Introduction to Statistical Machine Learning

Christfried Webers

Statistical Machine Learning Group
NICTA
and
College of Engineering and Computer Science
The Australian National University

Machine Learning Summer School
MLSS-2010, 27 September - 6 October

(Figures from C. M. Bishop, "Pattern Recognition and Machine Learning" and
T. Hastie, R. Tibshirani, J. Friedman, "The Elements of Statistical Learning")

Outlines

Overview

1. What is Machine Learning?
2. Definition
3. Examples of Machine Learning
4. Related Fields
5. Fundamental Types of Learning
6. Basic Probability Theory
7. Polynomial Curve Fitting

NICTA
The Australian National University

MLSS 2010

Linear Regression

8. Linear Basis Function Models
9. Maximum Likelihood and Least Squares
10. Regularized Least Squares
11. Bayesian Regression
12. Example for Bayesian Regression
13. Predictive Distribution
14. Limitations of Linear Basis Function Models

Linear Classification

Christfried Webers
NICTA
The Australian National University

MLSS 2010

Neural Networks

Christfried Webers
NICTA
The Australian National University

MLSS 2010

27 Kernel Methods

28 Maximum Margin Classifiers

Resources

34 Sampling from the Uniform Distribution

35 Sampling from Standard Distributions

36 Rejection Sampling

37 Importance Sampling

38 Markov Chain Monte Carlo - The Idea

Christfried Webers
NICTA
The Australian National University

MLSS 2010

More Machine Learning

Outlines

Part I

Overview

What is Machine Learning?

Definition

Machine learning is concerned with the design and development of algorithms that allow computers (machines) to improve their performance over time based on data.

- learning from past experience (training data)
- generalisation
- quantify ‘learning’: improve their performance over time
- need to quantify ‘performance’

What is Machine Learning?

Definition

Machine learning is concerned with the design and development of algorithms that allow computers (machines) to improve their performance over time based on data.

- learning from past experience (training data)
- generalisation
- quantify 'learning': improve their performance over time
- need to quantify 'performance'

Definition (Mitchell, 1998)

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Why Machine Learning?

Machine Learning is essential when

- humans are unable to explain their expertise (e.g. speech recognition).
- humans are not around for help (e.g. navigation on Mars, underwater robotics).
- large amount of data with possible hidden relationships and correlations (empirical sciences, e.g. discover unusual astronomical objects).
- environment changes (fast) in time (e.g. mobile phone network).
- solutions need to be adapted to many particular cases (e.g. junk mail).

Example: It is easier to write a program that learns to play checkers or backgammon well by self-play rather than converting the expertise of a master player to a program.

Junk Mail Filtering

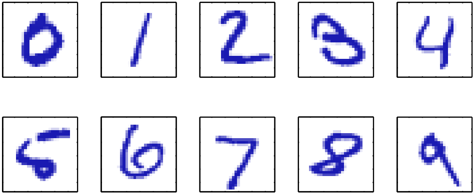
- Given examples of data (mail), and targets {*Junk*,*NoJunk*}.

	From / To	Date Sent	Thread
2949	Support	10/02/09 7:45 +0…	Message from eBay com au
2950	Ken Johnston	10/02/09 14:12 +…	Fool them once, fool them twice, fool…
2951	christfried.web…	10/02/09 3:14 -0…	Assistance, Petersen
2952	Air Sep	9/02/09 4:53 -0800	Un negocio de por vida 1000% Renta…
2953	Osita John	10/02/09 17:33 +…	Now Contact my secretary ask him fo…
2954	Air Sep	9/02/09 0:38 -0800	Un negocio de por vida 1000% Renta…
2955	Air Sep	9/02/09 10:12 -0…	Un negocio de por vida 1000% Renta…
2956	MISS MERCY…	29/01/09 23:13 -…	Urgent Attention(YOUR FILE HAVE…
2957	PEPSI BOTTL…	25/07/08 11:23 -…	OEP00934/UK
2958	JOSEPH POON	11/02/09 12:04 +…	MY PROPOSAL!!!
2959	MADAM ERL…	11/02/09 13:41 +…	LOOKING FOR A TRUSTWORTHY…
2960	REBECA RO…	11/02/09 18:48 +…	Dear sir/madam:
2961	REBECA RO…	11/02/09 18:48 +…	Dear sir/madam:
2962	Elinor Shannon	11/02/09 22:37 +…	I shall look forward to hearing from you
2963	Air Sep	10/02/09 14:37 -…	Un negocio de por vida 1000% Renta…
2964	Foreign Payme…	1/02/09 16:13 +0…	Goodday,
2965	JANET KUEN	12/02/09 16:11 +…	Dear sir/madam:
2966	Abubakar Mar…	10/02/09 19:04 +…	OUR DEAR FRIEND
2967	JAMES ROBE…	12/02/09 23:12 -…	From James Roberts
2968	Bases de Email…	13/02/09 10:50 -…	Nuevas Bases de Datos de Mexico
2969	Barrister Willi…	15/02/09 1:23 +0…	WILL AND TESTAMENT
2970	Isolde	15/02/09 9:45 -0…	A Valentine's Day Ecard Special Deli…
2971	NTI eNews	15/02/09 12:25 -…	Super Sweet Deals From NTIus.com

- Learn to identify new incoming mail as *Junk* or *NoJunk*.
- Continue to learn from the user classifying new mail.

Handwritten Digit Recognition

- Given handwritten ZIP codes on letters, money amounts on cheques etc.



- Learn to correctly recognise new handwritten digits.
- Nonsense input: “Don’t know” preferred to some wrong digit.

Backgammon

- World best computer program TD-GAMMON (Tesauro 1992, 1995) played over a million games against itself.
- Plays now on the level of human world champion.

Image Denoising

Original image

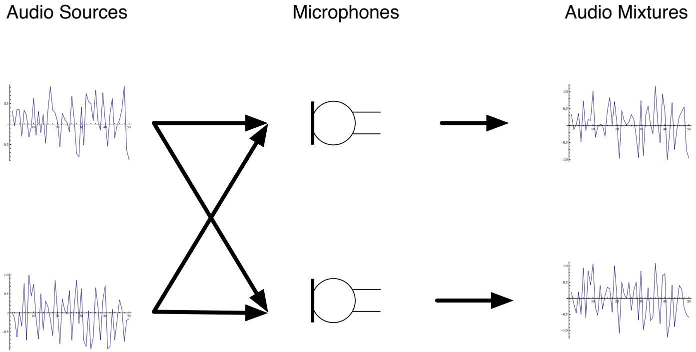
Noise added

Denoised

- McAuley et. al., "Learning High-Order MRF Priors of Color Images", ICML2006

Separating Audio Sources

Cocktail Party Problem (human brains may do it differently ;–)



Other applications of Machine Learning

Christfried Webers
NICTA
The Australian National University

- autonomous robotics,
- detecting credit card fraud,
- detecting network intrusion,
- bioinformatics,
- neuroscience,
- medical diagnosis,
- stock market analysis,
- ...

Related Fields

- Artificial Intelligence - AI
- Statistics
- Game Theory
- Neuroscience, Psychology
- Data Mining
- Computer Science
- Adaptive Control Theory
- ...

Christfried Webers
NICTA
The Australian National University

MLSS 2010

Fundamental Types of Learning

Unsupervised Learning

- Association
- Clustering
- Density Estimation
- Blind source separation

Supervised Learning

- Regression
- Classification

Reinforcement Learning

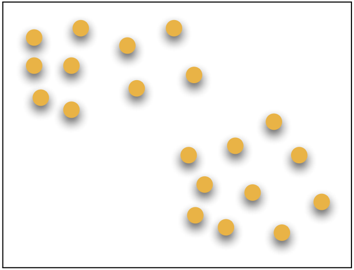
- Agents

Others

- Active Learning
- SemiSupervised Learning
- Transductive Learning
- ...

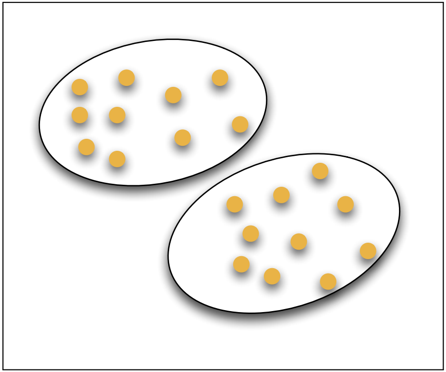
Unsupervised Learning

- Only input data given, no targets (labels).
- Goal: Determine how the data are organised.



Unsupervised Learning - Clustering

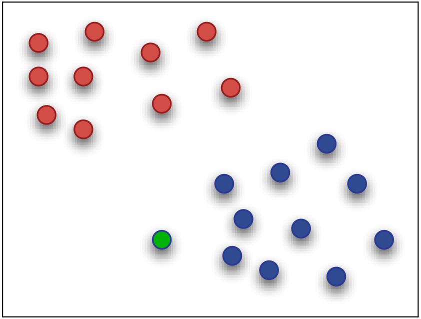
- Clustering : Group similar instances



- Example applications
  - Clustering customers in Customer-Relationship-Management
  - Image compression: color quantisation

Supervised Learning

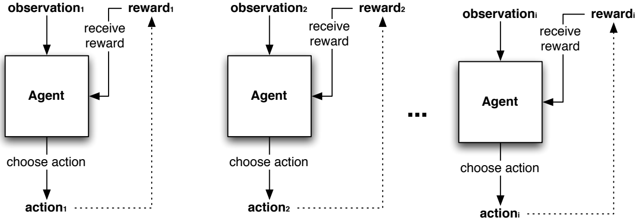
- Given pairs of data and targets (=labels).
- Learn a mapping from the data to the targets (training).
- Goal: Use the learned mapping to correctly predict the target for new input data.
- Need to generalise well from the training data/target pairs.



Reinforcement Learning

- Example: Game playing. There is one reward at the end of the game (negative or positive).
- Find suitable actions in a given environment with the goal of maximising some reward.
- correct input/output pairs never presented
- Reward might only come after many actions.
- Current action may not only influence the current reward, but future rewards too.

Reinforcement Learning



- Exploration versus Exploitation.
- Well suited for problems with a long-term versus short-term reward trade-off.
- Naturally focusing on online performance.

Basic Probability Theory

Probability

is a way of expressing knowledge or belief that an event will occur or has occurred.

Example: Fair Six-Sided Die

Sample space	$\Omega = \{1, 2, 3, 4, 5, 6\}$
Events	$Even = \{2, 4, 6\}$, $Odd = \{1, 3, 5\}$
Probability	$P(3) = \frac{1}{6}$, $P(Odd) = P(Even) = \frac{1}{2}$
Outcome	$3 \in \Omega$
Conditional Probability	$P(3 \mid Odd) = \frac{P(3 \text{ and } Odd)}{P(Odd)} = \frac{1/6}{1/2} = \frac{1}{3}$

General Axioms

- $P(\{\}) = 0 \leq P(A) \leq P(\Omega) = 1$,
- $P(A \cup B) + P(A \cap B) = P(A) + P(B)$,
- $P(A \cap B) = P(A \mid B) P(B)$.

Rules of Probability

- Sum rule: $P(X) = \sum_Y P(X, Y)$
- Product rule: $P(X, Y) = P(X|Y)\, P(Y)$

Probability Jargon

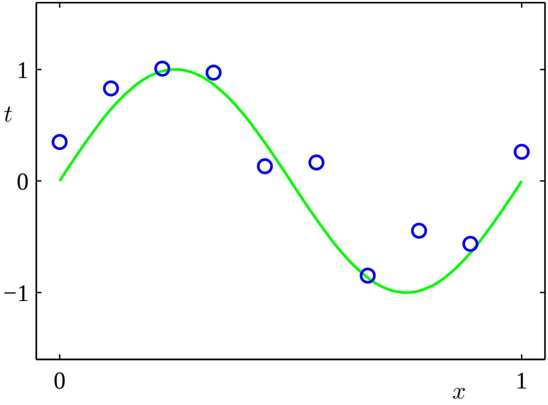
(Un)fair Coin: $\Omega = \{Tail = 0, Head = 1\}$. $P(1) = \theta \in [0, 1]$.

Likelihood $P(1101 \mid \theta) = \theta \times \theta \times (1 - \theta) \times \theta$
Maximum Likelihood (ML) estimate $\hat{\theta} = \arg\max_\theta P(1101 \mid \theta) = \frac{3}{4}$
Prior If we are indifferent, then $P(\theta) = const.$
Evidence $P(1101) = \sum_\theta P(1101 \mid \theta) P(\theta) = \frac{1}{20}$ (actually $\int$)
Posterior $P(\theta \mid 1101) = \frac{P(1101 \mid \theta) P(\theta)}{P(1101)} \propto \theta^3 (1 - \theta)$ (Bayes Rule)
Maximum a Posterior (MAP) estimate $\hat{\theta} = \arg\max_\theta P(\theta \mid 1101) = \frac{3}{4}$
Predictive Distribution $P(1 \mid 1101) = \frac{P(11011)}{P(1101)} = \frac{2}{3}$
Expectation $\mathbb{E}\left[f \mid \dots\right] = \sum_\theta f(\theta) P(\theta \mid \dots)$, e.g. $\mathbb{E}\left[\theta \mid 1101\right] = \frac{2}{3}$
Variance $\mathrm{var}(\theta) = \mathbb{E}\left[(\theta - \mathbb{E}\left[\theta\right])^2 \mid 1101\right] = \frac{2}{63}$
Probability Density $P(\theta) = \frac{1}{\epsilon} P([\theta, \theta + \epsilon])$ for $\epsilon \to 0$

Polynomial Curve Fitting

- some artificial data created from the function

$$\sin(2\pi x) + \text{random noise} \qquad x = 0, \ldots, 1$$



Polynomial Curve Fitting - Training Data

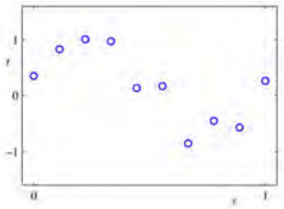
$$N = 10$$

$$\mathbf{x} \equiv (x_1, \dots, x_N)^T$$

$$\mathbf{t} \equiv (t_1, \dots, t_N)^T$$

$$x_i \in \mathbb{R} \quad i = 1, \dots, N$$

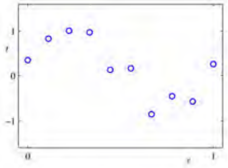
$$t_i \in \mathbb{R} \quad i = 1, \dots, N$$



Polynomial Curve Fitting - Model Specification

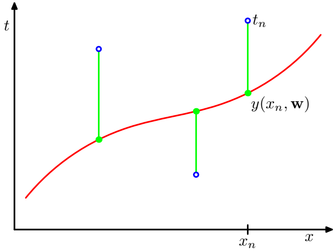
M : order of polynomial

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M$$
$$= \sum_{m=0}^{M} w_m x^m$$



- nonlinear function of x
- *linear* function of the unknown model parameter $\mathbf{w}$
- How can we find good parameters $\mathbf{w} = (w_1, \ldots, w_M)^T$?

