued, dense
- Text – discrete, sparse

Difficult to learn cross-modal features from low-level representations.
Multimodal DBM

\( h^3 \)

\( h^2 \)

\( h^1 \)

Gaussian model

Dense, real-valued image features

\( V_{\text{image}} \)

Replicated Softmax

Word counts

\( 0 \)

\( 0 \)

\( 0 \)

\( 1 \)

\( 1 \)

\( 1 \)

(Srivastava & Salakhutdinov, NIPS 2012, ICML 2014)
Text Generated from Images

**Given**
- dog, cat, pet, kitten, puppy, ginger, tongue, kitty, dogs, furry
- sea, france, boat, mer, beach, river, bretagne, plage, brittany
- portrait, child, kid, ritratto, kids, children, boy, cute, boys, italy

**Generated**
- insect, butterfly, insects, bug, butterflies, lepidoptera
- graffiti, streetart, stencil, sticker, urbanart, graff, sanfrancisco
- canada, nature, sunrise, ontario, fog, mist, bc, morning
Text Generated from Images

Given
- portrait, women, army, soldier, mother, postcard, soldiers
- obama, barackobama, election, politics, president, hope, change, sanfrancisco, convention, rally
- water, glass, beer, bottle, drink, wine, bubbles, splash, drops, drop

Generated
Human-level control through deep reinforcement learning

http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html#videos
Linear Models in Data Mining

As datasets grow \textit{wide}—i.e. many more features than samples—the linear model has regained favor in the dataminers toolbox.

\textbf{Document classification:} bag-of-words can lead to $p = 20K$ features and $N = 5K$ document samples.

\textbf{Image deblurring, classification:} $p = 65K$ pixels are features, $N = 100$ samples.

\textbf{Genomics, microarray studies:} $p = 40K$ genes are measured for each of $N = 100$ subjects.

\textbf{Genome-wide association studies:} $p = 500K$ SNPs measured for $N = 2000$ case-control subjects.

In all of these we use linear models — e.g. linear regression, logistic regression. Since $p \gg N$, we have to regularize.
Outline

Lasso
- Coordinate descent
- Jacobi method, Gauss-Seidel method
- Convergence proofs
- Implementation issues
- Numerical results
- Cross validation to select sparsity
- Correlated inputs
- Elastic net, graph Lasso, group Lasso
LASSO

Predictors $x_i^\mu, i = 1, \ldots, n, \mu = 1, \ldots, p$ and outcomes $y^\mu$.

Standardized: $\sum_\mu x_i^\mu = 0$ and $\frac{1}{p} \sum_\mu (x_i^\mu)^2 = 1$.

Lasso (Least absolute shrinkage and selection operator):

$$\min_{\beta} \frac{1}{2} \sum_{\mu=1}^p \left( y^\mu - \sum_{i=1}^n \beta_i x_i^\mu \right)^2 \quad \sum_{i=1}^n |\beta_i| \leq t$$

Similar to ridge regression with $\sum_{i=1}^n \beta_i^2 \leq t$. 
Solve the constrained optimization with a Lagrange multiplier $\gamma$.

$$f(\beta) = \frac{1}{2p} \sum_{\mu=1}^{p} \left( y^\mu - \sum_{i=1}^{n} \beta_i x_i^\mu \right)^2 + \gamma \sum_{i=1}^{n} |\beta_i|$$

When $\gamma = 0$ solution is OLS solution

$$\frac{df}{d\beta_j} = -\frac{1}{p} \sum_{\mu=1}^{p} \left( y^\mu - \sum_{i=1}^{n} \beta_i x_i^\mu \right) x_j^\mu = -b_j + \sum_i \chi_{ij} \beta_i = 0$$

$$\hat{\beta} = \chi^{-1} b \quad \chi_{ij} = \frac{1}{p} \sum_{\mu} x_i^\mu x_j^\mu \quad b_i = \frac{1}{p} \sum_{\mu} x_i^\mu y^\mu$$

With single predictor $n = 1$, we obtain

$$\frac{df}{d\beta} = -\frac{1}{p} \sum_{\mu} (y^\mu - \beta x^\mu) x^\mu + \gamma \text{sign}(\beta) = -\hat{\beta} + \beta + \gamma \text{sign}(\beta) = 0$$
\[ \hat{\beta} = \beta + \gamma \text{sign}(\beta) \] vs. \( \beta \) for \( \gamma = 1 \).

\[
\begin{align*}
\gamma \geq |\hat{\beta}| : & \quad \beta = 0 \\
\gamma < |\hat{\beta}| : & \quad \beta = \hat{\beta} - \gamma \text{sign}(\hat{\beta})
\end{align*}
\]

Thus, \( \beta = S(\hat{\beta}, \gamma) = \text{sign}(\hat{\beta})(|\hat{\beta}| - \gamma)_+ \) (blue) shrinks the solution compared to the OLS solution \( \beta = \hat{\beta} \) (green).

Same solution \( \beta_i = S(\hat{\beta}_i, \gamma) \) is found when \( \frac{1}{p} \sum_{\mu} x^\mu_i x^\mu_j = \delta_{ij} \) but not when inputs are correlated.
In the correlated case we single out $\beta_j$ and write

$$f(\beta) = \frac{1}{2p} \sum_{\mu=1}^{p} \left( y^\mu - \sum_{i=1}^{n} \beta_i x_i^\mu \right)^2 + \gamma \sum_{i=1}^{n} |\beta_i|$$

$$= \frac{1}{2p} \sum_{\mu=1}^{p} \left( \tilde{y}_j^\mu - \beta_j x_j^\mu \right)^2 + \gamma |\beta_j| + \gamma \sum_{i\neq j}^{n} |\beta_i|$$

with $\tilde{y}_j^\mu = y^\mu - \sum_{i\neq j} \beta_i x_i^\mu$.

This is a univariate problem for $\beta_j$ and the solution is:

$$\beta_j \leftarrow S \left( \frac{1}{p} \sum_{\mu} \tilde{y}_j^\mu x_j^\mu, \gamma \right)$$

We iterate until convergence.
Diabetes data (Efron et al. 2004), $p = 442, n = 10$.

Application of coordinate descent with $\gamma = 88$. Algorithm is applied for each $\gamma$. 
Coordinate descent

The coordinate descent method is well-known for solving linear systems of equations. It is an iterative method that solves exactly for one variable, keeping all other variables fixed. In the case of Lasso, when $\gamma = 0$, we must solve $\chi \beta = b$.

Consider the general case $Ax = b$ with $A$ an $n \times n$ matrix.

We first consider a simple method called the Jacobi method. Write

$$Ax = Dx + (A - D)x = b \quad D = \text{diag}(A_{11}, \ldots, A_{nn})$$

$$x_i = \frac{1}{A_{ii}} \left( b_i - \sum_{j \neq i} A_{ij}x_j \right)$$

We can execute the iteration in parallel until convergence

$$x_i' = \frac{1}{A_{ii}} \left( b_i - \sum_{j \neq i} A_{ij}x_j \right)$$
Convergence of Jacobi method

Define $\delta x_i = x_i - \hat{x}_i$ with $A\hat{x} = b$ the fixed point solution. Then,

$$
\delta x'_i = x'_i - \hat{x}_i = \frac{1}{A_{ii}} \left( b_i - \sum_{j \neq i} A_{ij} x_j - A_{ii} \hat{x}_i \right) = -\frac{1}{A_{ii}} \sum_{j \neq i} A_{ij} \delta x_j
$$

Let $\lambda_1, \ldots, \lambda_n$ be (real or complex) eigenvalues of a matrix $A$. We define the spectral radius

$$
\rho(A) = \max_i |\lambda_i|
$$

Jacobi method converges when $\rho(D^{-1}(A - D)) < 1$.

Sufficient condition is when $A$ is diagonally dominant

$$
|A_{ii}| > \sum_{j \neq i} |A_{ij}|
$$
Gershgoren theorem

Let $A$ be a complex $n \times n$ matrix. Define $R_i = \sum_{j \neq i} |A_{ij}|$. Define $D(A_{ii}, R_i)$ a disc in complex plane with center $A_{ii}$ and radius $R_i$.

Theorem: Every eigenvalue of $A$ lies within at least one disc $D(A_{ii}, R_i)$.

Proof: Let $Ax = \lambda x$ with $\lambda$ eigenvalue and $x$ eigenvector.

$$\sum_j A_{ij} x_j = \lambda x_i$$
$$\sum_{j \neq i} A_{ij} x_j = \lambda x_i - A_{ii} x_i$$

Choose $i$ such that $|x_i|$ maximal. Then

$$|\lambda - A_{ii}| = \left| \frac{\sum_{j \neq i} A_{ij} x_j}{x_i} \right| \leq \sum_{j \neq i} |A_{ij}| = R_i$$
Apply Gershgoren to matrix $B = D^{-1}(A - D)$:

$$
B_{ii} = 0 \quad B_{ij} = \frac{A_{ij}}{A_{ii}}, \ i \neq j
$$

$$
|\lambda| \leq \sum_{j \neq i} |B_{ij}| = \frac{1}{|A_{ii}|} \sum_{j \neq i} |A_{ij}| < 1
$$

Thus, when $A$ is diagonally dominant $\rho(B) < 1$ and Jacobi method converges.
Gauss-Seidel

Consider again the linear system $Ax = b$. Write

$$A = L + U$$

with $U$ upper-triangle part of $A$ and $L$ lower-triangle part including diagonal.

Then,

$$Ax = Lx + Ux = b \quad Lx' = (b - Ux)$$

Taking advantage of the triangular form of $L$, one can solve $Lx = c$ by forward substitution:

$$L_{11}x'_1 = c_1 \quad \rightarrow x'_1$$
$$L_{21}x'_1 + L_{22}x'_2 = c_2 \quad \rightarrow x'_2$$
$$L_{31}x'_1 + L_{32}x'_2 + L_{33}x'_3 = c_3 \quad \rightarrow x'_3$$
$$\ldots$$
Gauss-Seidel

In components:

$$\sum_{j \leq i} A_{ij} x'_j = b_i - \sum_{j > i} A_{ij} x_j$$

The resulting Gauss Seidel updates are

$$x'_i = \frac{1}{A_{ii}} \left( b_i - \sum_{j > i} A_{ij} x_j - \sum_{j < i} A_{ij} x'_j \right)$$  \hspace{1cm} (1)$$

The Gauss-Seidel method converges when

• $A$ is diagonally dominant or

• $A$ is symmetric and positive definite ($x^T A x > 0$ for all $x$)
Proof of convergence of Gauss-Seidel method

Thm: GS converges for $A$ positive definite.

Proof: Define $\delta x = x - \hat{x}$ with $\hat{x}$ solution of $A\hat{x} = b$.

$$\delta x' = x' - \hat{x} = L^{-1}(b - Ux - L\hat{x}) = L^{-1}(U\hat{x} - Ux) = -L^{-1}U\delta x$$

Define $M = -L^{-1}U$. We will show that $\rho(M) < 1$ when $A$ positive definite. Let $\lambda$ be an eigenvalue of $M$ with eigenvector $x$. Then

$$Ax = (L + U)x = L(1 + L^{-1}U)x = (1 - \lambda)Lx$$
$$x^T Ax = (1 - \lambda)x^T Lx = (1 - \lambda^*)x^T L^T x$$

where the last equality follows because $x^T Ax$ is real valued, $x^T Ax = (x^T Ax)^\dagger = (1 - \lambda^*)x^T L^T x$.

$^6\dagger$ is the Hermitian conjugate operator: $(A^\dagger)_{ij} = A_{ji}^*$ and $*$ is complex conjugation.
Proof of convergence of Gauss-Seidel method

Since $A$ is positive definite,

$$A_{ii} = e_i^T A e_i > 0$$

Denote $D = \text{diag}(A)$. Thus $x^T D x > 0$. Write $A = L + L^T - D$. Then

$$0 < \frac{x^T D x}{x^T A x} = \frac{x^T (L + L^T - A) x}{x^T A x} = \frac{\left( \frac{1}{1 - \lambda} + \frac{1}{1 - \lambda^*} - 1 \right) x^T A x}{x^T A x} = \frac{1 - |\lambda|^2}{|1 - \lambda|^2}$$

Thus $|\lambda| < 1$, which completes the proof.

The same idea can be used to prove convergence of coordinate descent for LASSO $\gamma > 0$. 
Fast implementation of coordinate descent for LASSO

\[
\beta_j \leftarrow S (\tilde{\beta}_j, \gamma) = \text{sign}(\tilde{\beta}_j)(|\tilde{\beta}_j| - \gamma)_+
\]

\[
\tilde{\beta}_j = \frac{1}{p} \sum_{\mu} \tilde{y}_j^\mu x_j^\mu = b_j - \sum_{i \neq j} \chi_{ij}\beta_i
\]

Many $\beta_j$ are zero because $|\tilde{\beta}_j| < \gamma$. When one $\beta_i$ becomes non-zero the new $\tilde{\beta}_i$ is computed in $O(1)$.

We can restrict to the active set \{i | \beta_i \neq 0\}. Helps when $n \gg p$. 
Fast implementation of coordinate descent for LASSO

Warm start: run coordinate descent for series of problems from $\gamma_{\text{max}}$ to $\gamma_{\text{min}} = \epsilon \gamma_{\text{max}}$ (on log scale). Solution does not change much from one $\gamma$ to the next. Initiate with previous solution.
**Numerical results**

**Linear Regression — Dense Features**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.5</th>
<th>0.9</th>
<th>0.95</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N = 5000, \ p = 100 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>glmnet</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>lars</td>
<td>0.29</td>
<td>0.29</td>
<td>0.29</td>
<td>0.30</td>
<td>0.29</td>
<td>0.29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2.66</th>
<th>2.46</th>
<th>2.84</th>
<th>3.53</th>
<th>3.39</th>
<th>2.43</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N = 100, \ p = 50000 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>glmnet</td>
<td>58.68</td>
<td>64.00</td>
<td>64.79</td>
<td>58.20</td>
<td>66.39</td>
<td>79.79</td>
</tr>
<tr>
<td>lars</td>
<td>58.68</td>
<td>64.00</td>
<td>64.79</td>
<td>58.20</td>
<td>66.39</td>
<td>79.79</td>
</tr>
</tbody>
</table>

Timings (secs) for glmnet and lars algorithms for linear regression with lasso penalty. Total time for 100 \( \lambda \) values, averaged over 3 runs.
## Numerical results

### Logistic Regression — Real Datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>$N$</th>
<th>$p$</th>
<th>glmnet</th>
<th>l1logreg</th>
<th>BBR</th>
<th>BMR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dense</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cancer</td>
<td>14 class</td>
<td>144</td>
<td>16,063</td>
<td>2.5 mins</td>
<td>NA</td>
<td>2.1 hrs</td>
<td></td>
</tr>
<tr>
<td>Leukemia</td>
<td>2 class</td>
<td>72</td>
<td>3571</td>
<td>2.50</td>
<td>55.0</td>
<td>450</td>
<td></td>
</tr>
<tr>
<td><strong>Sparse</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internet ad</td>
<td>2 class</td>
<td>2359</td>
<td>1430</td>
<td>5.0</td>
<td>20.9</td>
<td>34.7</td>
<td></td>
</tr>
<tr>
<td>Newsgroup</td>
<td>2 class</td>
<td>11,314</td>
<td>777,811</td>
<td>2 mins</td>
<td>3.5 hrs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Timings in seconds (unless stated otherwise). For Cancer, Leukemia and Internet-Ad, times are for ten-fold cross-validation over 100 $\lambda$ values; for Newsgroup we performed a single run with 100 values of $\lambda$, with $\lambda_{\text{min}} = 0.05\lambda_{\text{max}}$. 
Choosing $\gamma$

Use cross validation on independent data to find the value of $\gamma$ that gives best prediction.