Learning is Improving Performance



- Performance measure : Error between target and prediction of the model for the training data

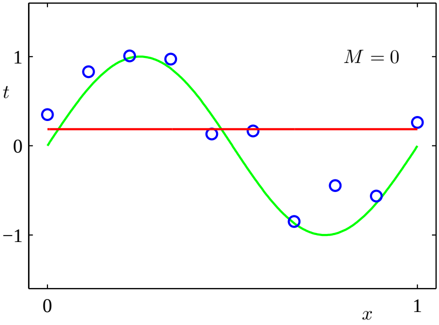
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left(y(x_n, \mathbf{w}) - t_n \right)^2$$

- unique minimum of $E(\mathbf{w})$ for argument $\mathbf{w}^\star$

Model is Constant Function

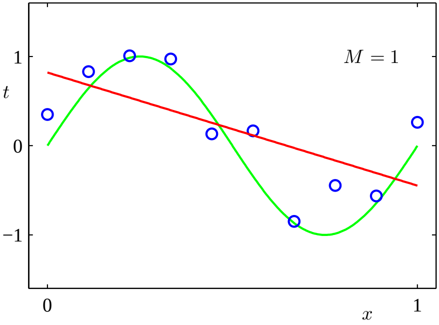
$$y(x, \mathbf{w}) = \sum_{m=0}^{M} w_m \, x^m \quad \Bigg|_{M=0}$$

$$= w_0$$



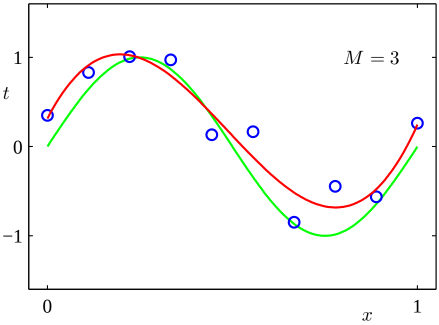
Model is Linear Function

$$y(x, \mathbf{w}) = \sum_{m=0}^{M} w_m x^m \quad \Big|_{M=1}$$
$$= w_0 + w_1 x$$



Model is Cubic Polynomial

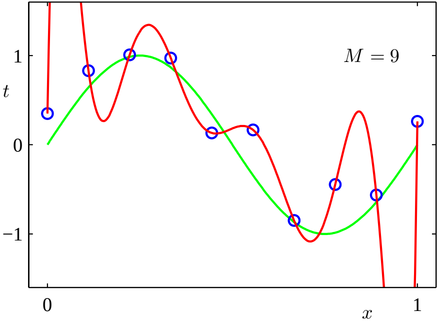
$$y(x, \mathbf{w}) = \sum_{m=0}^{M} w_m x^m \Big|_{M=3}$$
$$= w_0 + w_1 x + w_2 x^2 + w_3 x^3$$



Model is 9^{th} order Polynomial

$$y(x, \mathbf{w}) = \sum_{m=0}^{M} w_m x^m \bigg|_{M=9}$$
$$= w_0 + w_1 x + \cdots + w_8 x^8 + w_9 x^9$$

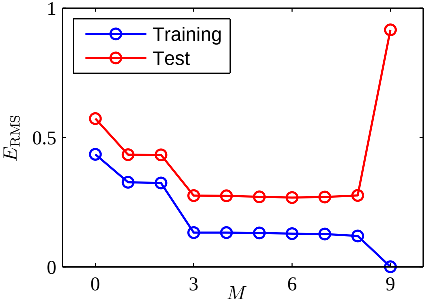
- overfitting



Testing the Fitted Model

- Train the model and get $\mathbf{w}^\star$
- Get 100 new data points
- Root-mean-square (RMS) error

$$E_{\text{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$$



Parameters of the Fitted Model

	M = 0	M = 1	M = 3	M = 9
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

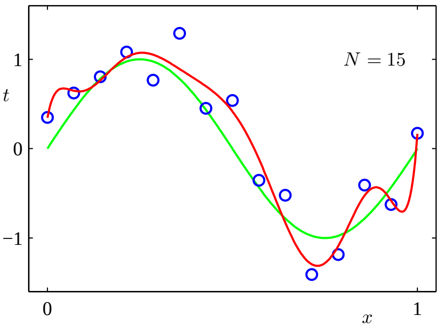
Table: Coefficients $\mathbf{w}^\star$ for polynomials of various order.

Christfried Webers
NICTA
The Australian National University

MLSS 2010

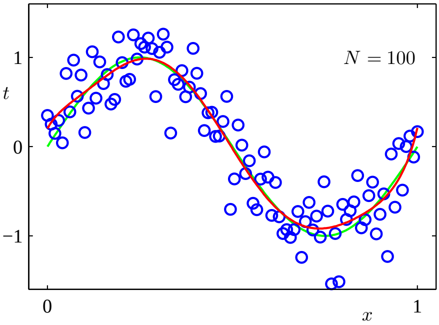
Get More Data

- $N = 15$



Get Even More Data

- $N = 100$
- heuristics : have no less than 5 to 10 times as many data points than parameters
- but number of parameters is not necessarily the most appropriate measure of model complexity !
- later: Bayesian approach



Regularisation

- How to constrain the growing of the coefficients $\mathbf{w}$?
- Add a *regularisation* term to the error function

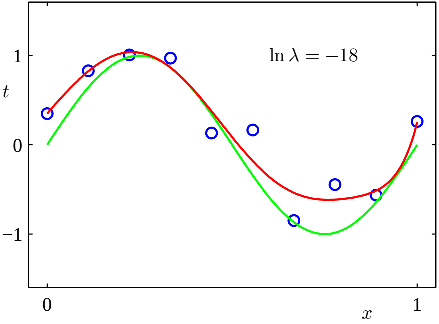
$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(y(x_n,\mathbf{w}) - t_n\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

- Squared norm of the parameter vector $\mathbf{w}$

$$\|\mathbf{w}\|^2 \equiv \mathbf{w}^T\mathbf{w} = w_0^2 + w_1^2 + \cdots + w_M^2$$

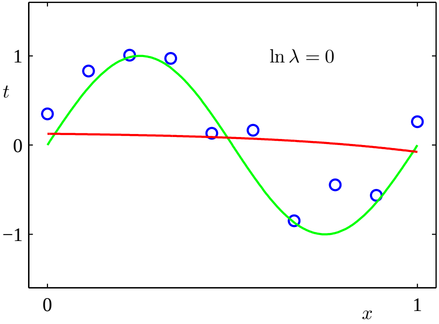
Regularisation

- $M = 9$



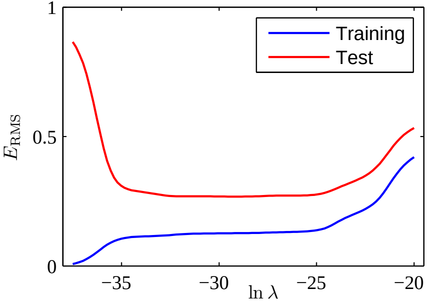
Regularisation

- $M = 9$



Regularisation

- $M = 9$



Part II

Linear Regression

©2010
Christfried Webers
NICTA
The Australian National University

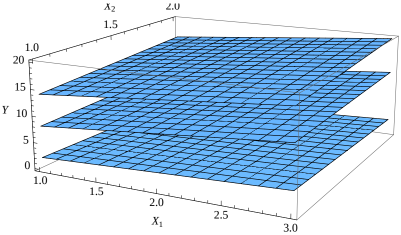
MLSS 2010

Linear Regression Model

- input "feature" vector $\mathbf{x} = (1 \equiv x^{(0)}, x^{(1)}, \dots, x^{(D)})^T \in \mathbb{R}^{D+1}$
- linear regression model

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{D} w_j \, \mathbf{x}^{(j)} = \mathbf{w}^T \mathbf{x}$$

- model parameter $\mathbf{w} = (w_0, \dots, w_D)^T$ where w_0 is the *bias*



Hyperplanes for $\mathbf{w} = \{(2, 1, -1), (5, 2, 1), (10, 2, 2)\}$

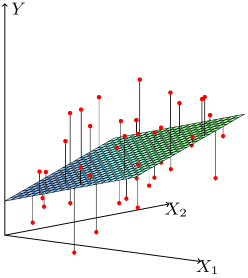
Linear Regression - Finding the Best Model

- Use training data $(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_N, t_N)$
- and loss function (performance measure) to find best $\mathbf{w}$.
- Example : Residual sum of squares

$$Loss(\mathbf{w}) = \sum_{n=1}^{N} (t_n - y(\mathbf{x}_n, \mathbf{w}))^2$$

- Least square regression

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} Loss(\mathbf{w})$$



Linear Basis Function Models

- *Linear* combination of *fixed* nonlinear basis functions $\phi_j(\mathbf{x}) \in \mathbb{R}$

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

- parameter $\mathbf{w} = (w_0, \dots, w_{M-1})^T$,
- w_0 is the *bias parameter*,
- basis functions $\boldsymbol{\phi} = (\phi_0, \dots, \phi_{M-1})^T$
- convention $\phi_0(\mathbf{x}) = 1$

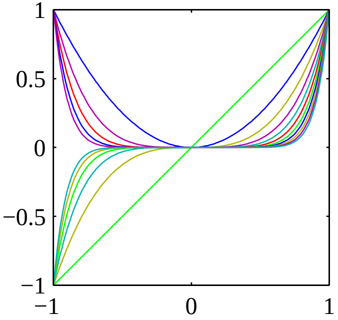
Christfried Webers
NICTA
The Australian National University

Polynomial Basis Functions

- Scalar input variable x

$$\phi_j(x) = x^j$$

- Limitation : Polynomials are global functions of the input variable x.
- Extension: Split the input space into regions and fit a different polynomial to each region (spline functions).

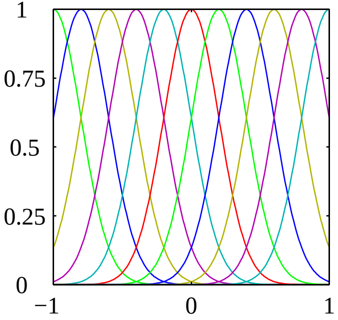


'Gaussian' Basis Functions

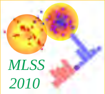
- Scalar input variable x

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

- Not a probability distribution.
- No normalisation required, taken care of by the model parameters w.



Sigmoidal Basis Functions



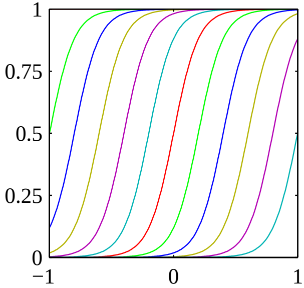
- Scalar input variable x

$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

where $\sigma(a)$ is the logistic sigmoid function defined by

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

- $\sigma(a)$ is related to the *hyperbolic tangent* $\tanh(a)$ by $\tanh(a) = 2\sigma(a) - 1$.

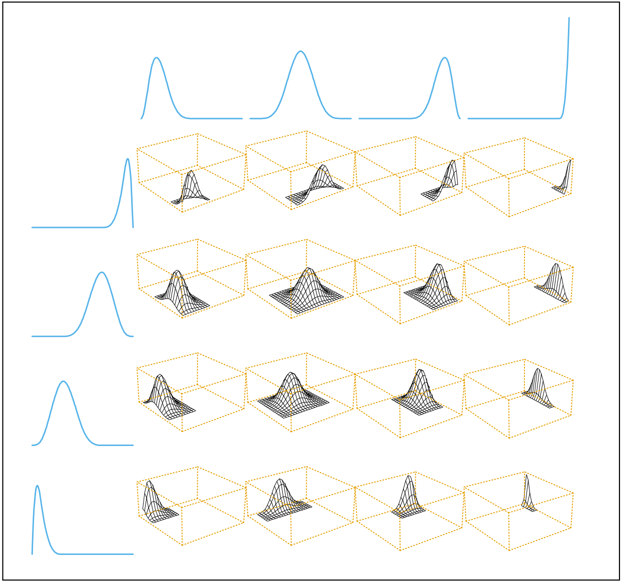


Other Basis Functions - Wavelets

- Wavelets : localised in both space and frequency
- mutually orthogonal to simplify application.

Other Basis Functions - 2D Splines

Splines: polynomials restricted to regions of the input space



Maximum Likelihood and Least Squares

- No special assumption about the basis functions $\phi_j(\mathbf{x})$. In the simplest case, one can think of $\phi_j(\mathbf{x}) = x_j$.
- Assume target t is given by

$$t = \underbrace{y(\mathbf{x}, \mathbf{w})}_{\text{deterministic}} + \underbrace{\epsilon}_{\text{noise}}$$

  where ϵ is a zero-mean Gaussian random variable with precision (inverse variance) β.
- Thus

$$p(t \,|\, \mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t \,|\, y(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

Maximum Likelihood and Least Squares

- Likelihood of one target t given the data $\mathbf{x}$

$$p(t \mid \mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t \mid y(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

- Set of inputs $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with corresponding target values $\mathbf{t} = (t_1, \dots, t_n)^T$.
- Assume data are independent and identically distributed (i.i.d.) (means : data are drawn independent and from the same distribution). The likelihood of the target $\mathbf{t}$ is then

$$\begin{aligned} p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \beta) &= \prod_{n=1}^{N} \mathcal{N}(t_n \mid y(\mathbf{x}_n, \mathbf{w}), \beta^{-1}) \\ &= \prod_{n=1}^{N} \mathcal{N}(t_n \mid \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1}) \end{aligned}$$

Maximum Likelihood and Least Squares

- Consider the logarithm of the likelihood $p(\mathbf{t} \,|\, \mathbf{X}, \mathbf{w}, \beta)$ (the logarithm is a monoton function!)

$$\begin{aligned}\ln p(\mathbf{t} \,|\, \mathbf{X}, \mathbf{w}, \beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n \,|\, \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \\ &= \sum_{n=1}^{N} \ln \left(\sqrt{\frac{\beta}{2\pi}} \exp\left\{ -\frac{\beta}{2} (t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n))^2 \right\} \right) \\ &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w})\end{aligned}$$

where the sum-of-squares error function is

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{t_n - \mathbf{w}^T \boldsymbol{\phi}(x_n)\}^2.$$

- $\arg\max_{\mathbf{w}} \ln p(\mathbf{t} \,|\, \mathbf{X}, \mathbf{w}, \beta) \to \arg\min_{\mathbf{w}} E_D(\mathbf{w})$

Maximum Likelihood and Least Squares

- Rewrite the Error Function

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{t_n - \mathbf{w}^T \phi(x_n)\}^2 = \frac{1}{2} (\mathsf{t} - \mathbf{\Phi w})^T (\mathsf{t} - \mathbf{\Phi w})$$

where $\mathsf{t} = (t_1, \dots, t_N)^T$, and

$$\mathbf{\Phi} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \dots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \dots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \dots & \phi_{M-1}(\mathbf{x}_N) \end{bmatrix}$$

- Maximum likelihood estimate

$$\begin{aligned} \mathbf{w}_{ML} &= \arg\max_{\mathbf{w}} \ln p(\mathsf{t} \mid \mathbf{w}, \beta) = \arg\min_{\mathbf{w}} E_D(\mathbf{w}) \\ &= (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathsf{t} = \mathbf{\Phi}^\dagger \mathsf{t} \end{aligned}$$

where $\mathbf{\Phi}^\dagger$ is the *Moore-Penrose pseudo-inverse* of $\mathbf{\Phi}$.

Regularized Least Squares

Christfried Webers
NICTA
The Australian National University

- Add regularisation in order to prevent overfitting

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

with regularisation coefficient λ.

- Simple quadratic regulariser

$$E_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

- Maximum likelihood solution

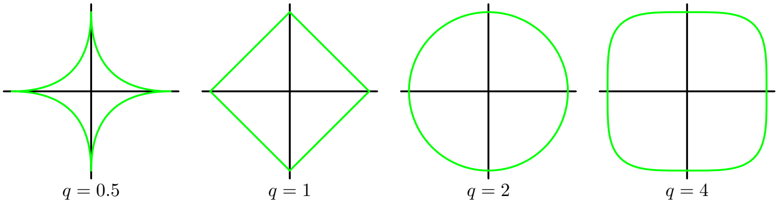
$$\mathbf{w}_{\mathbf{ML}} = \left(\lambda \mathbf{I} + \boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \mathbf{t}$$

Regularized Least Squares

- More general regulariser

$$E_W(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{M}|w_j|^q$$

- $q = 1$ (*lasso*) leads to a sparse model if λ large enough.



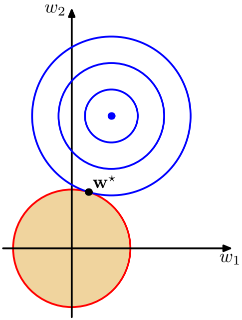
Comparison of Quadratic and Lasso Regulariser

MLSS 2010

Assume a sufficiently large regularisation coefficient λ.

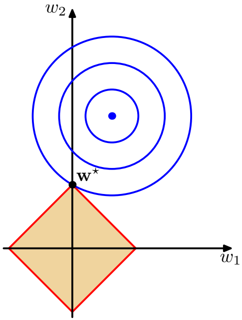
Quadratic regulariser

$$\frac{1}{2}\sum_{j=1}^{M} w_j^2$$



Lasso regulariser

$$\frac{1}{2}\sum_{j=1}^{M} |w_j|$$



Bayesian Regression

- Bayes Theorem

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{normalisation}} \qquad p(\mathbf{w} \mid \mathbf{t}) = \frac{p(\mathbf{t} \mid \mathbf{w})\, p(\mathbf{w})}{p(\mathbf{t})}$$

- likelihood for i.i.d. data

$$\begin{aligned} p(\mathbf{t} \mid \mathbf{w}) &= \prod_{n=1}^{N} \mathcal{N}(t_n \mid y(\mathbf{x}_n, \mathbf{w}), \beta^{-1}) \\ &= \prod_{n=1}^{N} \mathcal{N}(t_n \mid \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1}) \\ &= \text{const} \times \exp\{-\beta \frac{1}{2} (\mathbf{t} - \mathbf{\Phi w})^T (\mathbf{t} - \mathbf{\Phi w})\} \end{aligned}$$

where we left out the conditioning on $\mathbf{x}$ (always assumed), and β, which is assumed to be constant.

How to choose a prior?

- Can we find a prior for the given likelihood which
  - makes sense for the problem at hand
  - allows us to find a posterior in a 'nice' form

An answer to the second question:

Definition (Conjugate Prior)

A class of prior probability distributions $p(w)$ is conjugate to a class of likelihood functions $p(x \mid w)$ if the resulting posterior distributions $p(w \mid x)$ are in the same family as $p(w)$.

Examples of Conjugate Prior Distributions

Table: Discrete likelihood distributions

Likelihood	Conjugate Prior
Bernoulli	Beta
Binomial	Beta
Poisson	Gamma
Multinomial	Dirichlet

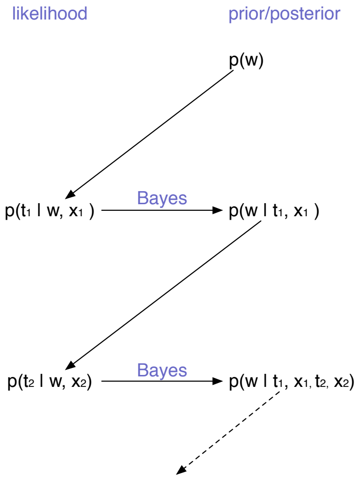
Table: Continuous likelihood distributions

Likelihood	Conjugate Prior
Uniform	Pareto
Exponential	Gamma
Normal	Normal
Multivariate normal	Multivariate normal

NICTA
The Australian National University

Bayesian Regression

- No data point ($N = 0$): start with prior.
- Each posterior acts as the prior for the next data/target pair.
- Nicely fits a sequential learning framework.



Sequential Update of the Posterior

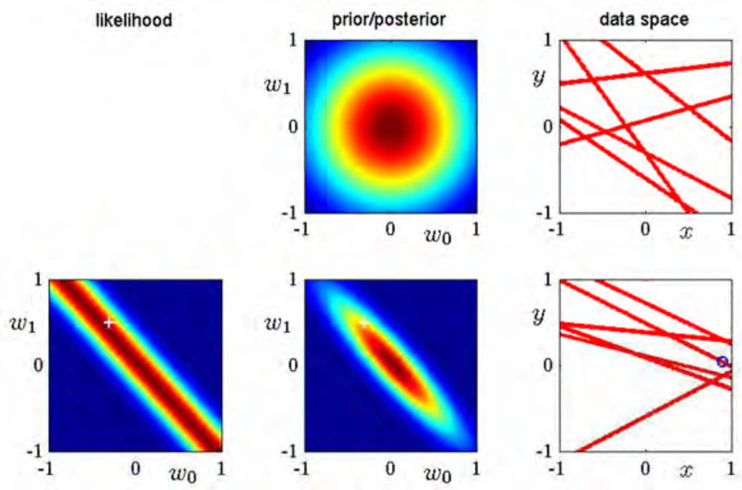
- Example of a linear (basis function) model
- Single input x, single output t
- Linear model $y(x, \mathbf{w}) = w_0 + w_1 x$.
- Data creation
  1. Choose an x_n from the uniform distribution $\mathcal{U}(x \mid -1, 1)$.
  2. Calculate $f(x_n, \mathbf{a}) = a_0 + a_1 x_n$, where $a_0 = -0.3$, $a_1 = 0.5$.
  3. Add Gaussian noise with standard deviation $\sigma = 0.2$,

$$t_n = \mathcal{N}(x_n \mid f(x_n, \mathbf{a}), 0.04)$$

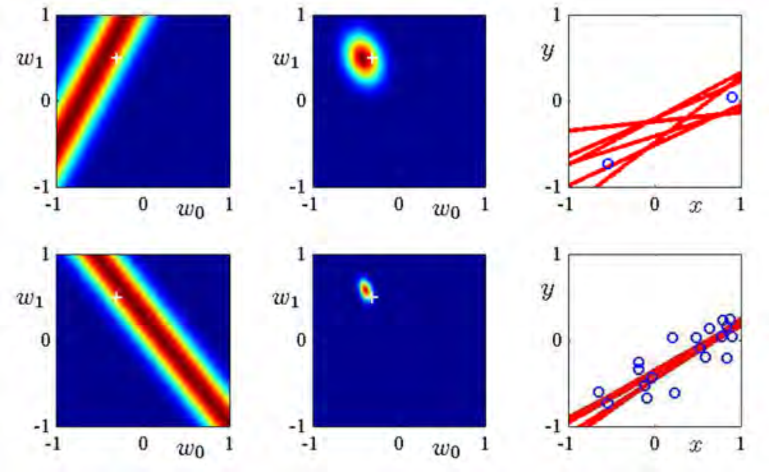
- Set the precision of the uniform prior to $\alpha = 2.0$.

Christfried Webers
NICTA
The Australian National University

Sequential Update of the Posterior



Sequential Update of the Posterior



Predictive Distribution

Definition (The Predictive Distribution)

The Predictive Distribution is the *probability* of the test target t given test data $\mathbf{x}$, the training data set $\mathbf{X}$ and the training targets $\mathbf{t}$.

$$p(t \mid \mathbf{x}, \mathbf{X}, \mathbf{t})$$

- How to calculate the Predictive Distribution?

$$\begin{aligned} p(t \mid \mathbf{x}, \mathbf{X}, \mathbf{t}) &= \int p(t, \mathbf{w} \mid \mathbf{x}, \mathbf{X}, \mathbf{t}) \, \mathrm{d}\mathbf{w} \qquad \text{(sum rule)} \\ &= \int \underbrace{p(t \mid \mathbf{w}, \mathbf{x}, \mathbf{X}, \mathbf{t})}_{\text{testing only}} \underbrace{p(\mathbf{w} \mid \mathbf{x}, \mathbf{X}, \mathbf{t})}_{\text{training only}} \, \mathrm{d}\mathbf{w} \\ &= \int p(t \mid \mathbf{w}, \mathbf{x}) \, p(\mathbf{w} \mid \mathbf{X}, \mathbf{t}) \, \mathrm{d}\mathbf{w} \end{aligned}$$

NICTA
The Australian National University

Predictive Distribution – Isotropic Gaussian Prior

- (Simplified) isotropic Gaussian prior

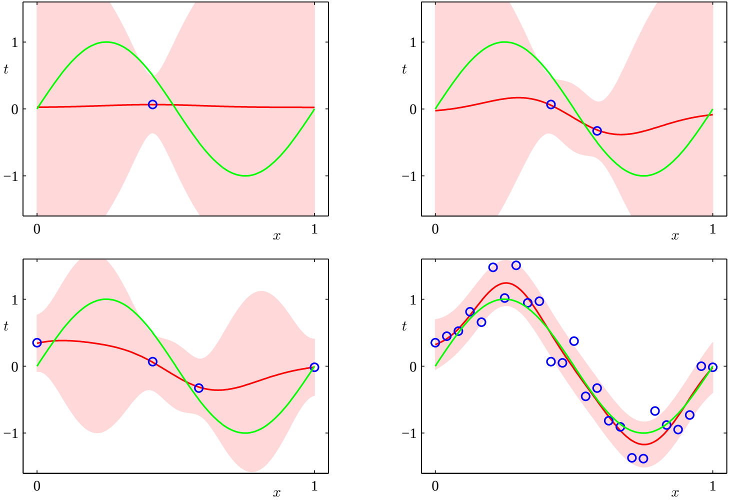
$$p(\mathbf{w} \mid \alpha) = \mathcal{N}(\mathbf{w} \mid \mathbf{0}, \alpha^{-1}\mathbf{I})$$

- Predictive distribution $p(t \mid \mathbf{x}, \mathbf{X}, \mathbf{t})$ is Gaussian, variance after N data points have been seen

$$\sigma_N^2(\mathbf{x}) = \underbrace{\frac{1}{\beta}}_{\text{noise of data}} + \underbrace{\boldsymbol{\phi}^T(\alpha\mathbf{I} + \beta\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\phi}}_{\text{uncertainty of } \mathbf{w}}$$

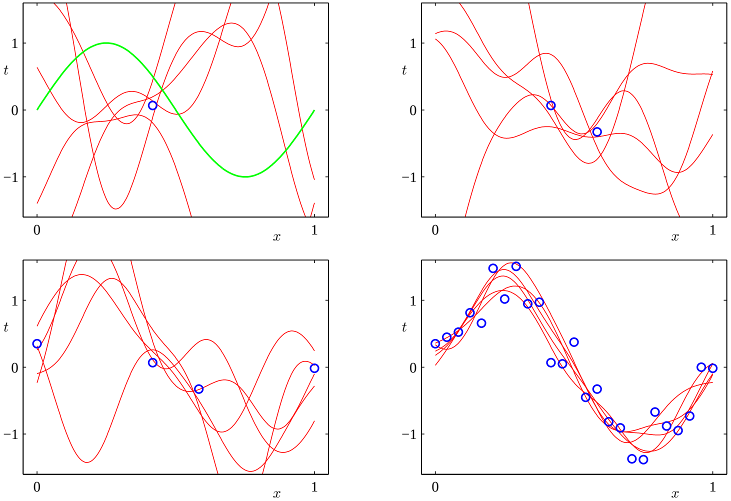
- $\sigma_{N+1}^2(\mathbf{x}) \leq \sigma_N^2(\mathbf{x})$ and $\lim_{N\to\infty} \sigma_N^2(\mathbf{x}) = \frac{1}{\beta}$

Predictive Distribution – Isotropic Gaussian Prior



Example with artificial sinusoidal data from $\sin(2\pi x)$ (green) and added noise. Mean of the predictive distribution (red) and regions of one standard deviation from mean (red shaded).

Samples from the Posterior Distribution



Example with artificial sinusoidal data from $\sin(2\pi x)$ (green) and added noise. Samples $y(x, \mathbf{w})$ (red) from the posterior distribution $p(\mathbf{w} \mid \mathbf{X}, \mathbf{t})$.

Limitations of Linear Basis Function Models

- Basis function $\phi_j(\mathbf{x})$ are fixed before the training data set is observed.
- Curse of dimensionality : Number of basis function grows rapidly, often exponentially, with the dimensionality D.
- But typical data sets have two nice properties which can be exploited if the basis functions are not fixed :
  - Data lie close to a nonlinear manifold with intrinsic dimension much smaller than D. Need algorithms which place basis functions only where data are (e.g. radial basis function networks, support vector machines, relevance vector machines, neural networks).
  - Target variables may only depend on a few significant directions within the data manifold. Need algorithms which can exploit this property (Neural networks).

MLSS 2010

Curse of Dimensionality

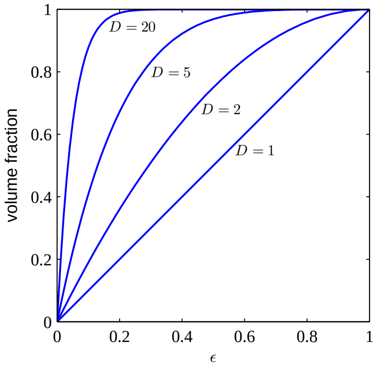
- Linear Algebra allows us to operate in n-dimensional vector spaces using the intution from our 3-dimensional world as a vector space. No surprises as long as n is finite.
- If we add more structure to a vector space (e.g. inner product, metric), our intution gained from the 3-dimensional world around us may be wrong.
- Example: Sphere of radius $r = 1$. What is the fraction of the volume of the sphere in a D-dimensional space which lies between radius $r = 1$ and $r = 1 - \epsilon$?
- Volume scales like r^D, therefore the formula for the volume of a sphere is $V_D(r) = K_D r^D$.

$$\frac{V_D(1) - V_D(1-\epsilon)}{V_D(1)} = 1 - (1-\epsilon)^D$$

Curse of Dimensionality

- Fraction of the volume of the sphere in a D-dimensional space which lies between radius $r = 1$ and $r = 1 - \epsilon$

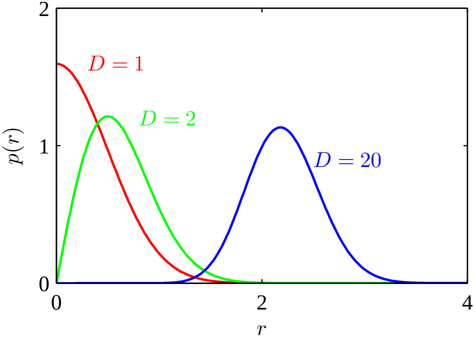
$$\frac{V_D(1) - V_D(1-\epsilon)}{V_D(1)} = 1 - (1-\epsilon)^D$$



Curse of Dimensionality

NICTA
The Australian National University

- Probability density with respect to radius r of a Gaussian distribution for various values of the dimensionality D.