![Gaussian Family](image)

Use ‘one-standard-error’ rule to choose the largest possible $\gamma$ (sparsest solution) within the uncertainty margin.
Correlated inputs

When the correlations \( \chi_{ij} \) are strong, the ordered Lasso solution \( \beta_1 > \beta_2 \ldots < \beta_n \) may not obey the ordering of the correlations in the data \( b_1 > b_2 \ldots < b_n \). A simple example is given in (Zhao and Yu 2006).

Generate independent random variables \( x_1^\mu, x_2^\mu, e^\mu, \epsilon^\mu \) with mean zero and std 1.

Generate \( x_3^\mu = \frac{2}{3}x_1^\mu + \frac{2}{3}x_2^\mu + \frac{1}{3}e^\mu \). Then, by construction \( x_3 \) has mean zero and variance 1.

Generate output \( y^\mu = x_1^\mu \beta_1 + x_2^\mu \beta_2 + \epsilon^\mu \). Then \( \mathbb{E}x_iy = [\beta_1, \beta_2, \frac{2}{3}(\beta_1 + \beta_2)] \). Consider two cases a: \( \beta_1 = 2, \beta_2 = 3 \) and b: \( \beta_1 = -2, \beta_2 = 3 \).

![Graphs showing the relationship between \( x_i \) and \( y \) for two cases](image1.png)

When inputs are correlated, Lasso may find the wrong features.
Elastic net penalty

Zou and Hastie (2005)

Compromise between Lasso and ridge

\[ f(\beta) = \frac{1}{2p} \sum_{\mu=1}^{p} \left( y^\mu - \sum_{i=1}^{n} \beta_i x_i^\mu \right)^2 + \gamma \sum_{i=1}^{n} \left( \alpha |\beta_i| + \frac{1}{2} (1 - \alpha) \beta_i^2 \right) \]

Coordinate update is now

\[ \beta_j \leftarrow \frac{S(\tilde{\beta}_j, \alpha \gamma)}{1 + (1 - \alpha) \gamma} \]

\( \tilde{\beta}_j \) as before.
Elastic net penalty

Leukemia Data, Logistic, N=72, p=3571, first 10 steps shown
Other Applications

Undirected Graphical Models — learning dependence structure via the lasso. Model the inverse covariance $\Theta$ in the Gaussian family with $L_1$ penalties applied to elements.

$$\max_{\Theta} \log \det \Theta - \text{Tr}(S\Theta) - \lambda \|\Theta\|_1$$

Modified block-wise lasso algorithm, which we solve by coordinate descent (FHT 2007). Algorithm is very fast, and solve moderately sparse graphs with 1000 nodes in under a minute.

Example: flow cytometry - p = 11 proteins measured in N = 7466 cells (Sachs et al 2003) (next page)
$\lambda = 36$

$\lambda = 27$

$\lambda = 7$

$\lambda = 0$
**Grouped lasso** (Yuan and Lin, 2007, Meier, Van de Geer, Buehlmann, 2008) — each term $P_j(\beta_j)$ applies to *sets* of parameters:

$$\sum_{j=1}^{J} \|\beta_j\|_2.$$

*Example: each block represents the levels for a categorical predictor.*

Leads to a block-updating form of coordinate descent.
Discussion

Coordinate descent fastest method for Lasso than any other method, for instance LARS (Efron et al. 2001).

Extensions to binary classification (logistic regression) and multi-class classification.
Papers, software, talks

Friedman, J. and Hastie, T. and Höfling, H. and Tibshirani, R.
Pathwise coordinate optimization
http://www.stanford.edu/~hastie/Papers/pathwise.pdf

Friedman, J. and Hastie, T. and Tibshirani, R.
Regularized Paths for Generalized Linear Models via Coordinate Descent.
Journal of Statistical Software 33(1), 2010
http://www.jstatsoft.org/v33/i01/paper

Matlab code for GLMNET:
http://www-stat.stanford.edu/~tibs/glmnet-matlab/

Trevor Hasties KDD 2008 videolecture http://videolectures.net/kdd08_hastie_rpcd/
$L_0$ methods

George McCullochs model
The Variational Garrote
Linear regression

Given data $x_i^\mu, y^\mu, \mu = 1, \ldots, p$ find weights $w_i$ that best describe the relation

$$y^\mu = \sum_{i=1}^{n} w_i x_i^\mu + \xi^\mu$$

The ordinary least square (OLS) minimizes

$$OLS = \sum_{\mu} \left( y^\mu - \sum_{i=1}^{n} w_i x_i^\mu \right)^2$$

Solution is given by

$$w = \chi^{-1} b \quad \chi_{ij} = \frac{1}{p} \sum_{\mu} x_i^\mu x_j^\mu \quad b_i = \frac{1}{p} \sum_{\mu} x_i^\mu y^\mu$$

Problems: low accuracy due to overfitting ($p < n$) and interpretation: the OLS solution is not sparse
Ridge regression

Add a regularization term

\[
\text{Ridge} = OLS + \lambda \sum_i w_i^2 \quad \lambda > 0
\]

\[
w = (\chi + \lambda 1)^{-1} b
\]

Ridge regression
- improves the prediction accuracy
- maximal rank
- solution is not sparse
Lasso

Solve the OLS problem problem under the linear constraint $\sum_i |w_i| \leq t$. Equivalently, add a regularization term

$$Lasso = OLS + \lambda \sum_i |w_i| \quad \lambda > 0$$

There exist efficient methods to solve this quadratic programming problem.

The solution tends to be sparse and improves both the prediction accuracy and the interpretability of the solution.

Both ridge regression and Lasso are shrinkage methods that find a solution that is biased to smaller $w$. 

Bert Kappen ML 233
The Garrote vil
The variational Garrote

Introduce $s_i = 0, 1$ that select features. The regression model becomes

$$y^\mu = \sum_{i=1}^{n} w_i s_i x_i^\mu + \xi^\mu \quad s_i = 0, 1$$

To optimize the $s_i$ is equivalent to find the optimal subset of relevant features. Since the number of subsets is exponential in $n$ one has to resort to heuristic methods to find a good subset of features.

Here we propose a variational approximation.
The variational Garrote

The likelihood term is given by

\[
p(y|\vec{x}, \vec{s}, \vec{w}, \beta) = \sqrt{\frac{\beta}{2\pi}} \exp \left( -\frac{\beta}{2} \left( y - \sum_{i=1}^{n} w_i s_i x_i \right)^2 \right)
\]

\[
p(D|\vec{s}, \vec{w}, \beta) = \prod_{\mu} p(y^{\mu}|\vec{x}^{\mu}, \vec{s}, \vec{w}, \beta)
\]

\[
= \left( \frac{\beta}{2\pi} \right)^{p/2} \exp \left( -\frac{\beta p}{2} \left( \sum_{i,j=1}^{n} s_i s_j w_i w_j \chi_{ij} - 2 \sum_{i=1}^{n} w_i s_i b_i + \sigma_y^2 \right) \right)
\]

with \(b_i, \chi_{ij}\) as before and \(\sigma_y^2 = \frac{1}{p} \sum_{\mu} (y^{\mu})^2\).
The variational Garrote

For concreteness, we assume that the prior over $\vec{s}$

$$p(\vec{s}|\gamma) = \prod_{i=1}^{n} p(s_i|\gamma) \quad p(s_i|\gamma) = \frac{\exp(\gamma s_i)}{1 + \exp(\gamma)}$$

with $\gamma$ given which specifies the sparsity of the solution.

We further assume priors $p(\beta, \vec{w})$. 
The variational Garrote

The posterior becomes

\[ p(\vec{s}, \vec{w}, \beta | D, \gamma) = \frac{p(\vec{w}, \beta) p(\vec{s} | \gamma) p(D | \vec{s}, \vec{w}, \beta)}{p(D | \gamma)} \]

Posterior is intractable:
- MCMC
- Variational Bayes
- Variational MAP
- BP, CVM, ...

Here we compute a variational MAP estimate. We approximate the marginal posterior

\[ p(\vec{w}, \beta | D, \gamma) = \sum_{\vec{s}} p(\vec{s}, \vec{w}, \beta | D, \gamma) \]

and computing the MAP solution with respect to \( \vec{w}, \beta \).
Breiman’s Garrote method

The proposed model is similar to Breiman’s Garrote method:

\[ y^\mu = \sum_{i=1}^{n} w_i s_i x_i^\mu + \xi^\mu \quad s_i = 0, 1 \]

which assumes \( 0 \leq s_i \leq 1 \) instead of binary.

It computes \( w_i \) using OLS and then finds \( s_i \) by minimizing

\[
\sum_{\mu} \left( y^\mu - \sum_{i=1}^{n} x_i^\mu w_i s_i \right)^2 \quad \text{subject to} \quad s_i \geq 0 \quad \sum_{i} s_i \leq t
\]

We refer to our method as the Variational Garrote (VG).
The variational approximation

We compute the variational approximation using Jensens inequality:

\[
\log \sum_{\vec{s}} p(\vec{s}|\gamma)p(D|\vec{s}, \vec{w}, \beta) \geq - \sum_{\vec{s}} q(\vec{s}) \log \frac{q(\vec{s})}{p(\vec{s}|\gamma)p(D|\vec{s}, \vec{w}, \beta)} = -F(q, \vec{w}, \beta)
\]

The optimal \(q(\vec{s})\) is found by minimizing \(F(q, \vec{w}, \beta)\) with respect to \(q(\vec{s})\).

We consider the simplest case

\[
q(\vec{s}) = \prod_{i=1}^{n} q_i(s_i) \quad q_i(s_i) = m_i s_i + (1 - m_i)(1 - s_i)
\]

So \(q(\vec{s})\) is parametrized by \(\vec{m}\).
The variational approximation

The expectation values with respect to \( q \) can now be easily evaluated and the result is

\[
F = -\frac{p}{2} \log \frac{\beta}{2\pi} + \frac{\beta p}{2} \left( \sum_{i,j} v_i v_j \chi_{ij} + \sum_i \frac{1 - m_i}{m_i} v_i^2 \chi_{ii} - 2 \sum_{i=1}^n v_i b_i + \sigma_y^2 \right) \\
\quad - \gamma \sum_{i=1}^n m_i + n \log(1 + \exp(\gamma)) \\
\quad + \sum_{i=1}^n \left( m_i \log m_i + (1 - m_i) \log(1 - m_i) \right)
\]

where we have defined \( v_i = m_i w_i \).
The variational approximation

The approximate posterior marginal posterior is then

\[
p(\vec{w}, \beta | D, \gamma) \propto p(\vec{w}, \beta) \sum_{\vec{s}} p(\vec{s} | \gamma) p(D | \vec{s}, \vec{w}, \beta)
\]

\[
\approx p(\vec{w}, \beta) \exp(-F(\vec{m}, \vec{w}, \beta, \gamma)) = \exp(-G(\vec{m}, \vec{w}, \beta, \gamma))
\]

\[
G(\vec{m}, \vec{w}, \beta, \gamma) = F(\vec{m}, \vec{w}, \beta, \gamma) - \log p(\vec{w}, \beta)
\]

We can compute the variational approximation $\vec{m}$ for given $\vec{w}, \beta, \gamma$ by minimizing $F$ with respect to $\vec{m}$. In addition, $p(\vec{w}, \beta | D, \gamma)$ needs to be maximized with respect to $\vec{w}, \beta$. 

---

Bert Kappen

ML 242
The variational approximation

Taking the derivative of $G$ with respect $\vec{m}, \vec{v}, \beta$ and setting the derivatives equal to zero gives the following set of fixed point equations:

$$m_i = \sigma \left( \gamma + \frac{\beta p v_i^2 \chi_{ii}}{2 m_i^2} \right)$$

$$\vec{v} = (\chi')^{-1} \vec{b}$$

$$\chi'_{ij} = \chi_{ij} + \frac{1 - m_i}{m_i} \chi_{ii} \delta_{ij}$$

$$\frac{1}{\beta} = - \sum_{i=1}^{n} v_i b_i + \sigma_y^2$$

with $\sigma(x) = (1 + \exp(-x))^{-1}$
The variational approximation is not simply

\[ w_i s_i \rightarrow w_i m_i \]

If this were the case, the substitution \( v_i = w_i m_i \) would remove \( m_i \) from the equations and the OLS problem would be recovered.

The reason is \( \langle s_i s_j \rangle = m_i m_j \) for \( i \neq j \), but \( \langle s_i^2 \rangle = \langle s_i \rangle = m_i \).

\( \chi' \) differs from \( \chi \) by adding a positive diagonal to it, making \( \chi' \) automatically of maximal rank when \( m_i < 1 \). Roughly speaking if \( \chi \) has rank \( p < n \), \( \chi' \) can be still of rank \( n \) when no more than \( p \) of the \( m_i = 1 \), the remaining \( n - p \) of the \( m_i < 1 \) making up for the rank deficiency.
Independent inputs

When inputs are uncorrelated: $\chi_{ij} = \delta_{ij}$,

$$
\begin{align*}
  w_i &= b_i = \langle x_i y \rangle \\
  m_i &= \sigma \left( \gamma + \frac{\beta p}{2} b_i^2 \right) \\
  1/\beta &= \sigma_y^2 - \sum_i b_i^2 m_i
\end{align*}
$$

\(\sum_i b_i^2 m_i\) is the explained variance. The Garrote solution with \(0 \leq m_i \leq 1\) has reduced explained variance with (hopefully) a better prediction accuracy and interpretability.
Univariate case

In the 1-dimensional case these equations become

\[ m = \sigma \left( \gamma + \frac{p \rho}{2(1 - \rho m)} \right) = f(m) \]

\[ \frac{1}{\beta} = \sigma^2_y (1 - m\rho) \]

with \( \rho = \frac{b^2}{\sigma^2_y} \) the squared correlation coefficient.
Univariate case

$f(m)$ is an increasing function of $m$ and crosses the line $m$ either 1 or three times, depending on the values of $p, \gamma, \rho$.

$f(m)$ vs $m$. Left: $p = 100, \gamma = -10$, different lines correspond to different values of $0 < \rho < 1$. Right: $p = 100, \gamma = -30$. solutions for $m$.

The solutions close to $m \approx 0, 1$ correspond to local minima of $F$. The intermediate solution corresponds to a local maximum of $F$. 
Univariate case

One can compute the critical $p$ for which multiple solutions occur.

$$p^* = \frac{4}{\rho^2} \sqrt{1 - \rho} \left( \sqrt{1 - \rho} + \sqrt{2} \right)$$

$p^*$ is a decreasing function of $\rho$. For $p > p^*$, we find two solutions for $m$. For $p < p^*$, we find one solution for $m$.

Left: Phase plot $\rho, \gamma$ for $p = 100$ Dotted line is solution for $\gamma$ when $m = 1/2$. Right: $m$ versus $\rho$ for $\gamma = -10, p = 100$ (top) and for $\gamma = -40, p = 100$ (bottom).
Suppose that data are generated from the model

\[ y = wx + \xi \quad \langle \xi^2 \rangle = \langle x^2 \rangle = 1 \]

Variational Garrote (VG) with \( \gamma = -10 \) and \( p = 100 \). Ridge regression with \( \lambda = 0.5 \). Garrote with \( \gamma = 1/4 \). Lasso with \( \gamma = 0.5 \).
Numerical examples

Inputs are generated from a mean zero multi-variate Gaussian distribution with specified covariance structure.

We generate outputs \( y^\mu = \sum_i \hat{w}_i x_i^\mu + \xi^\mu \) with \( \xi^\mu \in \mathcal{N}(0, \hat{\sigma}) \).

For each example, we generate a training set, a validation set and a test set \((p/p_v/p_t)\).

For each value of the hyper parameters (\( \gamma \) in the case of VG, \( \lambda \) in the case of ridge regression and Lasso), we optimize the model parameters on the training set.

We optimize the hyper parameters on the validation set.
Example 1

\[ x_i^\mu \in \mathcal{N}(0, 1) \text{ independently.} \]
\[ \hat{w} = (1, 0, \ldots, 0), \; n = 100 \text{ and } \hat{\sigma} = 1. \]
\[ p/p_v/p_t = 50/50/400. \]
Example 1

\( x_i^\mu \in \mathcal{N}(0, 1) \) independently.
\( \hat{w} = (1, 0, \ldots, 0), \ n = 100 \) and \( \hat{\sigma} = 1. \)
\( p/p_v/p_t = 50/50/400. \)

Lasso (top row) and ridge regression (bottom row)
Example 1

\( x_i^μ \in \mathcal{N}(0, 1) \) independently.
\( \hat{w} = (1, 0, \ldots, 0) \), \( n = 100 \) and \( \hat{σ} = 1 \).
\( p/p_v/p_t = 50/50/400 \).

|        | Train      | Val        | Test       | # non-zero | \( ||\delta w||_1 || \) | \( ||\delta w||_2 \) |
|--------|------------|------------|------------|------------|--------------------------|--------------------------|
| Ridge  | 0.44 ± 0.29| 1.75 ± 0.30| 1.79 ± 0.18| -          | 4.40 ± 1.08              | 0.80 ± 0.05              |
| Lasso  | 0.80 ± 0.22| 1.06 ± 0.24| 1.15 ± 0.23| 3.80 ± 2.04| 0.64 ± 0.39              | 0.16 ± 0.16              |
| VG     | 0.83 ± 0.18| 0.89 ± 0.19| 1.02 ± 0.09| 1.76 ± 1.36| 0.28 ± 0.26              | 0.04 ± 0.04              |
| True   | 0.93 ± 0.14| 0.87 ± 0.20| 0.98 ± 0.04| 1          | 0                        | 0                        |

Results on 10 random instances.
Example 2

\[ \tilde{\beta}^\mu \in \mathcal{N}(0, \Sigma) \text{ with } \Sigma_{ij} = \delta|i-j|, \delta = 0.5. \]
\[ \hat{w}_i = 1, i = 1, 2, 5, 10, 50 \text{ and all other } \hat{w}_i = 0, n = 100, \hat{\sigma} = 1. \]
\[ p/p_v/p_t = 50/50/400. \]

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Val</th>
<th>Test</th>
<th># non-zero</th>
<th>|\delta w|_1</th>
<th>|\delta w|_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lasso</td>
<td>0.78 ± 0.47</td>
<td>1.40 ± 0.31</td>
<td>1.49 ± 0.23</td>
<td>11.20 ± 3.12</td>
<td>2.09 ± 0.77</td>
<td>0.55 ± 0.22</td>
</tr>
<tr>
<td>VG</td>
<td>0.81 ± 0.21</td>
<td>1.18 ± 0.20</td>
<td>1.21 ± 0.16</td>
<td>5.50 ± 0.71</td>
<td>1.00 ± 0.46</td>
<td>0.32 ± 0.37</td>
</tr>
<tr>
<td>True</td>
<td>1.01 ± 0.18</td>
<td>0.97 ± 0.19</td>
<td>0.99 ± 0.07</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Results on 10 random instances.
Example of [Zhao and Yu, 2006].

Figure 1: Lasso and VG solution for the inconsistent Example a of [Zhao and Yu, 2006]. Top left: Lasso solution versus $\lambda$ is called inconsistent because it does not contain a $\lambda$ for which the correct sparsity ($w_{1,2} \neq 0, w_3 = 0$) is obtained. Top right: the VG solution for $\vec{v}$ versus $\gamma$ contains large range of $\gamma$ for which the correct solution is obtained. Bottom left: VG solution for $\vec{m}$ (curves for $m_{1,2}$ are identical). Bottom right: VG solution for $\vec{w}$. 
Example of [Zhao and Yu, 2006].

The average results over 100 instances for Example a and Example b are shown in table 1.

<table>
<thead>
<tr>
<th></th>
<th>Example a</th>
<th></th>
<th>Example b</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>|(\delta \vec{v})</td>
<td></td>
<td>\text{max}(\text{|v}_{3})</td>
</tr>
<tr>
<td>Ridge</td>
<td>0.64 ± 0.18</td>
<td>0.48</td>
<td>0.02 ± 0.02</td>
</tr>
<tr>
<td>Lasso</td>
<td>0.19 ± 0.14</td>
<td>0.30</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>VG</td>
<td>0.05 ± 0.03</td>
<td>0.00</td>
<td>0.00 ± 0.00</td>
</tr>
</tbody>
</table>

Table 1: Accuracy of Ridge, Lasso and VG for Example 1a,b from [Zhao and Yu, 2006]. \(p = p_v = 1000\). Parameters \(\lambda\) (Ridge and Lasso) and \(\gamma\) (VG) optimized through cross validation. \(\|\delta \vec{v}\|_1\) as before, \text{max}(\text{\|v}_{3})\) is maximum over 100 trials of the absolute value of \(v_3\). Example a is inconsistent for Lasso and yields much larger errors than the VG. Example b is consistent and the quality of the Lasso and VG are similar. Ridge regression is bad for both examples.
Performance as a function of number of training samples $p$ for two levels of sparsity (10% and 25% of non-zero entries). For each value averages over 20 runs are plotted. **Top:** area under the ROC curves. **Middle:** reconstruction error, defined as $||\delta \tilde{w}||_1 = \sum_{i=1}^{n} |v_i - \hat{w}_i|$. **Bottom:** generalization error. Lowest curve shows true model.
Correlated case: Genetic dataset

(LEFT) input correlation matrix. (RIGHT) Performance as a function of number of training samples $p$ for two levels of sparsity (10% and 25% of non-zero entries). **Top:** area under the ROC curves. **Middle:** reconstruction error, defined as $||\delta \mathbf{v}||_1 = \sum_{i=1}^{n} |v_i - \hat{w}_i|$. **Bottom:** generalization error.
Scaling with dimension $n$

Scaling with $n$: performance of VG, PMF and Lasso as a function of the number of features $n$. Data are generated as in Example 2. $p = 100, p_v = 100, \beta = 2, \zeta = 0$. 
Discussion

Local minima:
- appear for few and noisy data.
- seem modest for (very) sparse problems.
- increasing $\gamma$ increases $\beta$ and works as an annealing schedule.

Extensions:
- MAP: TAP, BP, CVM
- 'Full Bayes': MCMC, VB, ...
- Use of priors (on $\gamma$) instead of cross validation

Applications:
- Finding structure of networks, both static and dynamic
- Finding genes in GWAS
- ...

arxiv.org/abs/1109.0486
Control theory
Introduction

**Optimal control theory:** Optimize sum of a path cost and end cost. Result is optimal control sequence and optimal trajectory.

**Input:** Cost function.
**Output:** Optimal trajectory and controls.
Introduction

Control problems are delayed reward problems:

- Motor control: devise a sequence of motor commands to reach a goal
- Finance: devise a sequence of buy/sell commands to maximize profit
- Learning, exploration exploitation
Types of optimal control problems

Finite horizon (fixed horizon time)

- Dynamics and environment may depend explicitly on time.
- Optimal control depends explicitly on time.
Types of optimal control problems

Finite horizon (moving horizon)

- Dynamics and environment are static.
- Optimal control is time independent.

Infinite horizon

- discounted reward, Reinforcement learning
- total reward, absorbing states
- average reward

Other issues:

- discrete vs. continuous state
- discrete vs. continuous time
- observable vs. partial observable
Overview

**Optimal control theory, discrete time**
- Introduction of delayed reward problem in discrete time;
- Dynamic programming solution

**Optimal control theory, continuous time**
- Pontryagin maximum principle;

**Stochastic optimal control theory**
- Stochastic differential equations
- Kolmogorov and Fokker-Plack equations
- Hamilton-Jacobi-Bellman equation
- LQ control, Ricatti equation;
- Portfolio selection

**Path integral/KL control theory**
- Importance sampling
- KL control theory
Material


• Dimitri Bertsekas, Dynamic programming and optimal control

• website
Optimal control theory: discrete time
Discrete time control

Consider the control of a discrete time deterministic dynamical system:

\[ x_{t+1} = x_t + f(t, x_t, u_t), \quad t = 0, 1, \ldots, T - 1 \]

\( x_t \) describes the state and \( u_t \) specifies the control or action at time \( t \).

Given \( x_{t=0} = x_0 \) and \( u_{0:T-1} = u_0, u_1, \ldots, u_{T-1} \), we can compute \( x_{1:T} \).

Define a cost for each sequence of controls:

\[
C(x_0, u_{0:T-1}) = \phi(x_T) + \sum_{t=0}^{T-1} R(t, x_t, u_t)
\]

The problem of optimal control is to find the sequence \( u_{0:T-1} \) that minimizes \( C(x_0, u_{0:T-1}) \).
Dynamic programming

Find the minimal cost path from A to J.

\[ C(J) = 0, \quad C(H) = 3, \quad C(I) = 4 \]

\[ C(F) = \min(6 + C(H), 3 + C(I)) \]
Discrete time control

The optimal control problem can be solved by dynamic programming. Introduce the **optimal cost-to-go**:

\[
J(t, x_t) = \min_{u_{t:T-1}} \left( \phi(x_T) + \sum_{s=t}^{T-1} R(s, x_s, u_s) \right)
\]

which solves the optimal control problem from an intermediate time \( t \) until the fixed end time \( T \), for all intermediate states \( x_t \).

Then,

\[
J(T, x) = \phi(x) \\
J(0, x) = \min_{u_{0:T-1}} C(x, u_{0:T-1})
\]
Discrete time control

One can recursively compute $J(t, x)$ from $J(t + 1, x)$ for all $x$ in the following way:

$$J(t, x_t) = \min_{u_t:T-1} \left( \phi(x_T) + \sum_{s=t}^{T-1} R(s, x_s, u_s) \right)$$

$$= \min_{u_t} \left( R(t, x_t, u_t) + \min_{u_{t+1:T-1}} \left[ \phi(x_T) + \sum_{s=t+1}^{T-1} R(s, x_s, u_s) \right] \right)$$

$$= \min_{u_t} (R(t, x_t, u_t) + J(t + 1, x_{t+1}))$$

$$= \min_{u_t} (R(t, x_t, u_t) + J(t + 1, x_t + f(t, x_t, u_t)))$$

This is called the Bellman Equation.

Computes $u$ as a function of $x, t$ for all intermediate $t$ and all $x$. 
Discrete time control

The algorithm to compute the optimal control $u^*_{0:T-1}$, the optimal trajectory $x^*_{1:T}$ and the optimal cost is given by

1. Initialization: $J(T, x) = \phi(x)$

2. Backwards: For $t = T - 1, \ldots, 0$ and for all $x$ compute

$$u_t^*(x) = \arg \min_u \{R(t, x, u) + J(t + 1, x + f(t, x, u))\}$$
$$J(t, x) = R(t, x, u_t^*) + J(t + 1, x + f(t, x, u_t^*))$$

3. Forwards: For $t = 0, \ldots, T - 1$ compute

$$x_{t+1}^* = x_t^* + f(t, x_t^*, u_t^*(x_t^*))$$

NB: the backward computation requires $u_t^*(x)$ for all $x$. 
Stochastic case

\[ x_{t+1} = x_t + f(t, x_t, u_t, w_t) \quad t = 0, \ldots, T - 1 \]

At time \( t \), \( w_t \) is a random value drawn from a probability distribution \( p(w) \).

For instance,

\[
\begin{align*}
    x_{t+1} &= x_t + w_t, \quad x_0 = 0 \\
    w_t &= \pm 1, \quad p(w_t = 1) = p(w_t = -1) = 1/2 \\
    x_t &= \sum_{s=0}^{t-1} w_s
\end{align*}
\]

Thus, \( x_t \) random variable and so is the cost

\[
C(x_0) = \phi(x_T) + \sum_{t=0}^{T-1} R(t, x_t, u_t, \xi_t)
\]
Stochastic case

\[
C(x_0) = \left\langle \phi(x_T) + \sum_{t=0}^{T-1} R(t, x_t, u_t, \xi_t) \right\rangle
\]

\[
= \sum_{w_0:T-1} \sum_{\xi_0:T-1} p(w_0:T-1) p(\xi_0:T-1) \left\langle \phi(x_T) + \sum_{t=0}^{T-1} R(t, x_t, u_t, \xi_t) \right\rangle
\]

with \(\xi_t, x_t, w_t\) random. Closed loop control: find functions \(u_t(x_t)\) that minimizes the remaining expected cost when in state \(x\) at time \(t\). \(\pi = \{u_0(\cdot), \ldots, u_{T-1}(\cdot)\}\) is called a policy.

\[
x_{t+1} = x_t + f(t, x_t, u_t(x_t), w_t)
\]

\[
C_\pi(x_0) = \left\langle \phi(x_T) + \sum_{t=0}^{T-1} R(t, x_t, u_t(x_t), \xi_t) \right\rangle
\]

\(\pi^* = \arg\min_\pi C_\pi(x_0)\) is optimal policy.
Stochastic Bellman Equation

\[ J(t, x_t) = \min_{u_t} \langle R(t, x_t, u_t, \xi_t) + J(t + 1, x_t + f(t, x_t, u_t, w_t)) \rangle \]
\[ J(T, x) = \phi(x) \]

\( u_t \) is optimized for each \( x_t \) separately. \( \pi = \{u_0, \ldots, u_{T-1}\} \) is optimal a policy.
Inventory problem

- $x_t = 0, 1, 2$ stock available at the beginning of period $t$.

- $u_t$ stock ordered at the beginning of period $t$. Maximum storage is 2: $u_t \leq 2 - x_t$.

- $w_t = 0, 1, 2$ demand during period $t$ with $p(w = 0, 1, 2) = (0.1, 0.7, 0.2)$; excess demand is lost.

- $u_t$ is the cost of purchasing $u_t$ units. $(x_t + u_t - w_t)^2$ is cost of stock at end of period $t$.

$$x_{t+1} = \max(0, x_t + u_t - w_t)$$

$$C(x_0, u_{0:T-1}) = \left\langle \sum_{t=0}^{T=2} u_t + (x_t + u_t - w_t)^2 \right\rangle$$

Planning horizon $T = 3$. 
Inventory problem
Apply Bellman Equation

\[ J_t(x_t) = \min_{u_t} \langle R(x_t, u_t, w_t) + J_{t+1}(f(x_t, u_t, w_t)) \rangle \]

\[ R(x, u, w) = u + (x + u - w)^2 \]

\[ f(x, u, w) = \max(0, x + u - w) \]

Start with \( J_3(x_3) = 0, \forall x_3 \).
Dynamic programming in action

Assume we are at stage $t = 2$ and the stock is $x_2$. The cost-to-go is what we order $u_2$ and how much we have left at the end of period $t = 2$.

\[
J_2(x_2) = \min_{0 \leq u_2 \leq 2 - x_2} \left( u_2 + \left( x_2 + u_2 - w_2 \right)^2 \right)
\]

\[
= \min_{0 \leq u_2 \leq 2 - x_2} \left( u_2 + 0.1 \cdot (x_2 + u_2)^2 + 0.7 \cdot (x_2 + u_2 - 1)^2 + 0.2 \cdot (x_2 + u_2 - 2)^2 \right)
\]

\[
J_2(0) = \min_{0 \leq u_2 \leq 2} \left( u_2 + 0.1 \cdot u_2^2 + 0.7 \cdot (u_2 - 1)^2 + 0.2 \cdot (u_2 - 2)^2 \right)
\]

\[
u_2 = 0 : \quad \text{rhs} = 0 + 0.7 \cdot 1 + 0.2 \cdot 4 = 1.5
\]

\[
u_2 = 1 : \quad \text{rhs} = 1 + 0.1 \cdot 1 + 0.2 \cdot 1 = 1.3
\]

\[
u_2 = 2 : \quad \text{rhs} = 2 + 0.1 \cdot 4 + 0.7 \cdot 1 = 3.1
\]

Thus, $u_2(x_2 = 0) = 1$ and $J_2(x_2 = 0) = 1.3$
Inventory problem

The computation can be repeated for \(x_2 = 1\) and \(x_2 = 2\), completing stage 2 and subsequently for stage 1 and stage 0.