Curse of Dimensionality

- Probability density with respect to radius r of a Gaussian distribution for various values of the dimensionality D.
- Example: $D = 2$; assume $\mu = 0, \Sigma = I$

$$\mathcal{N}(x \,|\, 0, I) = \frac{1}{2\pi} \exp\left\{ -\frac{1}{2} x^T x \right\} = \frac{1}{2\pi} \exp\left\{ -\frac{1}{2} (x_1^2 + x_2^2) \right\}$$

- Coordinate transformation

$$x_1 = r\cos(\phi) \qquad x_2 = r\sin(\phi)$$

- Probability in the new coordinates

$$p(r, \phi \,|\, 0, I) = \mathcal{N}(r(x), \phi(x) \,|\, 0, I) \,|\, J \,|$$

where $|\, J \,| = r$ is the determinant of the Jacobian for the given coordinate transformation.

$$p(r, \phi \,|\, 0, I) = \frac{1}{2\pi} r \exp\left\{ -\frac{1}{2} r^2 \right\}$$

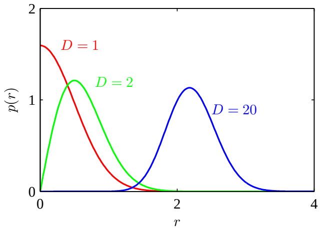
Curse of Dimensionality

- Probability density with respect to radius r of a Gaussian distribution for $D = 2$ (and $\mu = 0, \Sigma = I$)

$$p(r, \phi \,|\, 0, I) = \frac{1}{2\pi} r \exp\left\{-\frac{1}{2} r^2\right\}$$

- Integrate over all angles ϕ

$$p(r \,|\, 0, I) = \int_0^{2\pi} \frac{1}{2\pi} r \exp\left\{-\frac{1}{2} r^2\right\} \, \mathrm{d}\phi = r \exp\left\{-\frac{1}{2} r^2\right\}$$

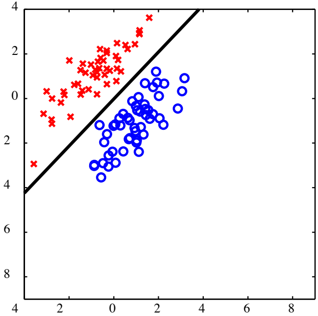


Part III

Linear Classification

Classification

- Goal : Given input data $\mathbf{x}$, assign it to one of K discrete classes $\mathcal{C}_k$ where $k = 1, \dots, K$.
- Divide the input space into different regions.



How to represent binary class labels?

- Class labels are no longer real values as in regression, but a discrete set.
- Two classes : $t \in \{0, 1\}$
  ($t = 1$ represents class $\mathcal{C}_1$ and $t = 0$ represents class $\mathcal{C}_2$)
- Can interpret the value of t as the probability of class $\mathcal{C}_1$, with only two values possible for the probability, 0 or 1.
- Note: Other conventions to map classes into integers possible, check the setup.

How to represent multi-class labels?

- If there are more than two classes ($K > 2$), we call it a multi-class setup.
- Often used: 1-of-K coding scheme in which $\mathbf{t}$ is a vector of length K which has all values 0 except for $t_j = 1$, where j comes from the membership in class C_j to encode.
- Example: Given 5 classes, $\{\mathcal{C}_1, \ldots, \mathcal{C}_5\}$. Membership in class $\mathcal{C}_2$ will be encoded as the target vector

$$\mathbf{t} = (0, 1, 0, 0, 0)^T$$

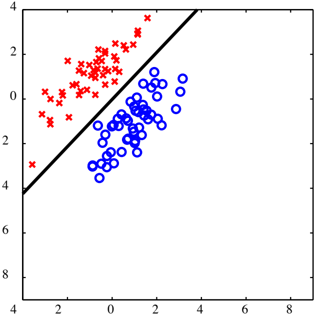
- Note: Other conventions to map multi-classes into integers possible, check the setup.

Linear Model

- Idea: Use again a *Linear Model* as in regression: $y(\mathbf{x}, \mathbf{w})$ is a linear function of the parameters $\mathbf{w}$

$$y(\mathbf{x}_n, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}_n)$$

- But generally $y(\mathbf{x}_n, \mathbf{w}) \in \mathbb{R}$.
  Example: Which class is $y(\mathbf{x}, \mathbf{w}) = 0.71623$?



Generalised Linear Model

- Apply a mapping $f : \mathbb{R} \to \mathbb{Z}$ to the linear model to get the discrete class labels.
- Generalised Linear Model

$$y(\mathbf{x}_n, \mathbf{w}) = f(\mathbf{w}^T \phi(\mathbf{x}_n))$$

- *Activation function*: $f(\cdot)$
- *Link function* : $f^{-1}(\cdot)$

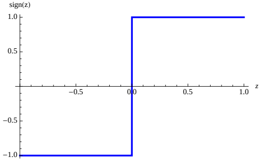


Figure: Example of an activation function $f(z) = \operatorname{sign}(z)$.

Three Models for Decision Problems

In increasing order of complexity

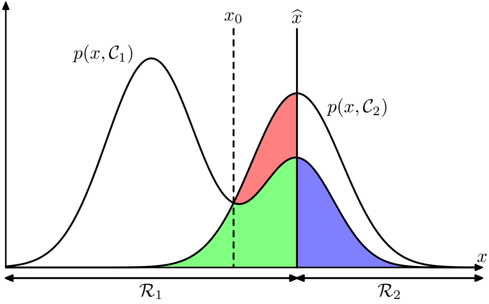
- Find a *discriminant function* $f(\mathbf{x})$ which maps each input directly onto a class label.
- Discriminative Models
  1. Solve the inference problem of determining the posterior class probabilities $p(\mathcal{C}_k \mid \mathbf{x})$.
  2. Use decision theory to assign each new $\mathbf{x}$ to one of the classes.
- Generative Models
  1. Solve the inference problem of determining the class-conditional probabilities $p(\mathbf{x} \mid \mathcal{C}_k)$.
  2. Also, infer the prior class probabilities $p(\mathcal{C}_k)$.
  3. Use Bayes' theorem to find the posterior $p(\mathcal{C}_k \mid \mathbf{x})$.
  4. Alternatively, model the joint distribution $p(\mathbf{x}, \mathcal{C}_k)$ directly.
  5. Use decision theory to assign each new $\mathbf{x}$ to one of the classes.

Decision Theory - Key Ideas

- probability of a mistake

$$p(\text{mistake}) = p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1)$$
$$= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2)\, \mathrm{d}\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1)\, \mathrm{d}\mathbf{x}$$

- goal: minimize $p(\text{mistake})$



Minimising the Expected Loss

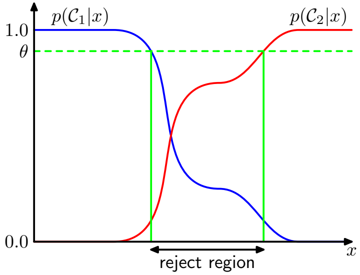
- Not all mistakes are equally costly.
- Weight each misclassification of $\mathbf{x}$ to the wrong class $\mathcal{C}_j$ instead of assigning it to the correct class $\mathcal{C}_k$ by a factor L_{kj}.
- The expected loss is now

$$\mathbb{E}\left[L\right] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj}\, p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}$$

- Goal: minimize the expected loss $\mathbb{E}\left[L\right]$

The Reject Region

- Avoid making automated decisions on difficult cases.
- Difficult cases:
  - posterior probabilities $p(\mathcal{C}_k \mid \mathbf{x})$ are very small
  - joint distributions $p(\mathbf{x}, \mathcal{C}_k)$ have comparable values



NICTA
The Australian National University

Least Squares for Classification

- Regression with a linear function of the model parameters and minimisation of sum-of-squares error function resulted in a closed-from solution for the parameter values.
- Is this also possible for classification?
- Given input data $\mathbf{x}$ belonging to one of K classes $\mathcal{C}_k$.
- Use 1-of-K binary coding scheme.
- Each class is described by its own linear model

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0} \qquad k = 1, \dots, K$$

Least Squares for Classification

- With the conventions

$$\widetilde{\mathbf{w}}_k = \begin{bmatrix} w_{k0} \\ \mathbf{w}_k \end{bmatrix} \qquad \in \mathbb{R}^{D+1}$$

$$\widetilde{\mathbf{x}} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} \qquad \in \mathbb{R}^{D+1}$$

$$\widetilde{\mathbf{W}} = \begin{bmatrix} \widetilde{\mathbf{w}}_1 & \dots & \widetilde{\mathbf{w}}_K \end{bmatrix} \qquad \in \mathbb{R}^{(D+1)\times K}$$

- we get for the discriminant function (vector valued)

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^T \widetilde{\mathbf{x}} \qquad \in \mathbb{R}^K.$$

- For a new input $\mathbf{x}$, the class is then defined by the index of the largest value in the row vector $\mathbf{y}(\mathbf{x})$

Determine $\widetilde{\mathbf{W}}$

- Given a training set $\{\mathbf{x}_n, \mathbf{t}\}$ where $n = 1, \dots, N$, and $\mathbf{t}$ is the class in the 1-of-K coding scheme.
- Define a matrix $\mathbf{T}$ where row n corresponds to $\mathbf{t}_n^T$.
- The sum-of-squares error can now be written as

$$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2} \operatorname{tr}\left\{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^T (\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})\right\}$$

- The minimum of $E_D(\widetilde{\mathbf{W}})$ will be reached for

$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^T \widetilde{\mathbf{X}})^{-1} \widetilde{\mathbf{X}}^T \mathbf{T} = \widetilde{\mathbf{X}}^{\dagger} \mathbf{T}$$

where $\widetilde{\mathbf{X}}^{\dagger}$ is the pseudo-inverse of $\widetilde{\mathbf{X}}$.

Discriminant Function for Multi-Class

- The discriminant function $\mathbf{y}(\mathbf{x})$ is therefore

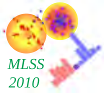
$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^T \widetilde{\mathbf{x}} = \mathbf{T}^T (\widetilde{\mathbf{X}}^\dagger)^T \widetilde{\mathbf{x}},$$

  where $\widetilde{\mathbf{X}}$ is given by the training data, and $\widetilde{\mathbf{x}}$ is the new input.

- Interesting property: If for every $\mathbf{t}_n$ the same linear constraint $\mathbf{a}^T \mathbf{t}_n + b = 0$ holds, then the prediction $\mathbf{y}(\mathbf{x})$ will also obey the same constraint

$$\mathbf{a}^T \mathbf{y}(\mathbf{x}) + b = 0.$$

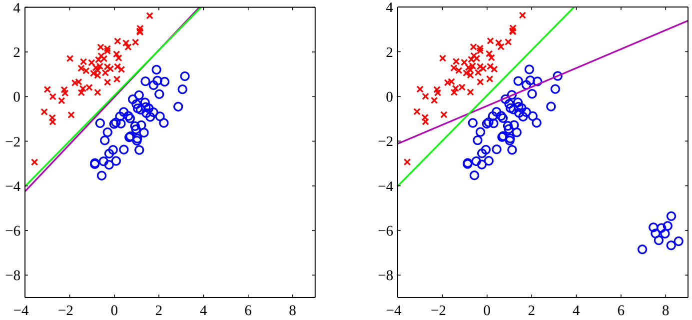
- For the 1-of-K coding scheme, the sum of all components in $\mathbf{t}_n$ is one, and therefore all components of $\mathbf{y}(\mathbf{x})$ will sum to one. BUT: the components are not probabilities, as they are not constraint to the interval $(0, 1)$.



Deficiencies of the Least Squares Approach

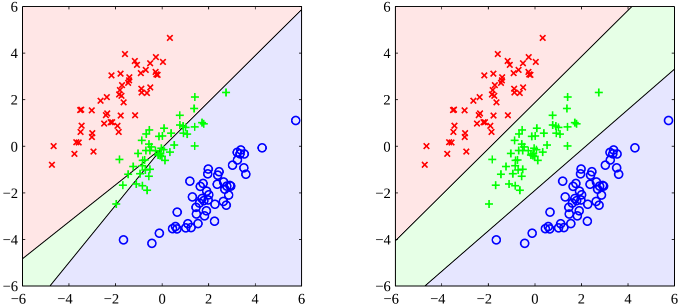
MLSS 2010

Magenta curve : Decision Boundary for the least squares approach (Green curve : Decision boundary for the logistic regression model described later)



Deficiencies of the Least Squares Approach

Magenta curve : Decision Boundary for the least squares approach (Green curve : Decision boundary for the logistic regression model described later)



Fisher's Linear Discriminant

- View linear classification as dimensionality reduction.

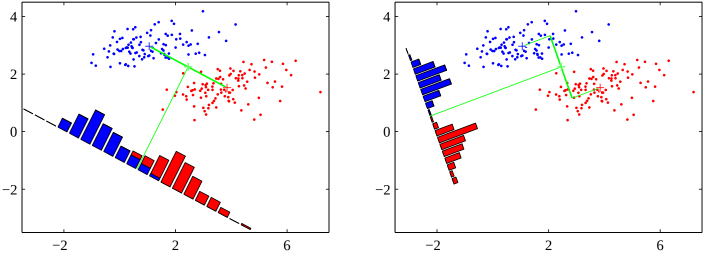
$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

  If $y \geq -w_0$ then class $\mathcal{C}_1$, otherwise $\mathcal{C}_2$.
- But there are many projections from a D-dimensional input space onto one dimension.
- Projection always means loss of information.
- For classification we want to preserve the class separation in one dimension.
- Can we find a projection which maximally preserves the class separation ?

NICTA
The Australian National University

Fisher's Linear Discriminant

Samples from two classes in a two-dimensional input space and their histogram when projected to two different one-dimensional spaces.



Fisher's Linear Discriminant - First Try

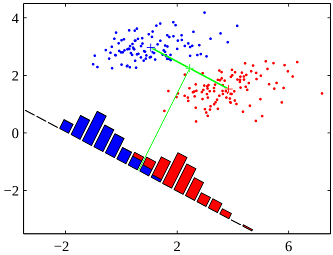
- Given N_1 input data of class $\mathcal{C}_1$, and N_2 input data of class $\mathcal{C}_2$, calculate the centres of the two classes

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n, \qquad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$$

- Choose $\mathbf{w}$ so as to maximise the projection of the class means onto $\mathbf{w}$

$$m_1 - m_2 = \mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2)$$

- Problem with non-uniform covariance



Fisher's Linear Discriminant

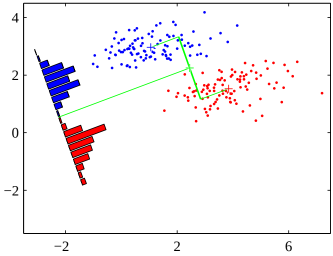
- Measure also the within-class variance for each class

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2$$

  where $y_n = \mathbf{w}^T \mathbf{x}_n$.