<table>
<thead>
<tr>
<th>Stock</th>
<th>Stage 0 Cost-to-go</th>
<th>Stage 0 Optimal stock to purchase</th>
<th>Stage 1 Cost-to-go</th>
<th>Stage 1 Optimal stock to purchase</th>
<th>Stage 2 Cost-to-go</th>
<th>Stage 2 Optimal stock to purchase</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.67</td>
<td>1</td>
<td>2.5</td>
<td>1</td>
<td>1.3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2.67</td>
<td>0</td>
<td>1.2</td>
<td>0</td>
<td>0.3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2.608</td>
<td>0</td>
<td>1.68</td>
<td>0</td>
<td>1.1</td>
<td>0</td>
</tr>
</tbody>
</table>
Exercise: Two ovens

A certain material is passed through a sequence of two ovens. Aim is to reach pre-specified final product temperature \( x^* \) with minimal oven energy.

\[ x_{t+1} = (1 - a)x_t + au_t \quad t = 0, 1 \]

\[ C = r(x_2 - x^*)^2 + u_0^2 + u_1^2 \]

- Find the optimal control solution \( u_0, u_1 \).
- Show that adding mean zero noise to the dynamics \( (x_{t+1} = (1 - a)x_t + au_t + w_t \text{ with } \langle w_t \rangle = 0) \), does not change the optimal control solution.
Example: Two ovens

End cost-to-go is $J(2, x_2) = r(x_2 - x^*)^2$. 

$J(1, x_1) = \min_{u_1} \left(u_1^2 + J(2, x_2)\right) = \min_{u_1} \left(u_1^2 + r((1-a)x_1 + au_1 - x^*)^2\right)$

$u_1 = \mu_1(x_1) = \frac{ra(x^* - (1-a)x_1)}{1 + ra^2}$

$J(1, x_1) = \frac{r((1-a)x_1 - x^*)^2}{1 + ra^2}$

$J(0, x_0) = \min_{u_0} \left(u_0^2 + J(1, x_1)\right) = \min_{u_0} \left(u_0^2 + \frac{r((1-a)x_1 - x^*)^2}{1 + ra^2}\right)$

$\quad = \min_{u_0} \left(u_0^2 + \frac{r((1-a)((1-a)x_0 + au_0) - x^*)^2}{1 + ra^2}\right)$

$u_0 = \mu_0(x_0) = \frac{r(1-a)a(x^* - (1-a)^2x_0)}{1 + ra^2(1 + (1-a)^2)}$

$J(0, x_0) = \frac{r((1-a)^2x_0 - x^*)^2}{1 + ra^2(1 + (1-a)^2)}$
Comments

- **Linear Quadratic Control:** Solution can be obtained in closed form because problem is linear quadratic.

- **Certainty equivalence:** Optimal control solution is unaffected by noise:

\[
x_{t+1} = (1 - a)x_t + au_t + w_t \quad t = 0, 1
\]

\[
C = r(x_2 - x^*)^2 + u_0^2 + u_1^2
\]

with \( \langle w_t \rangle = 0 \). Then

\[
J(1, x_1) = \min_{u_1} \left( u_1^2 + \langle r((1 - a)x_1 + au_1 + w_1 - x^*)^2 \rangle \right)
\]

\[
= \min_{u_1} \left( u_1^2 + r((1 - a)x_1 + au_1 - x^*)^2 + r \langle w_1 \rangle^2 \right)
\]
Optimal control theory, continuous time
Continuous limit

Replace \( t + 1 \) by \( t + dt \) with \( dt \to 0 \).

\[
\begin{align*}
x_{t+dt} &= x_t + f(x_t, u_t, t)dt \\
C(x_0, u_{0\rightarrow T}) &= \phi(x_T) + \int_0^T d\tau R(\tau, x(\tau), u(\tau))
\end{align*}
\]

Assume \( J(x, t) \) is smooth.

\[
\begin{align*}
J(t, x) &= \min_u (R(t, x, u)dt + J(t + dt, x + f(x, u, t)dt)) \\
&\approx \min_u (R(t, x, u)dt + J(t, x) + \partial_t J(t, x)dt + \partial_x J(t, x)f(x, u, t)dt) \\
-\partial_t J(t, x) &= \min_u (R(t, x, u) + f(x, u, t)\partial_x J(x, t))
\end{align*}
\]

with boundary condition \( J(x, T) = \phi(x) \).
Continuous limit

\[-\partial_t J(t, x) = \min_u (R(t, x, u) + f(x, u, t)\partial_x J(x, t))\]

with boundary condition \(J(x, T) = \phi(x)\).

This is called the Hamilton-Jacobi-Bellman Equation.

Computes the anticipated potential \(J(t, x)\) from the future potential \(\phi(x)\).
Example: Mass on a spring

The spring force $F_z = -z$ towards the rest position and control force $F_u = u$.

Newton’s Law

$$F = -z + u = m\ddot{z}$$

with $m = 1$.

Control problem: Given initial position and velocity $z(0) = \dot{z}(0) = 0$ at time $t = 0$, find the control path $-1 < u(0 \rightarrow T) < 1$ such that $z(T)$ is maximal.
Example: Mass on a spring

Introduce $x_1 = z, x_2 = \dot{z}$, then

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -x_1 + u
\end{align*}
\]

The end cost is $\phi(x) = -x_1$; path cost $R(x, u, t) = 0$.

The HJB takes the form:

\[
-\partial_t J = \min_u \left( x_2 \frac{\partial J}{\partial x_1} - x_1 \frac{\partial J}{\partial x_2} + \frac{\partial J}{\partial x_2} u \right)
\]

\[
= x_2 \frac{\partial J}{\partial x_1} - x_1 \frac{\partial J}{\partial x_2} - \left| \frac{\partial J}{\partial x_2} \right|, \quad u = -\operatorname{sign} \left( \frac{\partial J}{\partial x_2} \right)
\]
Example: Mass on a spring

We try \( J(t, x) = \psi_1(t)x_1 + \psi_2(t)x_2 + \alpha(t) \). The HJBE reduces to the ordinary differential equations

\[
\begin{align*}
\dot{\psi}_1 &= \psi_2 \\
\dot{\psi}_2 &= -\psi_1 \\
\dot{\alpha} &= -|\psi_2|
\end{align*}
\]

These equations must be solved for all \( t \), with final boundary conditions \( \psi_1(T) = -1, \psi_2(T) = 0 \) and \( \alpha(T) = 0 \).

Note, that the optimal control only requires \( \partial_x J(x, t) \), which in this case is \( \psi(t) \) and thus we do not need to solve \( \alpha \). The solution for \( \psi \) is

\[
\begin{align*}
\psi_1(t) &= -\cos(t - T) \\
\psi_2(t) &= \sin(t - T)
\end{align*}
\]
Example: Mass on a spring

The optimal control is

\[ u(x, t) = -\text{sign}(\psi_2(t)) = -\text{sign}(\sin(t - T)) \]

As an example consider \( T = 2\pi \). Then, the optimal control is

\[ u = -1, \quad 0 < t < \pi \]
\[ u = 1, \quad \pi < t < 2\pi \]
Pontryagin minimum principle

The HJB equation is a PDE with boundary condition at future time. The PDE is solved using discretization of space and time.

The solution is an optimal cost-to-go for all $x$ and $t$. From this we compute the optimal trajectory and optimal control.

An alternative approach is a variational approach that directly finds the optimal trajectory and optimal control.
Pontryagin minimum principle

We can write the optimal control problem as a constrained optimization problem with independent variables $u(0 \to T)$ and $x(0 \to T)$

$$
\min_{u(0 \to T), x(0 \to T)} \phi(x(T)) + \int_0^T dt R(x(t), u(t), t)
$$

subject to the constraint

$$
\dot{x} = f(x, u, t)
$$

and boundary condition $x(0) = x_0$.

Introduce the Lagrange multiplier function $\lambda(t)$:

$$
C = \phi(x(T)) + \int_0^T dt \left[ R(t, x(t), u(t)) - \lambda(t)(f(t, x(t), u(t)) - \dot{x}(t)) \right]
$$

$$
= \phi(x(T)) + \int_0^T dt \left[ -H(t, x(t), u(t), \lambda(t)) + \lambda(t)\dot{x}(t) \right]
$$

$$
-H(t, x, u, \lambda) = R(t, x, u) - \lambda f(t, x, u)
$$
Derivation PMP

The solution is found by extremizing $C$. This gives a necessary but not sufficient condition for a solution.

If we vary the action wrt to the trajectory $x$, the control $u$ and the Lagrange multiplier $\lambda$, we get:

$$
\delta C = \phi_x(x(T))\delta x(T) \\
+ \int_0^T dt \left[ -H_x \delta x(t) - H_u \delta u(t) + (-H_\lambda + \dot{x}(t)) \delta \lambda(t) + \lambda(t) \delta \dot{x}(t) \right] \\
= (\phi_x(x(T)) + \lambda(T)) \delta x(T) \\
+ \int_0^T dt \left[ (-H_x - \dot{\lambda}(t)) \delta x(t) - H_u \delta u(t) + (-H_\lambda + \dot{x}(t)) \delta \lambda(t) \right]
$$

For instance, $H_x = \frac{\partial H(t,x(t),u(t),\lambda(t))}{\partial x(t)}$.

We can solve $H_u(t, x, u, \lambda) = 0$ for $u$ and denote the solution as

$$u^*(t, x, \lambda)$$
Assumes $H$ convex in $u$.

The remaining equations are

$$
\begin{align*}
\dot{x} &= H_\lambda(t, x, u^*(t, x, \lambda), \lambda) \\
\dot{\lambda} &= -H_x(t, x, u^*(t, x, \lambda), \lambda)
\end{align*}
$$

with boundary conditions

$$
\begin{align*}
x(0) &= x_0 \\
\lambda(T) &= -\phi_x(x(T))
\end{align*}
$$

Mixed boundary value problem.
Again mass on a spring

Problem

\[ \dot{x}_1 = x_2, \quad \dot{x}_2 = -x_1 + u \]
\[ R(x, u, t) = 0 \quad \phi(x) = -x_1 \]

Hamiltonian

\[ H(t, x, u, \lambda) = -R(t, x, u) + \lambda^T f(t, x, u) = \lambda_1 x_2 + \lambda_2 (-x_1 + u) \]
\[ H^*(t, x, \lambda) = \lambda_1 x_2 - \lambda_2 x_1 - |\lambda_2| \quad u^* = -\text{sign}(\lambda_2) \]

The Hamilton equations

\[ \dot{x} = \frac{\partial H^*}{\partial \lambda} \quad \Rightarrow \quad \dot{x}_1 = x_2, \quad \dot{x}_2 = -x_1 - \text{sign}(\lambda_2) \]
\[ \dot{\lambda} = -\frac{\partial H^*}{\partial x} \quad \Rightarrow \quad \dot{\lambda}_1 = \lambda_2, \quad \dot{\lambda}_2 = -\lambda_1 \]

with \( x(t = 0) = x_0 \) and \( \lambda(t = T) = (1, 0) \).
Example

Consider the control problem:

\[
\begin{align*}
  dx &= udt \\
  C &= \frac{\alpha}{2}x(T)^2 + \int_{t_0}^{T} dt \frac{1}{2}u(t)^2
\end{align*}
\]

with initial condition \(x(t_0)\).

Solve the control problem using the PMP formalism.
Solution

The PMP recipe is

1. Construct the Hamiltonian

\[ H(t, x, u, \lambda) = -R(t, x, u) + \lambda f(t, u, x) = -\frac{1}{2}u^2 + \lambda u \]

2. Construct the optimized Hamiltonian

\[ H^*(t, x, \lambda) = H(t, x, u^*, \lambda) = \frac{1}{2}\lambda^2 \quad u^* = \lambda \]

3. Solve the Hamilton equations of motion

\[ \frac{dx}{dt} = \frac{\partial H^*}{\partial \lambda} = \lambda \]
\[ \frac{d\lambda}{dt} = -\frac{\partial H^*}{\partial x} = 0 \]
with boundary conditions $x(t_0)$ and $\lambda(t = T) = -\alpha x(T)\footnote{Note, that $\phi(x) = \frac{g}{2}x^2$ so that $\phi_x = \alpha x$.}$. The solution for $\lambda$ is constant $\lambda(t) = \lambda = -\alpha x(T)$. The solution for $x(t)$ is

$$x(t) = x(t_0) + \lambda(t - t_0)$$

Combining these two results, we get $\lambda = -\alpha x(T) = -\alpha(x(t_0) + \lambda(T - t_0))$, or

$$\lambda = \frac{-\alpha x(t_0)}{1 + \alpha(T - t_0)}$$

Since $u^* = \lambda$, this is the optimal control law.
Brownian bridge

Due to certainty equivalence, this is also the optimal control law for

\[ dx = u dt + d\xi \]

\[ C = \left\langle \alpha \frac{x(T)^2}{2} + \int_{t_0}^{T} dt \frac{1}{2} u(t)^2 \right\rangle \]

For \( \alpha \to \infty \) the process is known as a Brownian bridge.

The control law and dynamics becomes

\[ dx = u dt + d\xi \]

\[ u = \frac{-x(t_0)}{T - t_0} \]

\( x(T) \to 0 \) w.p. 1.
Relation to classical mechanics

The equations look like classical mechanics

\[ \dot{x} = H_\lambda(t, x, u^*(t, x, \lambda), \lambda) \quad x(0) = x_0 \]
\[ \dot{\lambda} = -H_x(t, x, u^*(t, x, \lambda), \lambda) \quad \lambda(T) = -\phi_x(x(T)) \]

In classical mechanics $H$ is called the Hamiltonian. Consider the time evolution of $H$:

\[ \dot{H} = H_t + H_u u + H_x \dot{x} + H_\lambda \dot{\lambda} = H_t \]
\[ H(t, x, u, \lambda) = -R(t, x, u) + \lambda f(t, u, x) \]

So, for problems where $R, f$ do not explicitly depend on time, $H$ is a constant of the motion.
Example

Consider the control problem:

\[
\begin{align*}
\dot{x} &= u dt \\
C &= \int_{t_0}^{T} dt \frac{1}{2} u(t)^2 + V(x(t))
\end{align*}
\]

with initial condition \(x(t_0)\).

1. \( H(x, u, \lambda) = -\frac{1}{2} u^2 - V(x) + \lambda u \)

2. \( u^* = \lambda, \ H^*(x, \lambda) = \frac{1}{2} \lambda^2 - V(x) \)

3. 

\[
\begin{align*}
\dot{x} &= \frac{\partial H^*}{\partial \lambda} = \lambda \\
\dot{\lambda} &= -\frac{\partial H^*}{\partial \lambda} = \frac{\partial V(x)}{\partial x}
\end{align*}
\]

Control cost \( V \) play role of \textit{minus} potential energy.
Control solution has constant \textit{difference} of kinetic energy and state cost
Comments

The solution of the HJB PDE is expensive.