- Maximise the Fisher criterion

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$



Fisher's Linear Discriminant

- The Fisher criterion can be rewritten as

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

- $\mathbf{S}_B$ is the between-class covariance

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

- $\mathbf{S}_W$ is the within-class covariance

$$\mathbf{S}_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

Fisher's Linear Discriminant

- The Fisher criterion

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

  has a maximum for Fisher's linear discriminant

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

- Fisher's linear discriminant is NOT a discriminant, but can be used to construct one by choosing a threshold y_0 in the projection space.

The Perceptron Algorithm

- Perceptron ("MARK 1", Cornell Univ., 1960) was the first computer which could learn new skills by trial and error

NICTA
The Australian National University

MLSS 2010

The Perceptron Algorithm

- Frank Rosenblatt (1928 - 1969)
- "Principles of neurodynamics: Perceptrons and the theory of brain mechanisms" (Spartan Books, 1962)

The Perceptron Algorithm

- Two class model
- Create feature vector $\phi(\mathbf{x})$ by a fixed nonlinear transformation of the input $\mathbf{x}$.
- Generalised linear model

$$y(\mathbf{x}) = f(\mathbf{w}^T \phi(\mathbf{x}))$$

with $\phi(\mathbf{x})$ containing some bias element $\phi_0(\mathbf{x}) = 1$.

- nonlinear *activation* function

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

- Target coding for perceptron

$$t = \begin{cases} +1, & \text{if } \mathcal{C}_1 \\ -1, & \text{if } \mathcal{C}_2 \end{cases}$$

Christfried Webers
NICTA
The Australian National University

The Perceptron Algorithm - Error Function

- Idea : Minimise total number of misclassified patterns.
- Problem : As a function of $\mathbf{w}$, this is piecewise constant and therefore the gradient is zero almost everywhere.
- Better idea: Using the $(-1, +1)$ target coding scheme, we want all patterns to satisfy $\mathbf{w}^T \phi(\mathbf{x}_n) t_n > 0$.
- *Perceptron Criterion* : Add the errors for all patterns belonging to the set of misclassified patterns $\mathcal{M}$

$$E_P(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^T \phi(\mathbf{x}_n) t_n$$

Perceptron - Stochastic Gradient Descent

- Perceptron Criterion (with notation $\phi_n = \phi(\mathbf{x}_n)$)

$$E_P(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^T \phi_n t_n$$

- One iteration at step τ
  1. Choose a training pair $(\mathbf{x}_n, t_n)$
  2. Update the weight vector $\mathbf{w}$ by

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \phi_n t_n$$

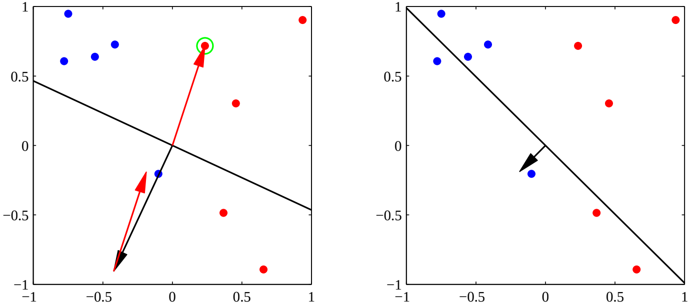
- As $y(\mathbf{x}, \mathbf{w})$ does not depend on the norm of $\mathbf{w}$, one can set $\eta = 1$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \phi_n t_n$$

The Perceptron Algorithm - Update 1

Update of the perceptron weights from a misclassified pattern (green)

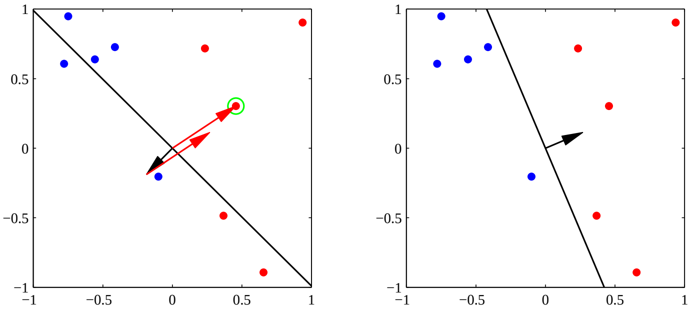
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \phi_n t_n$$



The Perceptron Algorithm - Update 2

Update of the perceptron weights from a misclassified pattern (green)

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \phi_n t_n$$



The Perceptron Algorithm - Convergence

- Does the algorithm converge ?
- For a single update step

$$-\mathbf{w}^{(\tau+1)T}\boldsymbol{\phi}_n t_n = -\mathbf{w}^{(\tau)T}\boldsymbol{\phi}_n t_n - (\boldsymbol{\phi}_n t_n)^T \boldsymbol{\phi}_n t_n < -\mathbf{w}^{(\tau)T}\boldsymbol{\phi}_n t_n$$

  because $(\boldsymbol{\phi}_n t_n)^T \boldsymbol{\phi}_n t_n = \|\boldsymbol{\phi}_n t_n\| > 0$.
- BUT: contributions to the error from the other misclassified patterns might have increased.
- AND: some correctly classified patterns might now be misclassified.
- Perceptron Convergence Theorem : If the training set is linearly separable, the perceptron algorithm is guaranteed to find a solution in a finite number of steps.

Three Models for Decision Problems

In increasing order of complexity

- Find a *discriminant function* $f(\mathbf{x})$ which maps each input directly onto a class label.
- Discriminative Models
  1. Solve the inference problem of determining the posterior class probabilities $p(\mathcal{C}_k \mid \mathbf{x})$.
  2. Use decision theory to assign each new $\mathbf{x}$ to one of the classes.
- Generative Models
  1. Solve the inference problem of determining the class-conditional probabilities $p(\mathbf{x} \mid \mathcal{C}_k)$.
  2. Also, infer the prior class probabilities $p(\mathcal{C}_k)$.
  3. Use Bayes' theorem to find the posterior $p(\mathcal{C}_k \mid \mathbf{x})$.
  4. Alternatively, model the joint distribution $p(\mathbf{x}, \mathcal{C}_k)$ directly.
  5. Use decision theory to assign each new $\mathbf{x}$ to one of the classes.

Probabilistic Generative Models

- Generative approach: model class-conditional densities $p(\mathbf{x} \,|\, \mathcal{C}_k)$ and priors $p(\mathcal{C}_k)$ to calculate the posterior probability for class $\mathcal{C}_1$

$$p(\mathcal{C}_1 \,|\, \mathbf{x}) = \frac{p(\mathbf{x} \,|\, \mathcal{C}_1) p(\mathcal{C}_1)}{p(\mathbf{x} \,|\, \mathcal{C}_1) p(\mathcal{C}_1) + p(\mathbf{x} \,|\, \mathcal{C}_2) p(\mathcal{C}_2)}$$
$$= \frac{1}{1 + \exp(-a(\mathbf{x}))} = \sigma(a(\mathbf{x}))$$

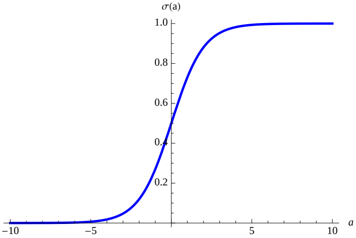
where a and the *logistic sigmoid* function $\sigma(a)$ are given by

$$a(\mathbf{x}) = \ln \frac{p(\mathbf{x} \,|\, \mathcal{C}_1)\, p(\mathcal{C}_1)}{p(\mathbf{x} \,|\, \mathcal{C}_2)\, p(\mathcal{C}_2)} = \ln \frac{p(\mathbf{x}, \mathcal{C}_1)}{p(\mathbf{x}, \mathcal{C}_2)}$$
$$\sigma(a) = \frac{1}{1 + \exp(-a)}.$$

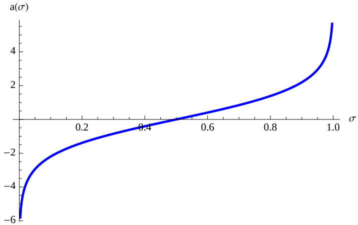
MLSS 2010

Logistic Sigmoid

- The *logistic sigmoid* function $\sigma(a) = \frac{1}{1+\exp(-a)}$
- "squashing function' because it maps the real axis into a finite interval $(0, 1)$
- $\sigma(-a) = 1 - \sigma(a)$
- Derivative $\frac{d}{da}\sigma(a) = \sigma(a)\,\sigma(-a) = \sigma(a)\,(1 - \sigma(a))$
- Inverse is called *logit* function $a(\sigma) = \ln\left(\frac{\sigma}{1-\sigma}\right)$



Logistic Sigmoid $\sigma(a)$



Logit $a(\sigma)$

Probabilistic Generative Models - Multiclass

- The *normalised exponential* is given by

$$p(\mathcal{C}_k \,|\, \mathbf{x}) = \frac{p(\mathbf{x} \,|\, \mathcal{C}_k)\, p(\mathcal{C}_k)}{\sum_j p(\mathbf{x} \,|\, \mathcal{C}_j)\, p(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

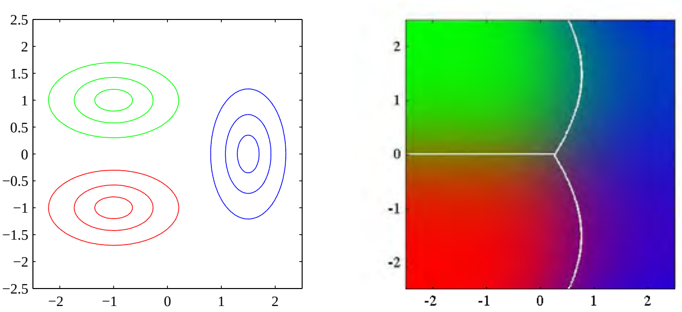
where

$$a_k = \ln(p(\mathbf{x} \,|\, \mathcal{C}_k)\, p(\mathcal{C}_k)).$$

- Also called *softmax function* as it is a smoothed version of the $\max$ function.
- Example: If $a_k \gg a_j$ for all $j \neq k$, then $p(\mathcal{C}_k \,|\, \mathbf{x}) \simeq 1$, and $p(\mathcal{C}_j \,|\, \mathbf{x}) \simeq 0$.

General Case - K Classes, Different Covariance

- If each class-conditional probability is Gaussian and has a *different* covariance, the quadratic terms $-\frac{1}{2}\mathbf{x}^T\mathbf{\Sigma}^{-1}\mathbf{x}$ do no longer cancel each other out.
- We get a quadratic discriminant.



Discrete Features - Naive Bayes

- Assume the input space consists of discrete features, in the simplest case $x_i \in \{0, 1\}$.
- For a D-dimensional input space, a general distribution would be represented by a table with 2^D entries.
- Together with the normalisation constraint, this are $2^D - 1$ independent variables.
- Grows exponentially with the number of features.
- The Naive Bayes assumption is that all features conditioned on the class $\mathcal{C}_k$ are independent of each other.

$$p(\mathbf{x} \,|\, \mathcal{C}_k) = \prod_{i=1}^{D} \mu_{k_i}^{x_i} (1 - \mu_{k_i})^{1-x_i}$$

Discrete Features - Naive Bayes

- With the naive Bayes

$$p(\mathbf{x} \mid \mathcal{C}_k) = \prod_{i=1}^{D} \mu_{k_i}^{x_i} (1 - \mu_{k_i})^{1-x_i}$$

- we can then again find the factors a_k in the normalised exponential

$$p(\mathcal{C}_k \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathcal{C}_k) p(\mathcal{C}_k)}{\sum_j p(\mathbf{x} \mid \mathcal{C}_j) p(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

- as a linear function of the x_i

$$a_k(\mathbf{x}) = \sum_{i=1}^{D} \{ x_i \ln \mu_{k_i} + (1 - x_i) \ln(1 - \mu_{k_i}) \} + \ln p(\mathcal{C}_k).$$

MLSS 2010

Three Models for Decision Problems

In increasing order of complexity

- Find a *discriminant function* $f(\mathbf{x})$ which maps each input directly onto a class label.
- Discriminative Models
  1. Solve the inference problem of determining the posterior class probabilities $p(\mathcal{C}_k \mid \mathbf{x})$.
  2. Use decision theory to assign each new $\mathbf{x}$ to one of the classes.
- Generative Models
  1. Solve the inference problem of determining the class-conditional probabilities $p(\mathbf{x} \mid \mathcal{C}_k)$.
  2. Also, infer the prior class probabilities $p(\mathcal{C}_k)$.
  3. Use Bayes' theorem to find the posterior $p(\mathcal{C}_k \mid \mathbf{x})$.
  4. Alternatively, model the joint distribution $p(\mathbf{x}, \mathcal{C}_k)$ directly.
  5. Use decision theory to assign each new $\mathbf{x}$ to one of the classes.

Logistic Regression is Classification

- Two classes where the posterior of class $\mathcal{C}_1$ is a logistic sigmoid $\sigma()$ acting on a linear function of the feature vector ϕ

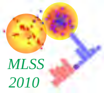
$$p(\mathcal{C}_1 \mid \phi) = y(\phi) = \sigma(\mathbf{w}^T \phi)$$

- $p(\mathcal{C}_2 \mid \phi) = 1 - p(\mathcal{C}_1 \mid \phi)$
- Model dimension is equal to dimension of the feature space M.
- Compare this to fitting two Gaussians

$$\underbrace{2M}_{\text{means}} + \underbrace{M(M+1)/2}_{\text{shared covariance}} = M(M+5)/2$$

- For larger M, the logistic regression model has a clear advantage.

Logistic Regression is Classification



- Determine the parameter via maximum likelihood for data (ϕ_n, t_n), $n = 1, \dots, N$, where $\phi_n = \phi(\mathbf{x}_n)$. The class membership is coded as $t_n \in \{0, 1\}$.
- Likelihood function

$$p(\mathbf{t} \,|\, \mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1 - t_n}$$

  where $y_n = p(\mathcal{C}_1 \,|\, \phi_n)$.
- Error function : negative log likelihood resulting in the *cross-entropy* error function

$$E(\mathbf{w}) = -\ln p(\mathbf{t} \,|\, \mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

Logistic Regression is Classification

- Error function (*cross-entropy* error)

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

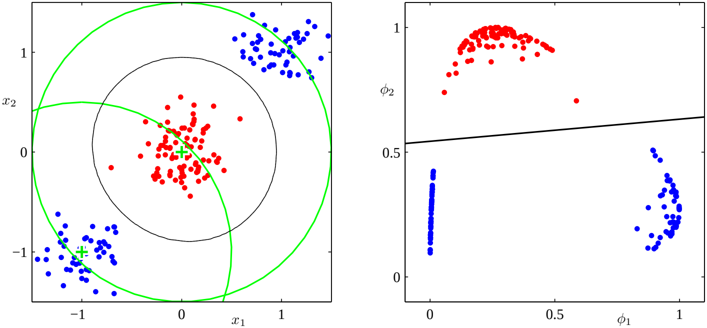
- $y_n = p(\mathcal{C}_1 \mid \phi_n) = \sigma(\mathbf{w}^T \phi_n)$
- Gradient of the error function (using $\frac{d\sigma}{da} = \sigma(1 - \sigma)$)

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\phi_n$$

- gradient does not contain any sigmoid function
- for each data point error is product of deviation $y_n - t_n$ and basis function ϕ_n.
- BUT : maximum likelihood solution can exhibit over-fitting even for many data points; should use regularised error or MAP then.

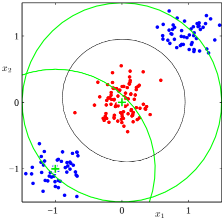
Original Input versus Feature Space

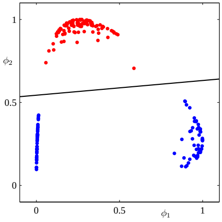
- Used direct input $\mathbf{x}$ until now.
- All classification algorithms work also if we first apply a fixed nonlinear transformation of the inputs using a vector of basis functions $\phi(\mathbf{x})$.
- Example: Use two Gaussian basis functions centered at the green crosses in the input space.



Original Input versus Feature Space

- Linear decision boundaries in the feature space correspond to nonlinear decision boundaries in the input space.
- Classes which are NOT linearly separable in the input space can become linearly separable in the feature space.
- BUT: If classes overlap in input space, they will also overlap in feature space.
- Nonlinear features $\phi(\mathbf{x})$ can not remove the overlap; but they may increase it !





Part IV

Neural Networks

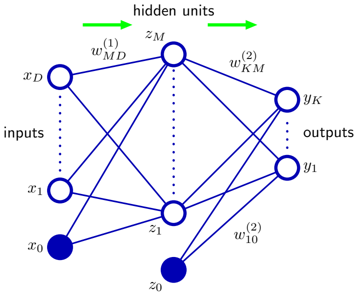
Functional Transformations

- As before, the biases can be absorbed into the weights by introducing an extra input $x_0 = 1$ and a hidden unit $z_0 = 1$.

$$y_k(\mathbf{x}, \mathbf{w}) = g\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right)$$

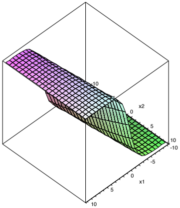
- Compare to Generalised Linear Model

$$y_k(\mathbf{x}, \mathbf{w}) = g\left(\sum_{j=0}^{M} w_{kj}^{(2)} \phi_j(\mathbf{x})\right)$$

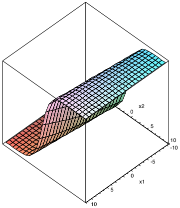


Variable Basis Functions in a Neural Networks

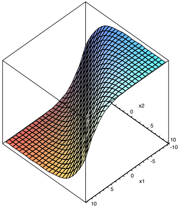
$\phi(\mathbf{x}) = \sigma(w_0 + w_1 x_1 + w_2 x_2)$ for different parameter $\mathbf{w}$.



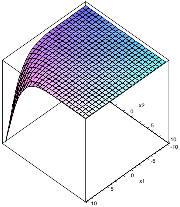
$\mathbf{w} = (0, 1, 0.1)$



$\mathbf{w} = (0, 0.1, 1)$



$\mathbf{w} = (0, -0.5, 0.5)$

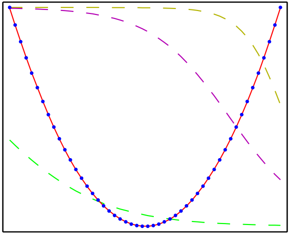


$\mathbf{w} = (10, -0.5, 0.5)$

Aproximation Capabilities of Neural Networks

- Neural network approximating

$$f(x) = x^2$$

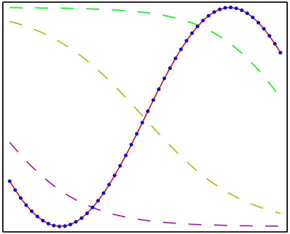


Two-layer network with 3 hidden units ($\tanh$ activation functions) and linear outputs trained on 50 data points sampled from the interval $(-1, 1)$. Red: resulting output. Dashed: Output of the hidden units.

Aproximation Capabilities of Neural Networks

- Neural network approximating

$$f(x) = \sin(x)$$

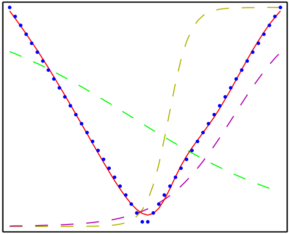


Two-layer network with 3 hidden units ($\tanh$ activation functions) and linear outputs trained on 50 data points sampled from the interval $(-1, 1)$. Red: resulting output. Dashed: Output of the hidden units.

Aproximation Capabilities of Neural Networks

- Neural network approximating

$$f(x) = |x|$$

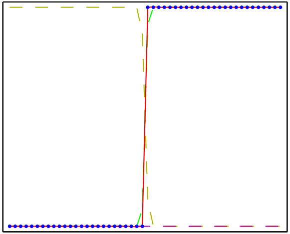


Two-layer network with 3 hidden units (tanh activation functions) and linear outputs trained on 50 data points sampled from the interval $(-1, 1)$. Red: resulting output. Dashed: Output of the hidden units.

Aproximation Capabilities of Neural Networks

- Neural network approximating Heaviside function

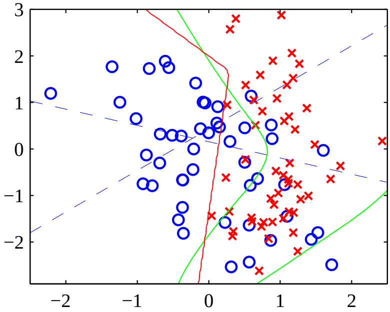
$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



Two-layer network with 3 hidden units ($\tanh$ activation functions) and linear outputs trained on 50 data points sampled from the interval $(-1, 1)$. Red: resulting output. Dashed: Output of the hidden units.

Aproximation Capabilities of Neural Networks

- Neural network for two-class classification.
- 2 inputs, 2 hidden units with $\tanh$ activation function, 1 output with logistic sigmoid activation function.



Red: $y = 0.5$ decision boundary. Dashed blue: $z = 0.5$ hidden unit contours. Green: Optimal decision boundary from the known data distribution.

Parameter Optimisation

- Nonlinear mapping from input $\mathbf{x}_n$ to output $\mathbf{y}(\mathbf{x}_n, \mathbf{w})$.
- Sum-of-squares error function over all training data

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n\|^2,$$

  where we have N pairs of input vectors $\mathbf{x}_n$ and target vectors $\mathbf{t}_n$.
- Find the parameter $\widehat{\mathbf{w}}$ which minimises $E(\mathbf{w})$

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} E(\mathbf{w})$$

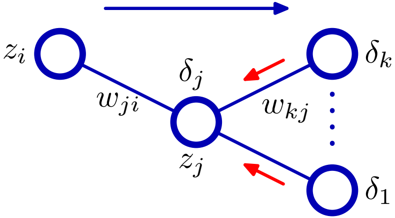
  by gradient descent.

Error Backpropagation

- Given current errors δ_k, the activation function $h(\cdot)$, its derivative $h'(\cdot)$, and its output z_i in the previous layer.
- Error in the previous layer via the *backpropagation formula*

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k.$$

- Components of the gradient ∇E_n are then $\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = \delta_j \, z_i$.



Efficieny of Error Backpropagation

- As the number of weights is usually much larger than the number of units (the network is well connected), the complexity of calculating the gradient $\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}}$ via error backpropagation is of $O(W)$ where W is the number of weights.
- Compare this to *numerical differentiation* using

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = \frac{E_n(w_{ji} + \epsilon) - E_n(w_{ji})}{\epsilon} + O(\epsilon)$$

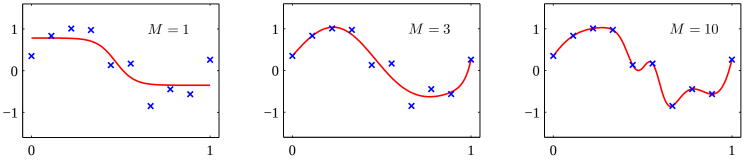
or the numerically more stable (fewer round-off errors) *symmetric differences*

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = \frac{E_n(w_{ji} + \epsilon) - E_n(w_{ji} - \epsilon)}{2\epsilon} + O(\epsilon^2)$$

which both need $O(W^2)$ operations.

Regularisation in Neural Networks

- Model complexity matters again.

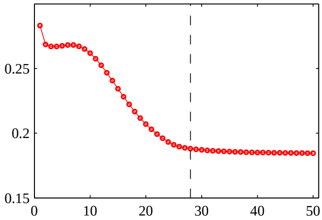


$M = 1$ $M = 3$ $M = 10$

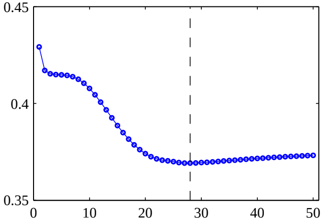
Examples of two-layer networks with M hidden units.

Regularisation via Early Stopping

- Stop training at the minimum of the validation set error.



Training set error.



Validation set error.

Part V

Kernel Methods and SVM

MLSS 2010

Kernel Methods

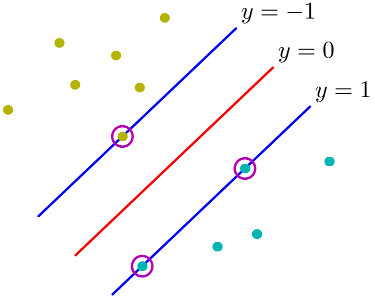
- Keep (some) of the training data and recast prediction as a linear combination of kernel functions which are evaluated at the kept training data points and the new test point.
- Let $L(t, y(x))$ be any loss function
- and $J(f)$ be any penalty quadratic in f,
- then minimum of penalised loss $\sum_{n=1}^{N} L(t_n, y(x_n)) + \lambda J(f)$
- has form $f(x) = \sum_{n=1}^{N} \alpha_n \, k(x_n, x)$
- with α minimising $\sum_{n=1}^{N} L(t_n, (\mathbf{K}\boldsymbol{\alpha})_n) + \lambda \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha}$,
- and Kernel $\mathbf{K}_{ij} = \mathbf{K}_{ji} = k(x_i, x_j)$
- Kernel trick based on Mercer's theorem: Any continuous, symmetric, positive semi-definite kernel function $k(x, y)$ can be expressed as a dot product in a high-dimensional (possibly infinite dimensional) space.

Maximum Margin Classifiers

Support Vector Machines choose the decision boundary which maximises the smallest distance to samples in both classes.

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}:\|\mathbf{w}\|=1}{\arg\max} \min_n \left[t_n (\mathbf{w}^T \phi(\mathbf{x}_n)) \right] \qquad \forall\ t_n \in \{-1, 1\}$$

Linear boundary for $\phi_k(\mathbf{x}) = x^{(k)}$



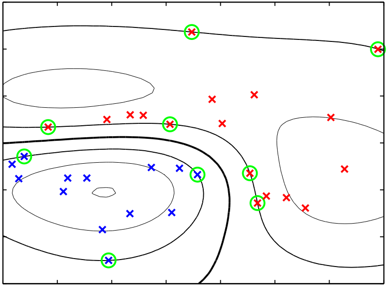
Maximum Margin Classifiers

Non-linear boundary for general $\phi(\mathbf{x})$.

$$\widehat{\mathbf{w}} = \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\mathbf{x_n})$$

for a few $\alpha_n \neq 0$ and corresponding $\mathbf{x_n}$ (support vectors).

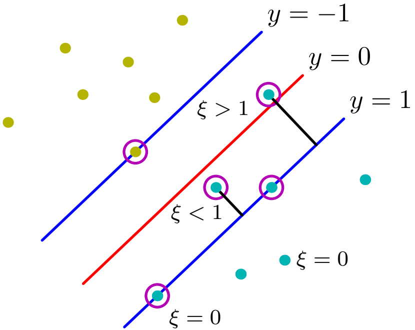
$$\hat{f}(\mathbf{x}) = \widehat{\mathbf{w}}^T \boldsymbol{\phi}(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n \, k(\mathbf{x}_n, \mathbf{x}) \qquad \text{with } k(\mathbf{x}_n, \mathbf{x}) = \boldsymbol{\phi}(\mathbf{x}_n)^T \boldsymbol{\phi}(\mathbf{x})$$



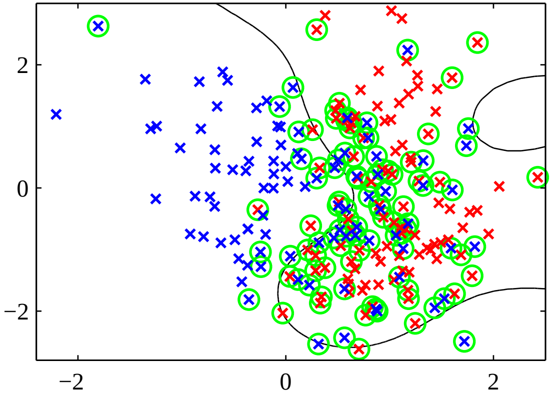
Overlapping Class distributions

- Introduce *slack* variable $\xi_n \geq 0$ for each data point n .

$$\xi_n = \begin{cases} 0, & \text{data point is correctly classified and on margin boundary or beyond} \\ |t_n - y(\mathbf{x})|, & \text{otherwise} \end{cases}$$



Overlapping Class distributions



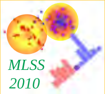
The ν-SVM algorithm using Gaussian kernels $\exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2)$ with $\gamma = 0.45$ applied to a nonseparable data set in two dimensions. Support vectors are indicated by circles.

Part VI

Mixture Models and EM

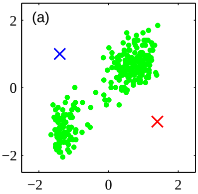
Christfried Webers
NICTA
The Australian National University

MLSS 2010

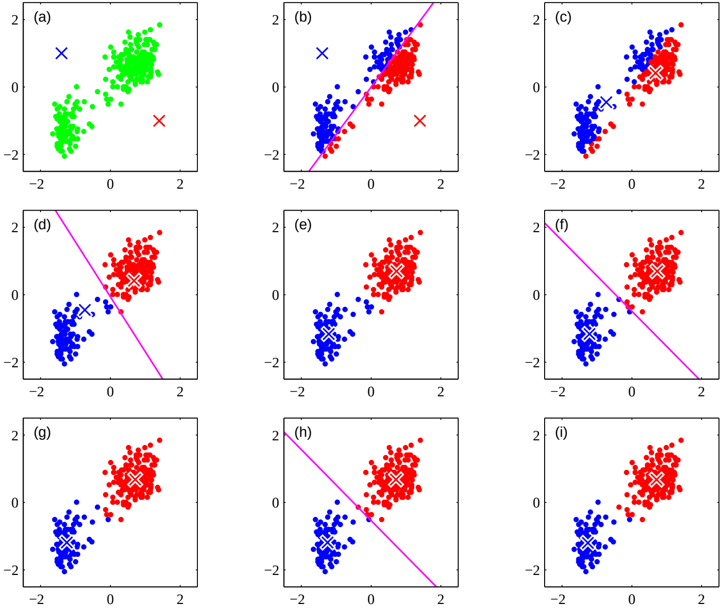


K-means Clustering

- Goal: Partition N features $\mathbf{x}_n$ into K clusters using Euclidian distance $d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|$ such that each feature belongs to the cluster with the nearest mean.
- Distortion measure : $J(\boldsymbol{\mu}, cl(\mathbf{x}_i)) = \sum_{n=1}^{N} d(\mathbf{x}_i, \boldsymbol{\mu}_{cl(\mathbf{x}_i)})^2$ where $cl(\mathbf{x}_i)$ is the index of the cluster centre closest to $\mathbf{x}_i$.
- Start with K arbitrary cluster centres $\boldsymbol{\mu}_k$.
- M-step: Minimise J w.r.t. $cl(\mathbf{x}_i)$: Assign each data point $\mathbf{x}_i$ to closest cluster with index $cl(\mathbf{x}_i)$.
- E-step: Minimise J w.r.t. $\boldsymbol{\mu}_k$: Find new $\boldsymbol{\mu}_k$ as the mean of points belonging to cluster k.
- Iteration over M/E-steps converges to local minimum of J.