The PMP method is computationally less complicated than the HJB method because it does not require discretisation of the state space.

HJB generalizes to the stochastic case, PMP does not (at least not easy).
Stochastic differential equations

Consider the random walk on the line:

\[ X_{t+1} = X_t + \xi_t \quad \xi_t = \pm 1 \]

with \( x_0 = 0 \). We can compute

\[ X_t = \sum_{i=1}^{t} \xi_i \]

Since \( x_t \) is a sum of random variables, \( x_t \) becomes Gaussian distributed with

\[ \langle x_t \rangle = \sum_{i=1}^{t} \langle \xi_i \rangle = 0 \]

\[ \langle x_t^2 \rangle = \sum_{i,j=1}^{t} \langle \xi_i \xi_j \rangle = \sum_{i=1}^{t} \langle \xi_i^2 \rangle + \sum_{i,j=1, j \neq i}^{t} \langle \xi_i \xi_j \rangle = t \]

Note, that the fluctuations \( \propto \sqrt{t} \).
Stochastic differential equations

In the continuous time limit we define

\[ dX_t = X_{t+dt} - X_t = dW_t \]

with \( dW_t \) an infinitesimal mean zero Gaussian variable with \( \langle dW_t^2 \rangle = \nu dt \).

Then

\[
\frac{d}{dt} \langle X \rangle = \lim_{dt \to 0} \frac{\langle X_{t+dt} - X_t \rangle}{dt} = \lim_{dt \to 0} \frac{\langle dW_t \rangle}{dt} = 0
\]

\[
\frac{d}{dt} \langle X^2 \rangle = \lim_{dt \to 0} \frac{\langle X_{t+dt}^2 - X_t^2 \rangle}{dt} = \lim_{dt \to 0} \frac{\langle (X_t + dW_t)^2 - X_t^2 \rangle}{dt} = \lim_{dt \to 0} \frac{\langle dW_t^2 \rangle}{dt} = \nu
\]

So for initial state \( x_0 \), \( \langle X \rangle(t) = x_0 \) and \( \langle X^2 \rangle(t) = \nu t \) which fully specifies the Gaussian distribution:

\[
\rho(x, t|x_0, 0) = \frac{1}{\sqrt{2\pi \nu t}} \exp \left( -\frac{(x - x_0)^2}{2\nu t} \right)
\]
Consider the stochastic differential equation

\[ dX_t = f(X_t, t)dt + dW_t \]

\( dW_t \) is a Wiener process with \( \langle dW_t \rangle = 0, \langle dW_t^2 \rangle = \nu dt \).

The probability to find the particle at \( y \) at time \( t + dt \) given that it was at \( x \) at time \( t \) is given by

\[ p(dX_t, t + dt|X, t) = \langle \delta(dX_t - f(X, t)dt - dW_t) \rangle \]

where \( \langle \rangle \) is expectation wrt the Wiener process.
Kolmogorov backward equation

Define $\psi(x, t) = p(z, T|x, t)$ the probability to reach a future state $z$ at time $T$, given that it is currently at $x, t$. Clearly,

$$\psi(x, t) = p(z, T|x, t) = \int dy p(z, T|y, t + dt)p(y, t + dt|x, t)$$

$$= \int dy \psi(y, t + dt) \langle \delta(y - x - f(x, t)dt - \xi) \rangle_\xi$$

$$= \langle \psi(x + f(x, t)dt + \xi, t + dt) \rangle_\xi$$

$$= \psi(x, t) + dt \partial_t \psi(x, t) + \langle f(x, t)dt + \xi \rangle_\xi \nabla \psi(x, t)$$

$$+ \frac{1}{2} \left\langle (f(x, t)dt + \xi)^2 \right\rangle_\xi \nabla^2 \psi(x, t)$$

Thus,

$$-\partial_t \psi(x, t) = f(x, t)\nabla \psi(x, t) + \frac{1}{2} \nu \nabla^2 \psi(x, t) \quad \psi(x, T) = \delta(z - x)$$

This equation is known as the Kolmogorov backwards equation.
Fokker Plank (forward) equation

We can similarly derive a forward equation for the quantity \( \rho(x, t) = p(x, t|x_0, 0) \).

\[
\rho(y, t + dt) = \int dxp(y, t + dt|x, t)\rho(x, t)
= \int dx \langle \delta(y - x - f(x, t)dt - \xi) \rangle \xi \rho(x, t)
= \frac{1}{1 + f'(y, t)dt} \langle \rho(y - f(y, t)dt - \xi, t) \rangle \xi
= \frac{1}{1 + f'(y, t)dt} \left. \rho(y, t) - (f(y, t)dt + \xi)\nabla \rho(y, t) + \frac{1}{2}(f(y, t) + \xi)^2\nabla^2 \rho(y, t) \right\rangle \xi
= \rho(y, t) - \nabla(f(y, t)\rho(y, t))dt + \frac{1}{2}\nu\nabla^2 \rho(y, t)dt
\]

Thus,

\[
\partial_t \rho(x, t) = -\nabla(f(x, t)\rho(x, t)) + \frac{1}{2}\nu\nabla^2 \rho(x, t), \quad \rho(x, 0) = \delta(x - x_0)
\]
Example: Brownian motion

\[ dX_t = dW_t \quad \langle dW_t^2 \rangle = \nu dt \]

\[ \rho(x, t) = p(x, t|x_0, 0) = \frac{1}{\sqrt{2\pi\nu t}} \exp\left(-\frac{(x-x_0)^2}{2\nu t}\right) \]

\[ \psi(x, t) = p(z, T|x, t) = \frac{1}{\sqrt{2\pi\nu(T-t)}} \exp\left(-\frac{(x-z)^2}{2\nu(T-t)}\right) \]

Denote \( p(x_2, t_2|x_1, t_1) \) the probability to go from \( x_1, t_1 \) to \( x_2, t_2 \). Then

\[ \rho(x, t) = p(x, t|x_0, t_0) \]

\[ \psi(x, t) = p(z, T|x, t) \]
Stochastic optimal control

Consider a stochastic dynamical system

\[ dX_t = f(t, X_t, u)dt + dW_t \]

\[ dW_t \text{ Gaussian noise } \langle dW_t dW_t^T \rangle = \nu(t, x, u)dt. \]

The cost becomes an expectation:

\[ C(t, x, u) = \left\langle \phi(X_T) + \int_t^T d\tau R(t, X_t, u(X_t, t)) \right\rangle \]

over all stochastic trajectories starting at \( x \) with control function \( u(\cdot, t). \)

Optimize with respect to the set of functions \( u(\cdot, t). \)
We obtain the Bellman recursion

\[ J(t, x_t) = \min_u \{ R(t, x_t, u_t) + \langle J(t + dt, X_{t+dt}) \rangle \} \]

\[ \langle J(t + dt, X_{t+dt}) \rangle = \int dx_{t+dt} \mathcal{N}(x_{t+dt}|x_t, \nu dt) J(t + dt, x_{t+dt}) \]

\[ \langle dx \rangle = f(x, u, t)dt \]

\[ \langle dx^2 \rangle = \nu(t, x, u)dt \]

Thus,

\[ -\partial_t J(t, x) = \min_u \left( R(t, x, u) + f(x, u, t) \partial_x J(x, t) + \frac{1}{2} \nu(t, x, u) \partial^2_x J(x, t) \right) \]

with boundary condition \( J(x, T) = \phi(x) \).
**Linear Quadratic control**

The dynamics is linear

\[
dX_t = [A(t)X_t + B(t)u_t + b(t)]dt + \sum_{j=1}^{m} (C_j(t)X_t + D_j(t)u_t + \sigma_j(t))dW_j, \quad \langle dW_j dW_{j'} \rangle = \delta_{jj'}
\]

The cost function is quadratic

\[
\phi(x) = \frac{1}{2} x^T G x
\]

\[
R(x, u, t) = \frac{1}{2} x^T Q(t)x + u^T S(t)x + \frac{1}{2} u^T R(t)u
\]

In this case the optimal cost-to-go is quadratic in \(x\):

\[
J(t, x) = \frac{1}{2} x^T P(t)x + \alpha^T (t)x + \beta(t)
\]

\[
u_t = -\Psi(t)x_t - \psi(t)
\]
Substitution in the HJB equation yields ODEs for $P, \alpha, \beta$:

\[
-\dot{P} = PA + A^T P + \sum_{j=1}^{m} C_j^T P C_j + Q - \hat{S}^T \hat{R}^{-1} \hat{S}
\]

\[
-\dot{\alpha} = [A - B\hat{R}^{-1} \hat{S}]^T \alpha + \sum_{j=1}^{m} [C_j - D_j \hat{R}^{-1} \hat{S}]^T P \sigma_j + Pb
\]

\[
\dot{\beta} = \frac{1}{2} \left| \sqrt{\hat{R}} \psi \right|^2 - \alpha^T b - \frac{1}{2} \sum_{j=1}^{m} \sigma_j^T P \sigma_j
\]

\[
\dot{\hat{R}} = R + \sum_{j=1}^{m} D_j^T P D_j
\]

\[
\dot{\hat{S}} = B^T P + S + \sum_{j=1}^{m} D_j^T P C_j
\]

\[
\Psi = \hat{R}^{-1} \hat{S}
\]

\[
\psi = \hat{R}^{-1} (B^T \alpha + \sum_{j=1}^{m} D_j^T P \sigma_j)
\]

with $P(t_f) = G$ and $\alpha(t_f) = \beta(t_f) = 0$. 
Example

Find the optimal control for the dynamics

$$dX_t = (X + u)dt + dW_t, \quad \langle dW_t^2 \rangle = \nu dt$$

with end cost $\phi(x) = 0$ and path cost $R(x, u) = \frac{1}{2}(Qx^2 + Ru^2)$.

The Riccati equations reduce to

$$-\dot{P} = 2P + Q - R^{-1}P^2$$
$$-\dot{\alpha} = (1 - R^{-1}P)\alpha = 0$$
$$\dot{\beta} = \frac{1}{2}R^{-1}\alpha^2 - \frac{1}{2}\nu P = -\frac{1}{2}\nu P$$

with $P(T) = \alpha(T) = \beta(T) = 0$ and

$$u(x, t) = -P(t)x$$
The solution is

\[ P(t) = R \frac{\exp(2 \sqrt{1 + R^{-1}Q(T - t)}) - 1}{\frac{1}{1 + \sqrt{1 + R^{-1}Q}} \exp(2 \sqrt{1 + R^{-1}Q(T - t)}) - \frac{1}{1 - \sqrt{1 + R^{-1}Q}}} \]

The optimal control is \( u(x, t) = -R^{-1}P(t)x \).
Comments

Note, that in the last example the optimal control is independent of \( \nu \), i.e. optimal stochastic control equals optimal deterministic control.

In general:

- If \( C_j = D_j = 0 \) (only ’additive noise’) \( \dot{P}, \dot{\alpha} \) independent of noise \( \sigma \), \( \dot{\beta} \) depends on \( \sigma \), but control independent of \( \beta \). Thus control independent of \( \sigma \) (certainty equivalence)
- If \( C_j \neq 0 \) or \( D_j \neq 0 \), control depends on \( C_j, D_j, \sigma_j \) (no certainty equivalence)
Example: Portfolio selection

Consider a market with $p$ stocks and one bond. The bond price process is subject to the following deterministic ordinary differential equation:

$$dP_0(t) = r(t)P_0(t)dt, \quad P_0(0) = p_0 > 0$$  \hspace{1cm} (2)

The other assets have price processes $P_i(t), i = 1, \ldots, p$ satisfying stochastic differential equations

$$dP_i(t) = P_i(t) \left( b_i(t)dt + \sum_{j=1}^{m} \sigma_{ij}(t) d\xi_j(t) \right), \quad P_i(0) = p_i > 0$$  \hspace{1cm} (3)

Consider an investor whose total wealth at time $t$ is denoted by $x(t)$

$$x(t) = \sum_{i=0}^{p} N_i(t)P_i(t)$$  \hspace{1cm} (4)

---

8 This section is from [Yong and Zhou, 1999] section 6.8 (pg. 335).
with $N_i$ the number of stocks/bond of type $i$. Then

$$dx(t) = \left( r(t)x(t) + \sum_{i=1}^{p} (b_i(t) - r(t))u_i(t) \right) dt + \sum_{i=1}^{p} \sum_{j=1}^{m} \sigma_{ij}(t)u_i(t)d\xi_j(t)$$

(5)

with $u_i(t) = N_i(t)P_i(t), i = 1, \ldots, p$ the portfolio of the investor.

The objective of the investor is to maximize the mean terminal wealth $\langle x(t_f) \rangle$ and minimize at the same time the variance

$$\Sigma^2 = \langle x(t_f)^2 \rangle - \langle x(t_f) \rangle^2$$

This is a multi-objective optimization problem with an efficient frontier of optimal solutions: for each given mean there is a minimal variance.

These pairs can be found by minimizing the single objective criterion

$$\mu \Sigma^2 - \langle x(t_f) \rangle$$

(6)

for different values of the weighting factor $\mu$. 

Bert Kappen
This objective, however, is not an expectation value of some stochastic quantity due to the $\langle \cdot \rangle^2$ term. Consider a slightly different problem, minimizing the objective

$$\left\langle \mu x(t_f)^2 - \lambda x(t_f) \right\rangle$$

(7)

which is of the standard stochastic optimization form. One can show that one can construct a solution of Problem 6 by solving problem 7 for suitable $\lambda(\mu)$. 9

Our goal is thus to minimize eq. 7 subject to the stochastic dynamics eq. 5.

This is an LQ problem. The solution is computed from the Ricatti equations

$$u_i(x, t) = \psi_i(t)x + \phi_i(t)$$

As an example we consider the simplest possible case: $p = m = 1$ and $r, b, \sigma$ independent of time.

---

9 and finding $\lambda$ from

$$\lambda = 1 + 2\mu \left\langle x(t_f) \right\rangle (\lambda, \mu)$$

([Yong and Zhou, 1999] Theorem 8.2 pg. 338)
Parameter values are: \( p = m = 1 \). Trading period is one year weekly. Annual bond rate 5 \% \((r = 0.0009758)\), annual expected stock rate is 10 \% \((b = 0.0019)\), volatility \( \sigma = 2b \). \( x_0 = 2 \). Shows \( \text{var} \, x \) versus \( \langle x \rangle \) scatter plot for various values of \( \mu \). Small \( \mu \) corresponds to risky investments with high expected return and large fluctuation. \( \mu \to \infty \) corresponds to riskless investment in bond only and a return of 5 \%.

\( \mu = 10 \) corresponds to \( \langle x \rangle = 3 \) and \( \sqrt{\text{var}} = 0.2 \).
Simulation of optimal control with $\mu = 10$, The optimal strategy is to borrow many stocks and sell them as soon as the objective is achieved.

Indeed, $\langle x \rangle = 3$ as expected. The strategy to get at this 50 % increase in wealth is to buy many stocks and hope they will give the expected wealth increase. As soon as this occurs, all stocks are sold and the money is put in the bank.
Why stochastic control?
Why optimal control theory?

Options:

• Reinforcement learning/optimal control theory
  – generic and flexible
  – modular (cost, dynamics, environment)
  – can be learned from data or specified explicitly

• Adaptive control theory
  – tracking and stabilization
  – fast
How to control?

Hard problems:
- a learning and exploration problem
- a stochastic optimal control computation
- a representation problem $u(x, t)$
The idea: Control, Inference and Learning

Linear Bellman equation and path integral solution
Express a control computation as an inference computation.
The idea: Control, Inference and Learning

Linear Bellman equation and path integral solution
Express a control computation as an inference computation.
Compute optimal control using MC sampling
The idea: Control, Inference and Learning

Linear Bellman equation and path integral solution
Express a control computation as an inference computation.
Compute optimal control using MC sampling

Importance sampling
Accellerate with importance sampling, a state-feedback controller
The idea: Control, Inference and Learning

Linear Bellman equation and path integral solution
Express a control computation as an inference computation.
Compute optimal control using MC sampling

Importance sampling
Accellerate with importance sampling, a state-feedback controller
Learn controller from self-generated data
The idea: Control, Inference and Learning

Linear Bellman equation and path integral solution
Express a control computation as an inference computation.
Compute optimal control using MC sampling

Importance sampling
Accellerate with importance sampling, a state-feedback controller
Learn controller from self-generated data

Optimal importance sampler is optimal control
The idea: Control, Inference and Learning

Linear Bellman equation and path integral solution
Express a control computation as an inference computation.
Compute optimal control using MC sampling

Importance sampling
Accellerate with importance sampling, a state-feedback controller
Learn controller from self-generated data

Optimal importance sampler is optimal control

Learn a good importance sampler using PICE
Path integral control

\[
dx_i = f_i(x, t) dt + \sum_j g_{ia}(x, t)(u_a dt + d\xi_a)
\]

\[
C(t, x, u(t \to T)) = \left\langle \phi(x(T)) + \int_t^T ds V(x, t) + \frac{1}{2} \sum_{ab} R_{ab} u_a u_b \right\rangle
\]

with \( \langle d\xi_a d\xi_b \rangle = \nu_{ab} dt \).

The cost is an expectation over all stochastic trajectories starting at \( x \) with control path \( u(t \to T) \).

The stochastic HJB equation becomes

\[
-\partial_t J = \min_u \left( \frac{1}{2} u^T R u + V + (\nabla J)^T (f + gu) + \frac{1}{2} \text{Tr} \left( g v g^T \nabla^2 J \right) \right)
\]

which we need to solve with end boundary condition \( J(x, t_f) = \phi(x) \).
Path integral control

Minimization wrt \( u \) yields:

\[
  u = -R^{-1}g^T \nabla J
\]

\[
  -\partial_t J = -\frac{1}{2}(\nabla J)^T gR^{-1}g^T(\nabla J) + V + (\nabla J)^T f + \frac{1}{2}\text{Tr}(gv^T\nabla^2 J)
\]

Define \( \psi(x, t) \) through \( J(x, t) = -\lambda \log \psi(x, t) \) and impose a relation between \( R \) and \( \nu \):

\[
  R = \lambda \nu^{-1}
\]

with \( \lambda \) a positive number.
Path integral control

Then the HJB becomes linear in $\psi$

$$\partial_t \psi = \left( \frac{V}{\lambda} - f^T \nabla - \frac{1}{2} \text{Tr} \left( g v g^T \nabla^2 \right) \right) \psi$$

with end condition $\psi(x, T) = \exp(-\phi(x)/\lambda)$ (our Kolmogorov backward equation) \footnote{We sketch the derivation for $g = 1$.}

$$-\frac{1}{2} (\nabla J)^T R^{-1} (\nabla J) + \frac{1}{2} \text{Tr} (v \nabla^2 J) = -\frac{1}{2} \sum_{ij} \nabla_i J R^{-1}_{ij} \nabla_j J + \frac{1}{2} \lambda \sum_{ij} R^{-1}_{ij} \nabla_i J \nabla_j J$$

$$= \frac{1}{2} \sum_{ij} R^{-1}_{ij} \left( -\nabla_i J \nabla_j J + \lambda \nabla_i J \nabla_j J \right)$$

$$= \frac{1}{2} \sum_{ij} R^{-1}_{ij} \left( -\lambda^2 \frac{1}{\psi} \nabla_i J \psi \nabla_j J \right)$$

since

$$-\nabla_i J \nabla_j J = -\lambda^2 \frac{1}{\psi^2} \nabla_i \psi \nabla_j \psi$$

$$\nabla_i J = -\lambda \nabla_i \nabla_j \log \psi = -\lambda \nabla_i \left( \frac{1}{\psi} \nabla_j \psi \right) = \lambda \frac{1}{\psi^2} \nabla_i \psi \nabla_j \psi - \lambda \frac{1}{\psi} \nabla_i \psi$$
Path integral control

The linearity allows us to reverse the direction of time.

We identify $\psi(x, t) \propto p(z, T|x, t)$, then the Bellman equation

$$\partial_t \psi = \left( \frac{V}{\lambda} - f^T \nabla - \frac{1}{2} \text{Tr} (g v g^T \nabla^2) \right) \psi$$

can be interpreted as a Kolmogorov backward equation for the process

$$dx_i = f_i(x, t) dt + \sum_a g_{ia}(x, t) d\xi_a$$

$$x(t) = \dagger \quad \text{with probability} \quad V(x, t) dt / \lambda$$

$$x(T) = \dagger \quad \text{with probability} \quad \phi(x) / \lambda$$
Path integral control

The corresponding forward equation is

\[ \partial_t \rho = -\frac{V}{\lambda} \rho - \nabla(f \rho) + \frac{1}{2} \text{Tr} \nabla^2 v g v^T \rho \]

with \( \rho(x, t) = p(x, t|z, 0) \) and \( \rho(x, 0) = \delta(x - z) \).
Feynman-Kac formula

Denote $Q(\tau|x, s)$ the distribution over uncontrolled trajectories that start at $x, t$:

$$dx = f(x, t)dt + g(x, t)d\xi$$

with $\tau$ a trajectory $x(t \rightarrow T)$. Then

$$\psi(x, t) = \int dQ(\tau|x, t) \exp\left(-\frac{S(\tau)}{\lambda}\right)$$

$$S(\tau) = \phi(x(T)) + \int_{t}^{T} dsV(x(s), s)$$

$\psi$ can be computed by forward sampling the uncontrolled process.
Alternative derivation

Uncontrolled dynamics specifies distribution $q(\tau|x, t)$ over trajectories $\tau$ from $x, t$.

Cost for trajectory $\tau$ is $S(\tau|x, t) = \phi(x_T) + \int_t^T ds V(x_s, s)$.

Find optimal distribution $p(\tau|x, t)$ that minimizes $\mathbb{E}_p S$ and is 'close' to $q(\tau|x, t)$.
KL control

Find $p^*$ that minimizes

$$C(p) = KL(p|q) + \mathbb{E}_p S \quad KL(p|q) = \int d\tau p(\tau|x,t) \log \frac{p(\tau|x,t)}{q(\tau|x,t)}$$

The optimal solution is given by

$$p^*(\tau|x,t) = \frac{1}{\psi(x,t)} q(\tau|x,t) \exp(-S(\tau|x,t))$$

$$\psi(x,t) = \int d\tau q(\tau|x,t) \exp(-S(\tau|x,t)) = \mathbb{E}_q e^{-S}$$

The optimal cost is:

$$C(p^*) = - \log \psi(x,t)$$
Controlled diffusions

In the case of controlled diffusions, $p(\tau|x, t)$ is parametrised by functions $u(x, t)$, $q(\tau|x, t)$ corresponds to $u(x,t) = 0$:

$$dX_t = f(X_t, t)dt + g(X_t, t)(u(X_t, t)dt + dW_t) \quad \mathbb{E}(dW_idW_j) = \nu_{ij}dt$$

$$C(p) = \mathbb{E}_p \left( \int dt \frac{1}{2} u(X_t, t)^T \nu^{-1} u(X_t, t) + S(\tau|x, t) \right)$$

$J(x, t) = - \log \psi(x, t)$ is the solution of the Bellman equation. \[12\]

$p^*$ is generated by optimal control $u^*(x, t)$:

$$u^*(x, t)dt = \mathbb{E}_p^*(dW_t) = \frac{\mathbb{E}_q(dWe^{-S})}{\mathbb{E}_q(e^{-S})}$$

$\psi, u^*$ can be computed by forward sampling from $q$.

\[12\] For the experts: The Bellman equation becomes a linear PDE in terms of $\psi$ and the solution is the Feynman-Kac path integral.
Delayed choice

\[
dx = u_t dt + d\xi_t \quad \langle \xi_t^2 \rangle = \nu dt
\]

\[V = 0, \text{ path cost is } \frac{1}{2}u^2 \text{ and end cost } \phi(z = \pm 1) = 0, \phi(z) = \infty \text{ else encodes two targets at } z = \pm 1 \text{ at } t = T.\]

PI recipe:

1.

\[\psi(x, t) = \int dQ(\tau | x, t) \exp(-S(\tau)/\lambda)\]

\[S(\tau) = \phi(x(T))\]

\[\psi(x, t) = \int dz q(z, T | x, t) \exp(-\phi(z)/\lambda) = q(1, T | x, t) + q(-1, T | x, t)\]

\[q(z, T | x, t) = \mathcal{N}(z | x, \nu(T - t))\]
2. Compute

\[ J(x, t) = -\lambda \log \psi(x, t) = \frac{1}{T-t} \left( \frac{1}{2} x^2 - \nu(T-t) \log 2 \cosh \frac{x}{\nu(T-t)} \right) \]

3.

\[ u(x, t) = -\nabla J(x, t) = \frac{1}{T-t} \left( \tanh \frac{x}{\nu(T-t)} - x \right) \]
Delayed choice

\[ dx = u dt + d\xi \quad \langle \xi^2 \rangle = \nu dt \]

\[ V = 0, \text{ path cost is } \frac{1}{2}u^2, \phi(x = \pm 1) = 0 \text{ and } \phi(x) = \infty, \text{ else.} \]
Delayed choice

\[ dx = u dt + d\xi \quad \langle \xi^2 \rangle = v dt \]

\[ V = 0, \text{ path cost is } \frac{1}{2} u^2, \phi(x = \pm 1) = 0 \text{ and } \phi(x) = \infty, \text{ else.} \]
Delayed choice

\[ dx = u dt + d\xi \quad \langle \xi^2 \rangle = \nu dt \]

\( V = 0 \), path cost is \( \frac{1}{2}u^2 \), \( \phi(x = \pm 1) = 0 \) and \( \phi(x) = \infty \), else.

"When the future is uncertain, delay your decisions."
Estimating optimal control by sampling

For given $x, t$, the optimal control is given by

$$udt = \int dP(\tau) d\xi(\tau) = \frac{\int dQ(\tau) \exp(-S(\tau)/\lambda) d\xi(\tau)}{\int dQ(\tau) \exp(-S(\tau)/\lambda)}$$

We generate $N$ trajectories $x^\mu_{t:T}$ starting at $x, t$ with initial noise $d\xi^\mu$. Define

$$S^\mu = \sum_{s=t}^T V(x^\mu_s, s) dt + \phi(x^\mu_T)$$

$$udt = \frac{\sum_\mu d\xi^\mu \exp(-S^\mu/\lambda)}{\sum_\mu \exp(-S^\mu/\lambda)}$$

Unbiased, but inefficient.
Sampling with uncontrolled dynamics is theoretically correct, but inefficient in practice.
Consider simple 1-d sampling problem. Given $q(x)$, compute

$$a = \text{Prob}(x < 0) = \int_{-\infty}^{\infty} I(x)q(x)dx$$

with $I(x) = 0, 1$ if $x > 0, x < 0$, respectively.

Naive method: generate $N$ samples $X_i \sim q$

$$\hat{a} = \frac{1}{N} \sum_{i=1}^{N} I(X_i)$$
Importance sampling

Consider another distribution \( p(x) \). Then

\[
a = \text{Prob}(x < 0) = \int_{-\infty}^{\infty} I(x) \frac{q(x)}{p(x)} p(x) dx
\]

Importance sampling: generate \( N \) samples \( X_i \sim p \)

\[
\hat{a} = \frac{1}{N} \sum_{i=1}^{N} I(X_i) \frac{q(X_i)}{p(X_i)}
\]

Unbiased (= correct) for any \( p \)!
The distribution

\[ p^*(x) = \frac{q(x)I(x)}{a} \]

is the optimal importance sampler.

One sample \( X \sim p^* \) is sufficient to estimate \( a \):

\[ \hat{a} = I(X) \frac{q(X)}{p^*(X)} = a \]
Importance sampling and control

In the case of control we must compute

\[ J(x, t) = - \log \mathbb{E}_q e^{-s} \quad u^*(x, t) = \frac{\mathbb{E}_q (dW e^{-s})}{\mathbb{E}_q (e^{-s})} \]

Instead of samples from uncontrolled dynamics \( q(u = 0) \), we sample with \( p(u \neq 0) \).

\[ \mathbb{E}_q e^{-s} = \mathbb{E}_p e^{-s} \frac{dq}{dp} \]

We can choose any \( p \), ie. any sampling control \( u \).
Importance sampling and control

In the case of control we must compute

\[ J(x, t) = - \log \mathbb{E}_q e^{-S} \quad u^*(x, t) = \frac{\mathbb{E}_q (dWe^{-S})}{\mathbb{E}_q (e^{-S})} \]

Instead of samples from uncontrolled dynamics \( q (u = 0) \), we sample with \( p (u \neq 0) \).

\[
\mathbb{E}_q e^{-S} = \mathbb{E}_p e^{-S} \frac{dq}{dp} = \mathbb{E}_p e^{-Su}
\]

\[
e^{-Su} = e^{-S - \int_t^T dt \frac{1}{2} u(x, t)^T \nu^{-1} u(x, t) - \int_t^T u(x, t)^T \nu^{-1} dW_t}
\]

\( \mathbb{E}_p = \mathbb{E}_u \). We can choose any \( p \), ie. any sampling control \( u \).
Relation between optimal sampling and optimal control

Definition 1.

1. The weight of a path is defined as \( \alpha^u = \frac{e^{-S^u(t_0, x_0)}}{\mathbb{E}[e^{-S^u(t_0, x_0)}]} \).

2. The fraction of effective samples is \( FES = \frac{1}{\mathbb{E}[(\alpha^u)^2]} = \frac{1}{\text{Var}(\alpha^u)+1} \).

Theorem 2. Let \( 0 < \epsilon < 1 \). Then:

1. \( (u^* - u)'(u^* - u) \leq \frac{\epsilon}{t_1-t_0} \) point-wise implies \( \text{Var}(\alpha^u) \leq \frac{\epsilon}{1-\epsilon} \)

2. \( \text{Var}(\alpha^u) \leq \epsilon \) implies \( \int_{t_0}^{t_1} \langle u^* - u \rangle' \langle u^* - u \rangle dt \leq \epsilon \).

1. Better \( u \) (in the sense of optimal control) provides a better sampler (in the sense of effective sample size).

2. Optimal \( u = u^* \) (in the sense of optimal control) requires only one sample.
The Cross-entropy method

Let $X$ be a random variable taking values in the space $\mathcal{X}$. Let $f_v(x)$ be a family of probability density function on $\mathcal{X}$ parametrized by $v$ and $h(x)$ be a positive function. Suppose that we are interested in the expectation value

$$a = \mathbb{E}_u h = \int dx f_u(x) h(x)$$

where $\mathbb{E}_u$ denotes expectation with respect to the pdf $f_u$ for a particular value of $v = u$.

The optimal importance sampling distribution is $g^*(x) = h(x)f_u(x)/a$.