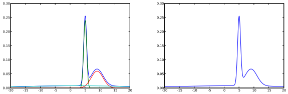


K-means Clustering - Example

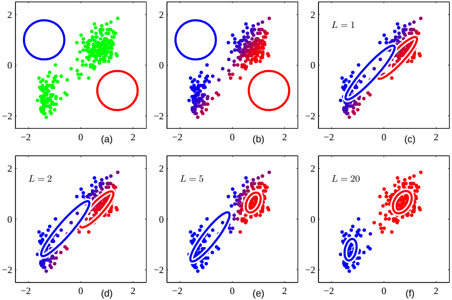


Mixture Models and EM

- Mixture of Gaussians:
  $P(\mathbf{x} \,|\, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Maximise likelihood
  $P(\mathbf{x} \,|\, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ w.r.t. $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$.



- M-step: Minimise J w.r.t. $cl(\mathbf{x}_i)$: Assign each data point $\mathbf{x}_i$ to closest cluster with index $cl(\mathbf{x}_i)$.
- E-step: Minimise J w.r.t. $\boldsymbol{\mu}_k$: Find new $\boldsymbol{\mu}_k$ as the mean of points belonging to cluster k.



EM for Gaussian Mixtures

- Given a Gaussian mixture and data $\mathbf{X}$, maximise the log likelihood w.r.t. the parameters $(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$.
  1. Initialise the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\mu}_k$ and mixing coefficients π_k. Evaluate the log likelihood function.
  2. *E step* : Evaluate the $\gamma(z_k)$ using the current parameters

$$\gamma(z_k) = \frac{\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

  3. *M step* : Re-estimate the parameters using the current $\gamma(z_k)$

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})\, \mathbf{x}_n \qquad \pi_k^{\text{new}} = \frac{N_k}{N}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T$$

  4. Evaluate the log likelihood, if not converged then goto 2.

$$\ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k^{\text{new}} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k^{\text{new}}, \boldsymbol{\Sigma}_k^{\text{new}}) \right\}$$

Mixture of Bernoulli Distributions

- Set of D binary variables x_i, $i = 1, \dots, D$.
- Each governed by a Bernoulli distribution with parameter μ_i. Therefore

$$p(\mathbf{x} \mid \boldsymbol{\mu}) = \prod_{i=1}^{D} \mu_i^{x_i} (1 - \mu_i)^{1 - x_i}$$

- Expecation and covariance

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}$$
$$\operatorname{cov}[\mathbf{x}] = \operatorname{diag}\{\mu_i(1 - \mu_i)\}$$

Mixture of Bernoulli Distributions

- Mixture

$$p(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{k=1}^{K} \pi_k \, p(\mathbf{x} \mid \boldsymbol{\mu}_k)$$

with

$$p(\mathbf{x} \mid \boldsymbol{\mu}_k) = \prod_{i=1}^{D} \mu_{ki}^{x_i} (1 - \mu_{ki})^{1-x_i}$$

- Similar calculation as with mixture of Gaussian

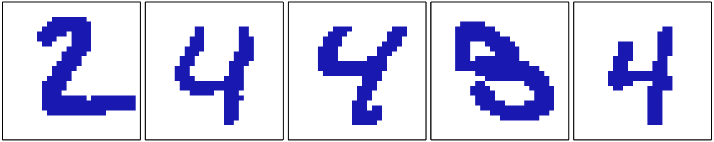
$$\gamma(z_{nk}) = \frac{\pi_k \, p(\mathbf{x}_n \mid \boldsymbol{\mu}_k)}{\sum_{j=1}^{K} \pi_j \, p(\mathbf{x}_n \mid \boldsymbol{\mu}_j)}$$

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

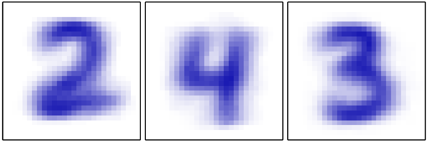
$$\bar{\mathbf{x}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \qquad \mu_k = \bar{\mathbf{x}}$$

$$\pi_k = \frac{N_k}{N}$$

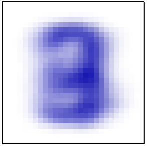
EM for Mixture of Bernoulli Distributions - Digits



Examples from a digits data set, each pixel taken only binary values.



Parameters μ_{ki} for each component in the mixture.



Fit to one multivariate Bernoulli distribution.

The Role of Latent Variables

- EM finds the maximum likelihod solution for models with latent variables.
- Two kinds of variables
  - Observed variables $\mathbf{X}$
  - Latent variables $\mathbf{Z}$

  plus model parameters $\boldsymbol{\theta}$.
- Log likelihood is then

$$\ln p(\mathbf{X} \mid \boldsymbol{\theta}) = \ln \left\{ \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) \right\}$$

- Optimisation problem due to the log-sum.
- Assume maximisation of the distribution $p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})$ over the *complete data set* $\{\mathbf{X}, \mathbf{Z}\}$ is straightforward.
- But we only have the *incomplete data set* $\{\mathbf{X}\}$ and the posterior distribution $p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})$.

EM - Key Idea

- Key idea of EM: As $\mathbf{Z}$ is not observed, work with an 'averaged' version $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ of the complete log-likelihood $\ln p(\mathbf{X}, \mathbf{Z} \,|\, \boldsymbol{\theta})$, averaged over all states of $\mathbf{Z}$.

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} \,|\, \boldsymbol{\theta})$$

EM Algorithm

1. Choose an initial setting for the parameters $\boldsymbol{\theta}^{\text{old}}$.
2. *E step* Evaluate $p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\theta}^{\text{old}})$.
3. *M step* Evaluate $\boldsymbol{\theta}^{\text{new}}$ given by

$$\boldsymbol{\theta}^{\text{new}} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$$

   where

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} \,|\, \boldsymbol{\theta})$$

4. Check for convergence of log likelihood or parameter values. If not yet converged, then

$$\boldsymbol{\theta}^{\text{old}} = \boldsymbol{\theta}^{\text{new}}$$

   and go to step 2.

EM Algorithm - Convergence

- Start with the product rule for the observed variables $\mathbf{x}$, the unobserved variables $\mathbf{Z}$, and the parameters $\boldsymbol{\theta}$

$$\ln p(\mathbf{X}, \mathbf{Z} \,|\, \boldsymbol{\theta}) = \ln p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X} \,|\, \boldsymbol{\theta}).$$

- Apply $\sum_{\mathbf{Z}} q(\mathbf{Z})$ with arbitrary $q(\mathbf{Z})$ to the formula

$$\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X}, \mathbf{Z} \,|\, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X} \,|\, \boldsymbol{\theta}).$$

- Rewrite as

$$\ln p(\mathbf{X} \,|\, \boldsymbol{\theta}) = \underbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{X}, \mathbf{Z} \,|\, \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\mathcal{L}(q, \boldsymbol{\theta})} \underbrace{- \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\mathrm{KL}(q \| p)}$$

- $\mathrm{KL}(q \| p)$ is the *Kullback-Leibler* divergence.

Kullback-Leibler Divergence

- 'Distance' between two distributions $p(y)$ and $q(y)$

$$\mathrm{KL}(q\|p) = \sum_y q(y) \ln \frac{q(y)}{p(y)} \qquad = -\sum_y q(y) \ln \frac{p(y)}{q(y)}$$

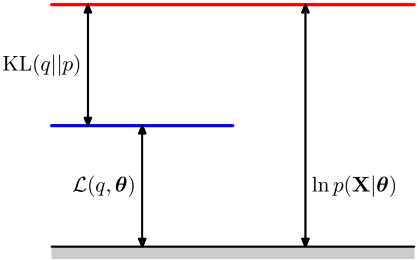
$$\mathrm{KL}(q\|p) = \int q(y) \ln \frac{q(y)}{p(y)} \,\mathrm{d}y \qquad = -\int q(y) \ln \frac{p(y)}{q(y)} \,\mathrm{d}y$$

- $\mathrm{KL}(q\|p) \geq 0$
- not symmetric: $\mathrm{KL}(q\|p) \neq \mathrm{KL}(p\|q)$
- $\mathrm{KL}(q\|p) = 0$ iff $q = p$.
- invariant under parameter transformations

EM Algorithm - Convergence

- The two parts of $\ln p(\mathbf{X} \mid \boldsymbol{\theta})$

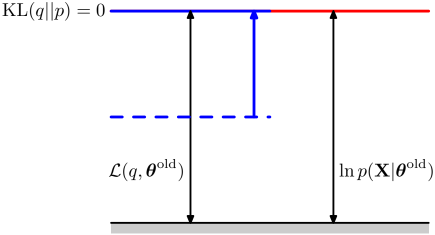
$$\ln p(\mathbf{X} \mid \boldsymbol{\theta}) = \underbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\mathcal{L}(q, \boldsymbol{\theta})} \underbrace{- \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\mathrm{KL}(q \| p)}$$



EM Algorithm - E Step

- Hold $\boldsymbol{\theta}^{\text{old}}$ fixed. Maximise the lower bound $\mathcal{L}(q, \boldsymbol{\theta}^{\text{old}})$ with respect to $q(\cdot)$.
- $\mathcal{L}(q, \boldsymbol{\theta}^{\text{old}})$ is a functional.
- $\ln p(\mathbf{X} \,|\, \boldsymbol{\theta})$ does NOT depend on $q(\cdot)$.
- Maximum for $\mathcal{L}(q, \boldsymbol{\theta}^{\text{old}})$ will occur when the Kullback-Leibler divergence vanishes.
- Therefore, choose $q(\mathbf{Z}) = p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\theta}^{\text{old}})$

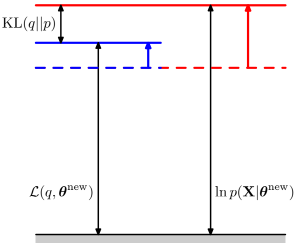
$$\ln p(\mathbf{X} \,|\, \boldsymbol{\theta}) = \underbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{X}, \mathbf{Z} \,|\, \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\mathcal{L}(q, \boldsymbol{\theta})} \underbrace{- \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\mathrm{KL}(q \| p)}$$



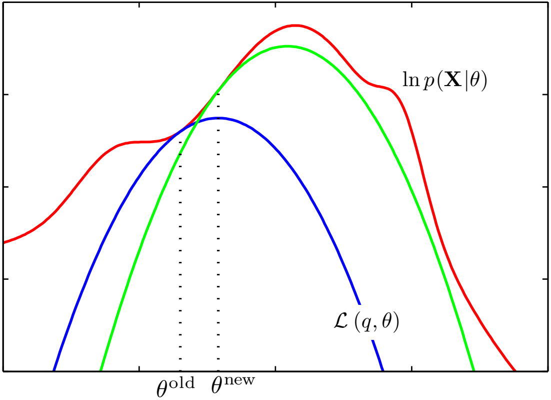
EM Algorithm - M Step

- Hold $q(\cdot) = p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\theta}^{\text{old}})$ fixed. Maximise the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$:
  $\boldsymbol{\theta}^{\text{new}} = \arg\max_{\theta} \mathcal{L}(q, \boldsymbol{\theta}^{\text{old}}) = \arg\max_{\theta} \sum_{\mathbf{Z}} q(\cdot) \ln p(\mathbf{X}, \mathbf{Z} \,|\, \boldsymbol{\theta})$
- $\mathcal{L}(q, \boldsymbol{\theta}^{\text{new}}) > \mathcal{L}(q, \boldsymbol{\theta}^{\text{old}})$ unless maximum already reached.
- As $q(\cdot) = p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\theta}^{\text{old}})$ is fixed, $p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\theta}^{\text{new}})$ will not be equal to $q(\cdot)$, and therefore the Kullback-Leiber distance will be greater than zero (unless converged).

$$\ln p(\mathbf{X} \,|\, \boldsymbol{\theta}) = \underbrace{\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{X}, \mathbf{Z} \,|\, \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\mathcal{L}(q,\boldsymbol{\theta})} \underbrace{- \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{Z} \,|\, \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})}}_{\mathrm{KL}(q\|p)}$$



EM Algorithm - Parameter View



Red curve : incomplete data likelihood.
Blue curve : After E step. Green curve : After M step.

Part VII

Sampling

Sampling from the Uniform Distribution

- In a computer usually via pseudorandom number generator : an algorithm generating a sequence of numbers that approximates the properties of random numbers.
- Example : *linear congruential generators*

$$z^{(n+1)} = (a\,z^{(n)} + c) \quad \mathrm{mod}\ m$$

  for modulus $m > 0$, multiplier $0 < a < m$, increment $0 \leq c < m$, and seed z_0.
- Other classes of pseudorandom number generators:
  - Lagged Fibonacci generators
  - Linear feedback shift registers
  - Generalised feedback shift registers

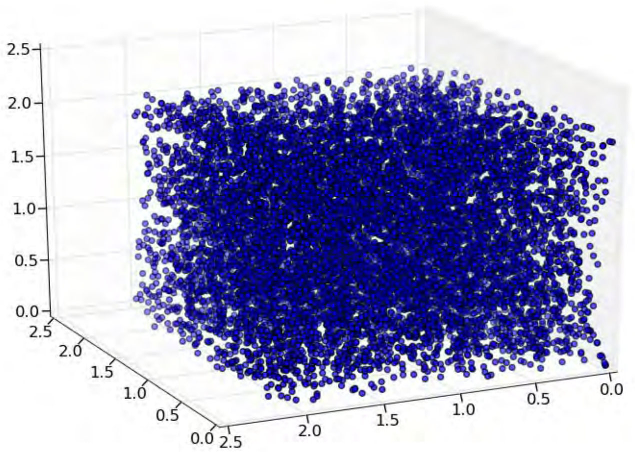
Example: RANDU Random Number Generator

- Used since the 1960s on many machines
- Defined by the recurrence

$$z^{(n+1)} = (2^{16} + 3)\, z^{(n)} \mod 2^{31}$$

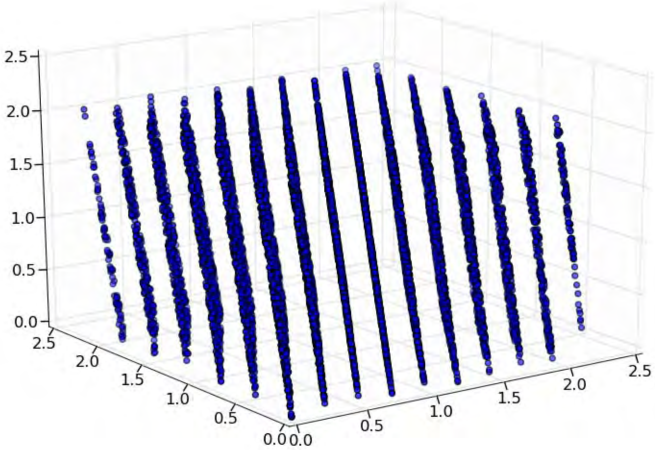
RANDU looks somehow ok?