The cross entropy method suggests to find the distribution $f_v$ in the parametrized family of distributions that minimises the KL divergence

$$KL(g^*|f_v) = \int dx g^*(x) \log \frac{g^*(x)}{f_v(x)} \propto -\mathbb{E}_{g^*} \log f_v(X) \propto -\mathbb{E}_u h(X) \log f_v(X) = -D(v)$$
The Cross-entropy method

We can use again importance sampling to compute $D(v)$:

\[ D(v) = \mathbb{E}_u h(X) \log f_v(X) = \mathbb{E}_w h(X) \frac{f_u(X)}{f_w(X)} \log f_v(X) \]

We estimate the expectation value by drawing $N$ samples from $f_w$. If $D$ is convex and differentiable with respect to $v$, the optimal $v$ is given by

\[ \frac{1}{N} \sum_{i=1}^{N} h(X_i) \frac{f_u(X_i)}{f_w(X_i)} d \log f_v(X_i) = 0 \quad X_i \sim f_w \]
The CE algorithm

Initialize $w_0 = u$.

for $k = 0, \ldots, K$ do

generate $N$ samples $X_{1:N}$ from $f_{w_k}$

compute $v$ by solving

$$\frac{1}{N} \sum_{i=1}^{N} h(X_i) \frac{f_u(X_i)}{f_w(X_i)} \log \frac{f_v(X_i)}{f_v(X_i)} = 0$$

Set $w_{k+1} = v$.

end for

return $w_K$
The CE method for PI control: Preliminaries

Let $X$ denote the space of continuous trajectories on the interval $[t, T]$: $\tau = X(s), t \leq s \leq T$ with fixed initial value $X(t) = x$ satisfying the dynamics

$$dX_t = f(X_t, t)dt + g(X_t, t)(u(X_t, t)dt + dW_t)$$

Denote $p_u(\tau)$ the distribution over trajectories $\tau$ with control $u$.

We wish to compute a near optimal control $\hat{u}$ such that $p_{\hat{u}}$ is close to $p^*$. Following the CE argument, we minimise

$$KL(p^*|p_{\hat{u}}) = \frac{1}{\psi(t, x)} \mathbb{E}_p e^{-S(t, x, u)} \int_t^T ds \left( \frac{1}{2} \hat{u}(s, X(s))^2 - \hat{u}(s, X(s)) \left( u(s, X(s)) + \frac{dW_s}{ds} \right) \right)$$

The expression must be optimized with respect to the functions $\hat{u}_{t:T} = \{\hat{u}(s, X_s), t \leq s \leq T\}$. It is independent of the sampling control $u_{t:T} = \{u(s, X_s), t \leq s \leq T\}$. 
The CE method for PI control: Time-dependent solution

We now assume that $\hat{u}$ is a parametrized function with parameters $\theta$. In the time-dependent case, we consider different $\theta_s$ for each of the functions $\hat{u}(s, x|\theta_s)$ separately. The gradient is given by:

$$
\frac{\partial KL(p^*|\hat{p})}{\partial \theta_s} = \frac{1}{\psi(t, x)} \mathbb{E}_p e^{-S(t,x,u)} \left( \hat{u}(s, X(s)) - u(s, X(s)) - \frac{dW_s}{ds} \right) \frac{\partial \hat{u}(s, X(s))}{\partial \theta_s}
$$

Choosing $u = \hat{u}$ yields the gradient procedure

$$
\theta_{s,n+1} = \theta_{s,n} - \eta \frac{\partial KL(p^*|\hat{p})}{\partial \theta_{s,n}} \bigg|_{u=\hat{u}_n} = \theta_{s,n} + \eta \left\langle \frac{dW_s \partial \hat{u}(s, X(s))}{ds} \frac{\partial \hat{u}(s, X(s))}{\partial \theta_{s,n}} \right\rangle
$$

with

$$
\langle F \rangle = \frac{1}{\psi(t, x)} \mathbb{E}_p e^{-S(t,x,u)} F
$$

and $\eta > 0$ a small parameter. Convergence is guaranteed. We refer to this gradient method as PICE.
Path integral stochastic optimal control

compute_control(model)

  Initialize control \( u_0 \)

  for \( s = 0, \ldots \) do
    \[ data_s = \text{generate data}(model, u_s) \] % infinite data
    \[ u_{s+1} = \text{learn control}(data_s, u_s) \] % infinitely complex
  end for

- compute_control implements an (adaptive) Monte Carlo importance sampler that simulates future trajectories weighted by their exponentiated reward.

- Thm: better \( u_s \) (in terms of control cost) are better importance samplers (in terms of effective sample size)

- PICE: Path integral cross entropy method learns arbitrary accurate state-feedback controller
Controlled noisy Lorenz attractor

\[ u(t, x) = A(t)x + b(t). \quad N = 6000 \]
Control of a deterministic unknown plant

We consider a deterministic control problem of the form

\[ dx_i = f_i(x, t)dt + \sum_a g_{ia}(x, t)u_a dt \]

\[ C = \int_0^T \frac{1}{2} u^T R u + V(x, t) \]

the problem is to compute the optimal control law \( u_a(x, t) \) from a sequence of states that we generate with some chosen control

\[ u_{0:T}^\mu, x_{0:T}^\mu, \quad \mu = 1, \ldots, N \]

Suppose that we choose random controls from a Gaussian distribution: \( u_a dt = \)
\[ d\xi_a, \nu = \lambda R^{-1}. \] The dynamics becomes

\[ dx_i = f_i(x, t)dt + \sum_a g_{ia}(x, t)d\xi_a \]

the uncontrolled dynamics of the stochastic control

\[ dx_i = f_i(x, t)dt + \sum_a g_{ia}(x, t)(u_a dt + d\xi_a) \]

\[ C = \langle \int_0^T \frac{1}{2} u^T Ru + V(x, t) \rangle \]

which is equivalent to the original control problem when \( \lambda \to 0. \)
Acrobot

$q_1(0) = q_2(0) = -\pi/2$, $\dot{q}_1(0) = \dot{q}_2(0) = 0$, maximize final height

$$H = l_1 \sin q_1(T) + l_2 \sin q_2(T)$$
Acrobot

\[d_{11}(q)\ddot{q}_1 + d_{12}(q)\ddot{q}_2 + h_1(q, \dot{q}) + \phi_1(q) = 0\]

\[d_{21}(q)\ddot{q}_1 + d_{22}(q)\ddot{q}_2 + h_2(q, \dot{q}) + \phi_2(q) = u\]

We can write these equations in standard form

\[dx_i = f_i(x)dt + g_i(x)udt\]

with \(x_1 = q_1, x_2 = q_2, x_3 = \dot{q}_1, x_4 = \dot{q}_2\) and

\[f_1(x) = x_3\]
\[f_2(x) = x_4\]
\[f_3(x) = \frac{-d_{22}(h_1+\phi_1)+d_{12}(h_2+\phi_2)}{D}\]
\[f_4(x) = \frac{d_{12}(h_1+\phi_1)-d_{11}(h_2+\phi_2)}{D}\]

\[g_1(x) = 0\]
\[g_2(x) = 0\]
\[g_3(x) = -\frac{d_{12}}{D}\]
\[g_4(x) = \frac{d_{11}}{D}\]
100 iterations. At each iteration 50 stochastic trajectories were generated. The new control was computed from a deterministic trajectory. Noise was lowered at each iteration. Top left: final height for each stochastic trajectory for each iteration (red) and for each deterministic solution (blue).
Acrobot

Result after 100 trials
Model predictive control: UAVs

≈ 100,000 trajectories/second for 1 second of 1 quadrotor simulation. Compute importance sampler adaptively.

Kappen et al. 2015
The CE method for PI control: Time-independent solution

We consider \( \hat{u}(X_s) \) independent of time parametrised by \( \theta \). The gradient of the KL divergence involves an integral:

\[
\frac{\partial KL(p^*|\hat{p})}{\partial \theta} = \frac{1}{\psi(t, x)} \mathbb{E}_p e^{-S(t,x,u)} \left( \int_t^T ds (\hat{u}(X(s)) - u(X(s))) - \int_t^T dW(s) \frac{\partial \hat{u}(X(s))}{\partial \theta} \right)
\]

Choosing \( u = \hat{u} \) yields the gradient procedure

\[
\theta_{n+1} = \theta_n - \eta \frac{\partial KL(p^*|\hat{p})}{\partial \theta_n} \bigg|_{u=\hat{u}_n} = \theta_n + \eta \left( \int_t^T dW_s \frac{\partial \hat{u}(X(s))}{\partial \theta_n} \right)
\]
Discussion

PICE presents challenging learning problems, as is evident from the large fluctuations despite the large number of samples for these relatively small problems.

- The weights of the trajectories are proportional to $e^{-S}$ with $S \propto 1/\lambda$ and $\lambda = R\nu$
  - Small $\lambda$ yields small sample size and difficult learning
  - Large $\nu$ requires large controls, requires small $R$.

This problem is due to the log transform that is used to linearize the Bellman equation.

- Small deviations from optimallity may yield large decrease in sample size.
  - Optimal model is infinitely large
  - An infinite model requires infinitely many samples to avoid overfitting.
  - for finite samples there is an optimal finite model
Conclusion

Importance sampling improves sampling efficiency:
- optimal control = optimal sampling
Conclusion

Importance sampling improves sampling efficiency:
- optimal control = optimal sampling

Learning state dependent/feedback control with PICE
- CE provides a criterion for parametrized controllers
- learn from self-generated data
- use $\infty$ data to learn $\infty$ models
- Connecting Control, Inference and Learning
- application in robotics
Conclusion

Importance sampling improves sampling efficiency:
- optimal control = optimal sampling

Learning state dependent/feedback control with PICE
- CE provides a criterion for parametrized controllers
- learn from self-generated data
- use $\infty$ data to learn $\infty$ models
- Connecting Control, Inference and Learning
- application in robotics

Inference:
- reformulate as control problem
- improve estimates through importance sampling controls
KL control theory

$x$ denotes state of the agent and $x_{1:T}$ is a path through state space from time $t = 1$ to $T$.

$q(x_{1:T}|x_0)$ denotes a probability distribution over possible future trajectories given that the agent at time $t = 0$ is state $x_0$, with

$$q(x_{1:T}|x_0) = \prod_{t=0}^{T} q(x_{t+1}|x_t)$$

$q(x_{t+1}|x_t)$ implements the allowed moves.

$R(x_{1:T}) = \sum_{t=1}^{T} R(x_t)$ is the total cost when following path $x_{1:T}$.

The KL control problem is to find the probability distribution $p(x_{1:T}|x_0)$ that minimizes

$$C(p|x_0) = \sum_{x_{1:T}} p(x_{1:T}|x_0) \left( \log \frac{p(x_{1:T}|x_0)}{q(x_{1:T}|x_0)} + R(x_{1:T}) \right) = KL(p||q) + \langle R \rangle_p$$
KL control theory

$p(x_{1:T}|x_0)$ and $q(x_{1:T}|x_0)$ distributions over trajectories.

Given $q$, find $p$ that minimizes

\[ C(p|x_0) = KL(p||q) - \langle R \rangle_p \]

The solution and the optimal control cost are

\[
p(x_{1:T}|x_0) = \frac{1}{Z(x_0)} q(x_{1:T}|x_0) \exp(R(x_{1:T}))
\]

\[
C = - \log Z(x_0)
\]

\[
Z(x_0) = \sum_{x_{1:T}} q(x_{1:T}|x_0) \exp(R(x_{1:T}))
\]

NB: $Z(x_0)$ is an integral over paths.
KL control theory

The optimal control at time $t = 0$ is given by

$$ p(x_1|x_0) = \sum_{x_2:T} p(x_{2:T}|x_0) \propto q(x_1|x_0) \exp(R(x_1)) \beta_1(x_1) $$

with $\beta_t(x)$ the backward messages.

$$ \beta_T(x_T) = 1 $$
$$ \beta_{t-1}(x_{t-1}) = \sum_{x_t} q(x_t|x_{t-1}) \exp(R(x_t)) \beta_t(x_t) $$
Multi Agent cooperative game

Model of cooperation: either hunt a hare alone or a stag together.

<table>
<thead>
<tr>
<th></th>
<th>Stag</th>
<th>Hare</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stag</td>
<td>3, 3</td>
<td>0, 1</td>
</tr>
<tr>
<td>Hare</td>
<td>1, 0</td>
<td>1, 1</td>
</tr>
</tbody>
</table>

We define the KL-stag-hunt game as a multi-agent version where agents move on a grid to hunt stag or hare.
Approximate inference of the KL-stag-hunt problem

$M = 10$ agents, $N = 400$ locations, $10^{26}$ states per time slice

Sequential BP. If converges, converges in less than 500 iterations. Trajectories are marginal beliefs.
Phase transition (?)
Average cost KL control (Todorov 2006)

When $T \to \infty$ and $q$ ergodic the backward message recursion

$$
\beta_{t-1}(x_{t-1}) = \sum_{x_t} H(x_{t-1}, x_t) \beta_t(x_t) \quad H(x, y) = q(y|x) \exp(R(y))
$$

becomes the computation of the Perron-Frobenius eigen pair $(\beta(\cdot), \lambda)$:

$$
H\beta = \lambda \beta \quad H(x, y) = q(y|x) \exp(R(x))
$$

The optimal control satisfies

$$
p(y|x) = q(y|x) \exp(R(x)) \frac{\beta(y)}{\lambda \beta(x)}
$$

$$
C = - \log \lambda
$$

$$
J(x) = - \log \beta(x)
$$
KL-learning [Bierkens, Kappen 2012]

- Goal: find Perron-Frobenius solution $Hz = \lambda z$, with $H = [q(y|x)\exp(-R(x))]$, while stepping through state space according to $q$ and observing incurred cost.

- Algorithm (KL-learning):
  
  $z \leftarrow (1/n, \ldots, 1/n), \lambda > 0, x \leftarrow$ any state
  
  for $m = 1 : M$ do
  
  $y \leftarrow$ independent draw from $q(\cdot|x)$
  $\Delta \leftarrow \exp(-R(x))z(y)/\lambda - z(x)$
  $z(x) \leftarrow z(x) + \gamma\Delta$
  $\lambda \leftarrow \lambda + \gamma\Delta$
  $x \leftarrow y$
  
  end for

- Invariants: $z > 0, \lambda = \|z\|_1$.

Generalization of $z$-learning (Todorov) to $\lambda \neq 1$
Summary and discussion

Control as inference links control to machine learning and statistical physics
- efficient computational methods
Summary and discussion

Control as inference links control to machine learning and statistical physics
- efficient computational methods
- insight in the role of noise: phase transitions (delayed choice and collaboration)
Summary and discussion

Control as inference links control to machine learning and statistical physics
- efficient computational methods
- insight in the role of noise: phase transitions (delayed choice and collaboration)
- favorable comparison with state-of-the-art RL methods in robotics (Theodorou 2010-2012)
Other topics

- Variational approximation, $n$ joint arm (Kappen tutorial 2011)
- Sampling approach to control of robotics arm (van den Broek 2011)
- Coordination of continuous agents using MF and BP (Wiegerinck et al. 2006, van den Broek et al. 2006)
- Risk sensitive path integral control (van den Broek 2010)
- Inference and control (Kappen tutorial 2011)
The variational method

Consider an arm consisting of $n$ joints of length 1. The location of the $i$th joint in the 2d plane is

$$x_i = \sum_{j=1}^{i} \cos \theta_j, \quad y_i = \sum_{j=1}^{i} \sin \theta_j$$

with $i = 1, \ldots, n$. Each of the joint angles is controlled by a variable $u_i$. The dynamics of each joint is

$$d\theta_i = u_i dt + d\xi_i, \quad i = 1, \ldots, n$$

with $d\xi_i$ independent Gaussian noise with $\langle d\xi_i^2 \rangle = \nu dt$. Denote by $\vec{\theta}$ the vector of joint angles, and $\vec{u}$ the vector of controls.
The variational method

The expected cost for the control path $\tilde{u}_{t:T}$ is

\[ C(\tilde{\theta}, t, \tilde{u}_{t:T}) = \left[ \phi(\theta(T)) + \int_t^T \frac{1}{2} \tilde{u}^T(t) \tilde{u}(t) \right] \]

\[ \phi(\tilde{\theta}) = \frac{\alpha}{2}\left((x_n(\tilde{\theta}) - x_{\text{target}})^2 + (y_n(\tilde{\theta}) - y_{\text{target}})^2\right) \]

with $x_{\text{target}}, y_{\text{target}}$ the target coordinates of the end joint.
The variational method

Because $V = 0, f = 0, g = 1$, the solution to uncontrolled dynamics is Gaussian

$$
\psi(\vec{\theta}_0, t) = \int d\vec{\theta} \left( \frac{1}{\sqrt{2\pi\nu(T-t)}} \right)^n \exp \left( - \sum_{i=1}^{n} (\theta_i - \theta_i^0)^2 / 2\nu(T-t) - \phi(\vec{\theta}) / \nu \right)
$$

The control at time $t$ for all components $i$ is computed from Eq. ?? and is given by

$$
u_i = \frac{1}{T-t} \left( \langle \theta_i \rangle - \theta_i^0 \right)
$$

where $\langle \theta_i \rangle$ is the expectation value of $\theta_i$ computed wrt the probability distribution

$$
p(\vec{\theta}) = \frac{1}{\psi(\vec{\theta}_0, t)} \exp \left( - \sum_{i=1}^{n} (\theta_i - \theta_i^0)^2 / 2\nu(T-t) - \phi(\vec{\theta}) / \nu \right)
$$

\[13\]This is not exactly correct because $\theta$ is a periodic variable. One should use the solution to diffusion on a circle instead. We can ignore this as long as $\sqrt{\nu(T-t)}$ is small compared to $2\pi$. 
The variational method

We compute the expectations $\langle \vec{\theta} \rangle$ by introducing a factorized Gaussian variational distribution $q(\vec{\theta}) = \prod_{i=1}^{n} \mathcal{N}((\theta_i|\mu_i, \sigma_i)$. We compute $\mu_i$ and $\sigma_i$ by by minimizing the KL divergence between $q(\vec{\theta})$ and $p(\vec{\theta})$:

$$KL = \int d\theta q(\theta) \log \frac{q(\theta)}{p(\theta)}$$

$$= - \sum_{i=1}^{n} \log \sqrt{2\pi\sigma_i^2} + \log \psi(\vec{\theta}_0, t) + \frac{1}{2\nu(T-t)} \sum_{i=1}^{n} (\sigma_i^2 + (\mu_i - \theta_i^0)^2) + \frac{1}{\nu} \langle \phi(\vec{\theta}) \rangle_q$$

where we omit irrelevant constants. $\langle \phi(\vec{\theta}) \rangle$ can be computed in closed form. Setting the derivative of the KL with respect to $\mu_i$ and $\sigma_i^2$ equal to zero:

$$\mu_i \leftarrow \theta_i^0 + \alpha(T-t) \left( \sin \mu_i e^{-\sigma_i^2/2} (\langle x_n \rangle - x_{\text{target}}) - \cos \mu_i e^{-\sigma_i^2/2} (\langle y_n \rangle - y_{\text{target}}) \right)$$

$$\frac{1}{\sigma_i^2} \leftarrow \frac{1}{\nu} \left( \frac{1}{(T-t)} + \alpha e^{-\sigma_i^2} - \alpha (\langle x_n \rangle - x_{\text{target}}) \cos \mu_i e^{-\sigma_i^2/2} - \alpha (\langle y_n \rangle - y_{\text{target}}) \sin \mu_i e^{-\sigma_i^2/2} \right)$$
The variational method
The variational method

(a) $t = 0.05$

(b) $t = 0.55$

(c) $t = 1.8$

(d) $t = 2.0$
The variational method

Note, that the computation of $\langle \theta_i \rangle$ solves the coordination problem between the
different joints. Once $\langle \theta_i \rangle$ is known, each $\theta_i$ is steered independently to its target value $\langle \theta_i \rangle$ using the control law Eq. 8. The computation of $\langle \theta_i \rangle$ in the variational approximation is very efficient and can be used to control arms with hundreds of joints.
Coordination of agents

$n$ agents with independent dynamics

\[ dx_\alpha = (f_\alpha(x_\alpha, t) + u_\alpha) + d\xi_\alpha, \quad \alpha = 1, \ldots, n \]

should coordinate their actions to minimize a cost at a future time $t = T$:

\[ \phi(y_1, \ldots, y_n) \quad y_\alpha \in \{z_1, \ldots, z_k\} \]

and $\phi = \infty$ elsewhere.
Coordination of agents

Then,

\[
\Psi(x_1, \ldots, x_n, t) = \int dy_1 \ldots dy_n \prod_{\alpha} \rho(y_\alpha, T|x_\alpha, t) \exp(-\phi(y_1, \ldots, y_n)/\nu)
\]
\[
= \sum_{\vec{y}} \exp(-E(\vec{y}|\vec{x}, t)/\nu)
\]

\[
p(\vec{y}) = \frac{1}{Z} \exp(-E(\vec{y}|\vec{x}, t)/\nu)
\]

\[
u_\alpha(\vec{x}, t) = -\partial_{x_\alpha} J = \left\langle \frac{\partial \log \rho(y_\alpha, T|x_\alpha, t)}{\partial x_\alpha} \right\rangle
\]

with \(\vec{x} = (x_1, \ldots, x_n), \vec{y} = (y_1, \ldots, y_n)\).

\(E\) has a graphical model structure if \(\phi\) has.
Pseudo code

Loop:

1. Compute the cost and its log derivative for each agent to move to each target:

   \[ \rho(z_i, T|x_\alpha, t), \quad i = 1, \ldots, k, \quad \alpha = 1, \ldots, n \]

   This path integral can be estimated using MC sampling or variational approximation.

2. Compute \( u_\alpha \) using graphical model inference in \( p(\vec{y}) \) (exact, BP, MF).
A simple 1d example

Intrinsic dynamics $f_\alpha = 0$, $V(x_1, \ldots, x_n) = 0$:

$$p(y_\alpha, T|x_\alpha, t) \propto \exp(-(y_\alpha - x_\alpha)^2 / 2\nu(T - t))$$

End cost $\phi(y_1, \ldots, y_n) = \sum_{j=1}^{k} (n_j(\vec{y}) - n_j)^2$, with $n_j(\vec{y})$ the # of agents that go to target $j$.

Optimal control is for agent $\alpha$ is

$$u_\alpha = \frac{1}{T-t} (\langle y_\alpha \rangle - x_\alpha)$$
A simple 1d example

(i) Agent predicted target $\langle y_\alpha \rangle$

(j) Agent position $x$
A simple 1d example

**Control cost**
greedy control (red)  
MF control (blue)  
BP control (green)

**CPU time**
extact control (black)  
MF control (blue)  
BP control (green)  
greedy control (red)
Nonlinear Coordination

Agents $a = 1, \ldots, n$ in 2D:

\[
\begin{align*}
dx_a(t) &= v_a(t) \cos \varphi_a(t) \, dt \\
\frac{dy_a(t)}{dt} &= v_a(t) \sin \varphi_a(t) \, dt \\
\frac{dv_a(t)}{dt} &= u_a(t) \, dt + d\xi_a(t) \\
\frac{d\varphi_a(t)}{dt} &= \omega_a(t) \, dt + d\zeta_a(t)
\end{align*}
\]

Initial states $O$, $v_a(0) = 0$, $\varphi_a(0) = 0$
Targets $X$, $v_a(T) = 0$, $\varphi_a(T) = 0$

Sample paths specified at
$t_i = t + i \, dt$,
i = 0, \ldots, 6, \, dt = (T - t)/6
Inference methods:

- Junction Tree (· − ·)
- MF (—)

(100 sample paths per agent-target)

JT: exponential in number of agents
   (intractable for # agents > 10)

MF: polynomial in number of agents

CPU time (s) vs. number of agents:

(# agents = # targets)
Risk sensitive control

It is relatively straightforward to generalize the path integral method to optimize a cost of the form

\[ \tilde{C} = \phi(x_T) + \int \frac{1}{2} u^T R u + V(x) \]

\[ C = \frac{1}{\theta} \log \langle \exp(\theta \tilde{C}) \rangle \]

For \( \theta = 0 \) the risk neutral control is recovered. For \( \theta \) small:

\[ C = \langle \tilde{C} \rangle + \frac{\theta}{2} \left( \langle \tilde{C}^2 \rangle - \langle \tilde{C} \rangle^2 \right) + h.o. \]

\( \theta > 0 \) is risk averse, \( \theta < 0 \) is risk seeking.

vd Broek et al. UAI 2010
Risk sensitive control

We illustrate the behavior for the (well known) LQ case. \( V = f = 0, \phi = \alpha/2x^2 \).

The optimal control is given by

\[
    u = \frac{-\alpha x}{R + \alpha(T - t)(1 - \nu R \theta)}
\]

For \( \theta < 0 \) control is weaker
For \( 0 < \theta < 1/R\nu \) control is stronger
In both cases control increases with time.

For \( \theta > 1/R\nu \), control is only well-defined when the denominator is positive:

\[
    \alpha(T - t) < \frac{R}{\nu R \theta - 1}
\]

Control decreases with time. For larger time-to-go, the expected cost is infinite.

vd Broek et al. UAI 2010
Inference and control

As an example of the intricacies of joint inference and control, consider the simple LQ control problem [Florentin, 1962, Kumar, 1983]

\[
\begin{align*}
\frac{dx}{dt} &= \alpha u dt + d\xi \quad (10) \\
C(x_0, \theta_0, u(0 \to T)) &= \left\langle \phi(x(T)) + \int_0^T dtR(x, u, t) \right\rangle \quad (11)
\end{align*}
\]

with \( \alpha \) unobserved and \( x \) observed. Path cost \( R(x, u, t) \) and end cost \( \phi(x) \) and noise variance \( \nu \) are given.

Although \( \alpha \) is unobserved, we have a means to observe \( \alpha \) indirectly through the sequence \( x_t, u_t, t = 0, \ldots \). Each time step we observe \( dx \) and \( u \) and we can thus update our belief about \( \alpha \) using the Bayes formula:

\[
p_{t+dt}(\alpha|dx, u) \propto p(dx|\alpha, u)p_t(\alpha) \quad (12)
\]

\( p(dx|\alpha, u) \) is Normal in \( dx \) with variance \( \nu dt \)

\( p_t(\alpha) \) our belief at time \( t \) about the values of \( \alpha \)
The information that we receive about $\alpha$ increases with $u$, because the $\alpha u dt$ term dominates the $d\xi$ term. However, large $u$ values are more costly and also may drive us away from our target state $x(T)$.

Thus, the optimal control is a balance between optimal inference and minimal control cost.

The solution is to augment the state space with parameters $\theta_t$ (sufficient statistics) that describe $p_t(\alpha) = p(\alpha|\theta_t)$ and $\theta_0$ known, which describes our initial belief in the possible values of $\alpha$. The cost that must be minimized is

$$C(x_0, \theta_0, u(0 \rightarrow T)) = \left< \phi(x(T)) + \int_0^T dt R(x, u, t) \right>$$

(13)

where the average is with respect to the noise $d\xi$ as well as the uncertainty in $\alpha$.

NB: the average over $\alpha$ depends on $\theta_t$ which is not known beforehand.
For simplicity, consider the example that $\alpha$ attains only two values $\alpha = \pm 1$. Then $p_t(\alpha|\theta) = \sigma(\alpha \theta)$, with the sigmoid function $\sigma(x) = \frac{1}{2}(1 + \tanh(x))$. The update equation Eq. 12 implies a dynamics for $\theta$:

$$d\theta = \frac{u}{\nu} dx = \frac{u}{\nu}(\alpha \nu dt + d\xi)$$

With $z_t = (x_t, \theta_t)$ we obtain a standard HJB Eq.

$$-\partial_t J(t, z) dt = \min_u \left( R(t, x, u) dt + \langle dz \rangle_z \partial_z J(z, t) + \frac{1}{2} \langle dz^2 \rangle_z \partial^2_z J(z, t) \right)$$

with boundary condition $J(z, T) = \phi(x)$ (NB independent of $\theta$).

---

14 The rhs of the Bayes rule is

$$p(dx|\alpha, u)p(\alpha|\theta_t) \propto \exp \left( -\frac{(dx - \alpha \nu dt)^2}{2\nu dt} \right) \exp(\alpha \theta_t) \propto \exp \left( \frac{dxu}{\nu} + \alpha \theta_t \right) = \exp \left( \alpha \left( \theta_t + \frac{dxu}{\nu} \right) \right)$$
The result is

$$-\partial_t J = \min_u \left( R(x, u, t) + \tilde{\alpha} u \partial_x J + \frac{u^2 \tilde{\alpha}}{\nu} \partial_\theta J + \frac{1}{2} \nu \partial^2_x J + \frac{1}{2} \frac{u^2}{\nu} \partial^2_\theta J + u \partial_x \partial_\theta J \right)$$

with boundary conditions $J(x, \theta, T) = \phi(x)$.

Thus, the dual control problem (joint inference on $\alpha$ and control problem on $x$) has become an ordinary control problem in $x, \theta$ (Florentin, 1962).

Note that if $R, \phi$ are quadratic and $\alpha$ is known, this is an LQ problem. However, when $\alpha$ is not known, the corresponding dual control problem is not LQ (because of the additional $u$ dependent terms).

---

\[ 15 \] The expectation values appearing in this equation are conditioned on $(x_t, \theta_t)$ and are averages over $p(\alpha|\theta_t)$ and the Gaussian noise. $\langle dx \rangle_{x,\theta} = \tilde{\alpha} u dt$, $\langle d\theta \rangle_{x,\theta} = \frac{\tilde{\alpha} u^2}{\nu} dt$, $\langle dx^2 \rangle_{x,\theta} = \nu dt$, $\langle d\theta^2 \rangle_{x,\theta} = \frac{u^2}{\nu} dt$, $\langle dxd\theta \rangle = u dt$, with $\tilde{\alpha} = \tanh(\theta)$ the expected value of $\alpha$ for a given value $\theta$. 
Probing

Dual control solution with end cost $\phi(x) = x^2$ and path cost $\int_t^f dt' \frac{1}{2} u(t')^2$ and $\nu = 0.5$. Plot shows the deviation of the control from the certain case: $u_t(x, \theta)/u_t(x, \theta = \pm \infty)$ as a function of $\theta$ for different values of $t$ and $x = 2$. The curves with the larger values are for larger times-to-go.

'Probing': $u$ is much larger when $\alpha$ is uncertain ($\theta$ small) then when $\alpha$ is certain $\theta = \pm \infty$. 
Symmetry breaking and non-differentiability of $J$

The observed probing behavior arises as the result of a symmetry breaking in the right hand side of the Bellman equation.
Figure 5: Rhs of the Bellman equation as a function of $u$ and its derivative for $\Theta = 0$. The different curves correspond to different values of $x$. Explorative behavior ($u$, $0$) arises in the no-knowledge state $\Theta = 0$ by proposing non-zero controls. The singularity is absent at $t = T - 2$ and present starting from $t = T - 3$. 
Symmetry breaking and non-differentiability of $J$

As a result of the local minima in the Bellman optimization, the optimal value function is not differentiable.

The optimal cost-to-go is convex in the belief [Sondik, 1971].

Left) $J_t(x, \theta)$ for $t = T - 2, x = -2$ (grey) and $t = T - 2, x = -6$ (black) versus $\theta$ Right) Same as left, but as a function of the belief $p = p(b = 1|\theta)$. 
Further reading

http://www.snn.ru.nl/~bertk/
http://www.snn.ru.nl

References