- Plotting $\left(z^{(n+2)}, z^{(n+1)}, z^{(n)}\right)^T$ in 3D ...



RANDU not really ok

- Plotting $\left(z^{(n+2)}, z^{(n+1)}, z^{(n)}\right)^T$ in 3D ... and changing the viewpoint results in 15 planes.



A Bad Generator - RANDU

- Analyse the recurrence

$$z^{(n+1)} = (2^{16} + 3)\, z^{(n)} \quad \bmod 2^{31}$$

- Assuming every equation to be modulo 2^{31}, we can correlate three samples

$$\begin{aligned} z^{(n+2)} &= (2^{16} + 3)^2\, z^{(n)} \\ &= (2^{32} + 6 \cdot 2^{16} + 9) z^{(n)} \\ &= (6(2^{16} + 3) - 9) z^{(n)} \\ &= 6 z^{(n+1)} - 9 z^{(n)} \end{aligned}$$

- Marsaglia, George "Random Numbers Fall Mainly In The Planes", Proc National Academy of Sciences 61, 25-28, 1968.

Sampling from Standard Distributions

Christfried Webers
NICTA
The Australian National University

- Goal: Sample from $p(y)$ which is given in analytical form.
- Suppose uniformly distributed samples of z in the interval $(0, 1)$ are available.
- Calculate the cumulative distribution function

$$h(y) = \int_{-\infty}^{y} p(x) \, \mathrm{d}x$$

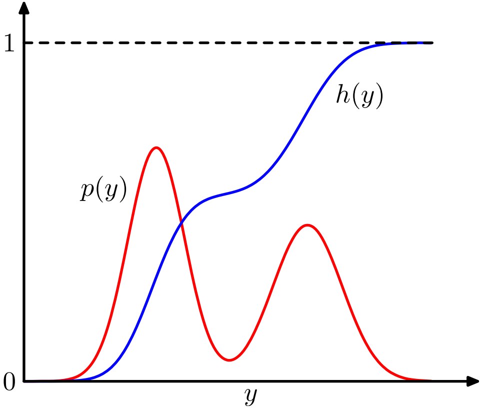
- Transform the samples from $\mathcal{U}(z \,|\, 0, 1)$ by

$$y = h^{-1}(z)$$

  to obtain samples y distributed according to $p(y)$.

Sampling from Standard Distributions

- Goal: Sample from $p(y)$ which is given in analytical form.
- If a uniformly distributed random variable z is transformed using $y = h^{-1}(z)$ then y will be distributed according to $p(y)$.



Sampling from the Exponential Distribution

- Goal: Sample from the exponential distribution

$$p(y) = \begin{cases} \lambda e^{-\lambda y} & 0 \leq y \\ 0 & y < 0 \end{cases}$$

with *rate parameter* $\lambda > 0$.

- Suppose uniformly distributed samples of z in the interval $(0, 1)$ are available.
- Calculate the cumulative distribution function

$$h(y) = \int_{-\infty}^{y} p(x)\,\mathrm{d}x = \int_{0}^{y} \lambda e^{-\lambda y}\,\mathrm{d}x = 1 - e^{-\lambda y}$$

- Transform the samples from $\mathcal{U}(z \,|\, 0, 1)$ by

$$y = h^{-1}(z) = -\frac{1}{\lambda} \ln(1 - z)$$

to obtain samples y distributed according to the exponential distribution.

Sampling the Gaussian Distribution - Box-Muller

1. Generate pairs of uniformly distributed random numbers $z_1, z_2 \in (-1, 1)$ (e.g. $z_i = 2z - 1$ for z from $\mathcal{U}(z \,|\, 0, 1)$)
2. Discard any pair (z_1, z_2) unless $z_1^2 + z_2^2 \le 1$. Results in a uniform distribution inside of the unit circle $p(z_1, z_2) = 1/\pi$.
3. Evaluate $r^2 = z_1^2 + z_2^2$ and

$$y_1 = z_1 \left(\frac{-2 \ln r^2}{r^2} \right)^{1/2}$$

$$y_2 = z_2 \left(\frac{-2 \ln r^2}{r^2} \right)^{1/2}$$

4. y_1 and y_2 are independent with joint distribution

$$p(y_1, y_2) = p(z_1, z_2) \left| \frac{\partial(z_1, z_2)}{\partial(y_1, y_2)} \right| = \frac{1}{\sqrt{2\pi}} e^{-y_1^2/2} \frac{1}{\sqrt{2\pi}} e^{-y_2^2/2}$$

MLSS 2010

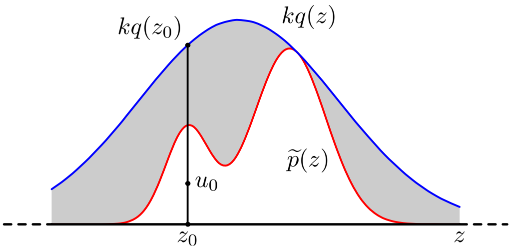
Rejection Sampling

- Assumption 1 : Sampling directly from $p(z)$ is difficult, but we can evaluate $p(z)$ up to some unknown normalisation constant Z_p

$$p(z) = \frac{1}{Z_p} \widetilde{p}(z)$$

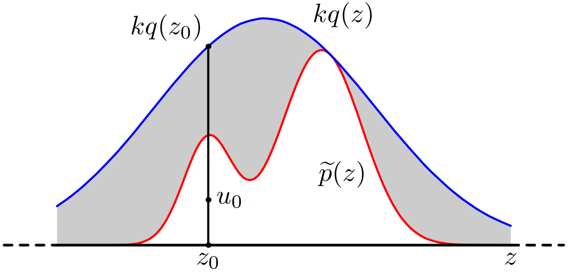
- Assumption 2 : We can draw samples from a simpler distribution $q(z)$ and for some constant k and all z holds

$$kq(z) \geq \widetilde{p}(z)$$



Rejection Sampling

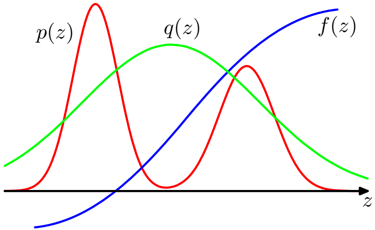
1. Generate a random number z_0 from the distribution $q(z)$.
2. Generate a number from the u_0 from the uniform distribution over $[0, k\,q(z_0)]$.
3. If $u_0 > \widetilde{p}(z_0)$ then reject the pair (z_0, u_0).
4. The remaining pairs have uniform distribution under the curve $\widetilde{p}(z)$.
5. The z values are distributed according to $p(z)$.



Importance Sampling

- Provides a framework to directly calculate the expectation $\mathbb{E}_p\left[f(z)\right]$ with respect to some distribution $p(z)$.
- Does NOT provide $p(z)$.
- Again use a proposal distribution $q(z)$ and draw samples z from it.
- Then

$$\mathbb{E}\left[f\right] = \int f(z)\,p(z)\;\mathrm{d}z = \int f(z)\,\frac{p(z)}{q(z)}q(z)\;\mathrm{d}z \approx \frac{1}{L}\sum_{l=1}^{L}\frac{p(z^{(l)})}{q(z^{(l)})}f(z^{(l)})$$



Importance Sampling - Unnormalised

- Consider both $\widetilde{p}(z)$ and $\widetilde{q}(z)$ to be not normalised.

$$p(z) = \frac{\widetilde{p}(z)}{Z_p} \qquad q(z) = \frac{\widetilde{q}(z)}{Z_q}.$$

- It follows then that

$$\mathbb{E}\left[f\right] \approx \frac{Z_q}{Z_p} \frac{1}{L} \sum_{l=1}^{L} \widetilde{r}_l f(z^{(l)}) \qquad \widetilde{r}_l = \frac{\widetilde{p}(z^{(l)})}{\widetilde{q}(z^{(l)})}.$$

- Use the same set of samples to calculate

$$\frac{Z_p}{Z_q} \approx \frac{1}{L} \sum_{l=1}^{L} \widetilde{r}_l,$$

- resulting in the formula for unnormalised distributions

$$\mathbb{E}\left[f\right] \approx \sum_{l=1}^{L} w_l f(z^{(l)}) \qquad w_l = \frac{\widetilde{r}_l}{\sum_{m=1}^{L} \widetilde{r}_m}$$

Importance Sampling - Key Points

- Try to choose sample points in the input space where the product $f(z)\,p(z)$ is large.
- Or at least where $p(z)$ is large.
- *Importance weights* r_l correct the bias introduced by sampling from the proposal distribution $q(z)$ instead of the wanted distribution $p(z)$.
- Success depends on how well $q(z)$ approximates $p(z)$.
- If $p(z) > 0$ in same region, then $q(z) > 0$ necessary.

NICTA
The Australian National University

Markov Chain Monte Carlo

- Goal : Generate samples from the distribution $p(z)$.
- Idea : Build a machine which uses the current sample to decide which next sample to produce in such a way that the overall distribution of the samples will be $p(z)$.
  1. Current sample $z^{(r)}$ is known. Generate a new sample $z^{\star}$ from a proposal distribution $q(z \mid z^{(r)})$ we know how to sample from.
  2. Accept or reject the new sample according to some appropriate criterion.

$$z^{(l+1)} = \begin{cases} z^{\star} & \text{if accepted} \\ z^{(r)} & \text{if rejected} \end{cases}$$

  3. Proposal distribution depends on the current state.

MLSS 2010

Metropolis Algorithm

1. Choose a symmetric proposal distribution $q(z_A \mid z_B) = q(z_B \mid z_A)$.
2. Accept the new sample $z^\star$ with probability

$$A(z^\star, z^{(r)}) = \min\left(1, \frac{\widetilde{p}(z^\star)}{\widetilde{p}(z^{(r)})}\right)$$

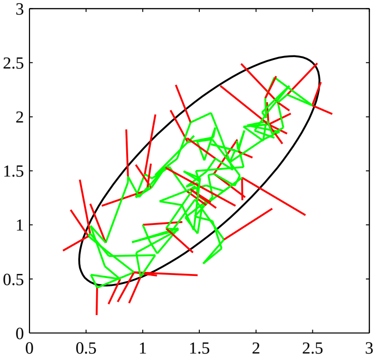
3. How? Choose a random number u with uniform distribution in $(0, 1)$. Accept new sample if $A(z^\star, z^{(r)}) > u$.
4.

$$z^{(l+1)} = \begin{cases} z^\star & \text{if accepted} \\ z^{(r)} & \text{if rejected} \end{cases}$$

Rejection of a point leads to inclusion of the previous sample. (Different from rejection sampling.)

Metropolis Algorithm - Illustration

- Sampling from a Gaussian Distribution (black contour shows one standard deviation).
- Proposal distribution is isotropic Gaussian with standard deviation 0.2.
- 150 candidates generated; 43 rejected.



accepted steps, rejected steps.

Markov Chain Monte Carlo - Metropolis-Hasting

- Generalisation of the Metropolis algorithm for nonsymmetric proposal distributions q_k.
- At step τ, draw a sample $z^\star$ from the distribution $q_k(z \mid z^{(\tau)})$ where k labels the set of possible transitions.
- Accept with probability

$$A_k^\star(z, z^{(\tau)}) = \min\left(1, \frac{\widetilde{p}(z^\star)\, q_k(z^{(\tau)} \mid z^\star)}{\widetilde{p}(z^{(\tau)})\, q_k(z^\star \mid z^{(\tau)})}\right)$$

- Choice of proposal distribution critical.
- Common choice : Gaussian centered on the current state.
    - small variance $\rightarrow$ high acceptance rate, but slow walk through the state space; samples not independent
    - large variance $\rightarrow$ high rejection rate

©2010
Christfried Webers
NICTA
The Australian National University

Part VIII

More Machine Learning

NICTA
The Australian National University

More Machine Learning

- Graphical Models
- Gaussian Processes
- Sequential Data
- Sequential Decision Theory
- Learning Agents
- Reinforcement Learning
- Theoretical Model Selection
- Additive Models and Trees and Related Methods
- Approximate (Variational) Inference
- Boosting
- Concept Learning
- Computational Learning Theory
- Genetic Algorithms
- Learning Sets of Rules
- Analytical Learning
- ...

Part IX

Resources

Journals

- Journal of Machine Learning Research
- Machine Learning
- IEEE Transactions on Pattern Analysis and Machine Intelligence
- IEEE Transactions on Neural Networks
- Neural Computation
- Neural Networks
- Annals of Statistics
- Journal of the American Statistical Association
- SIAM Journal on Applied Mathematics (SIAP)
- . . .

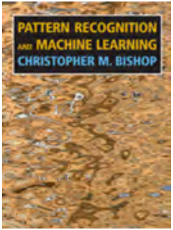
Christfried Webers
NICTA
The Australian National University

Conferences

- International Conference on Machine Learning (ICML)
- European Conference on Machine Learning (ECML)
- Neural Information Processing Systems (NIPS)
- Algorithmic Learning Theory (ALT)
- Computational Learning Theory (COLT)
- Uncertainty in Artificial Intelligence (UAI)
- International Joint Conference on Artificial Intelligence (IJCAI)
- International Conference on Artificial Neural Networks (ICANN)
- ...

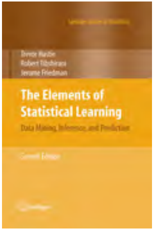
Books

Pattern Recognition and Machine Learning



Christopher M. Bishop

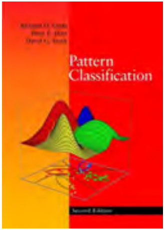
The Elements of Statistical Learning



Trevor Hastie, Robert Tibshirani, Jerome Friedman

Books

Pattern Classification



Richard O. Duda, Peter E. Hart, David G. Stork

Introduction to Machine Learning

Ethem Alpaydin

Datasets

- UCI Repository
  `http://archive.ics.uci.edu/ml/`
- UCI Knowledge Discovery Database Archive
  `http://kdd.ics.uci.edu/summary.data.application.html`
- Statlib
  `http://lib.stat.cmu.edu/`
- Delve
  `http://www.cs.utoronto.ca/~delve/`
- Time Series Database
  `http://robjhyndman.com/TSDL`