Introduction to Machine Learning

Mark Gales

Lent 2012

Machine Learning for Language Processing: Lectures 1 & 2

MPhil in Advanced Computer Science

Decision Making

In this world nothing can be said to be certain, except death and taxes.

- Benjamin Franklin

- We make decisions under uncertainty all the time

  gambling (not recommended), weather forecasting (not very successfully)
  insurance (risk assessment), stock market

Need to formalise "intuitive decisions" mathematically

- Basically, how to quantify and manipulate uncertainty.
- Various tasks we can consider
  - classification: predict class from observations
  - regression (prediction): predict value from observations
  - clustering: group observations into "meaningful" groups

Machine Learning

- One definition is (Mitchell):

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

  alternatively

  "Systems built by analysing data sets rather than by using the intuition of experts"

- Multiple specific conferences:
  - {International,European} Conference on Machine Learning;
  - Neural Information Processing Systems;
  - International Conference on Pattern Recognition etc etc;
- as well as sessions in other conferences:
  - ICASSP - machine learning for signal processing.

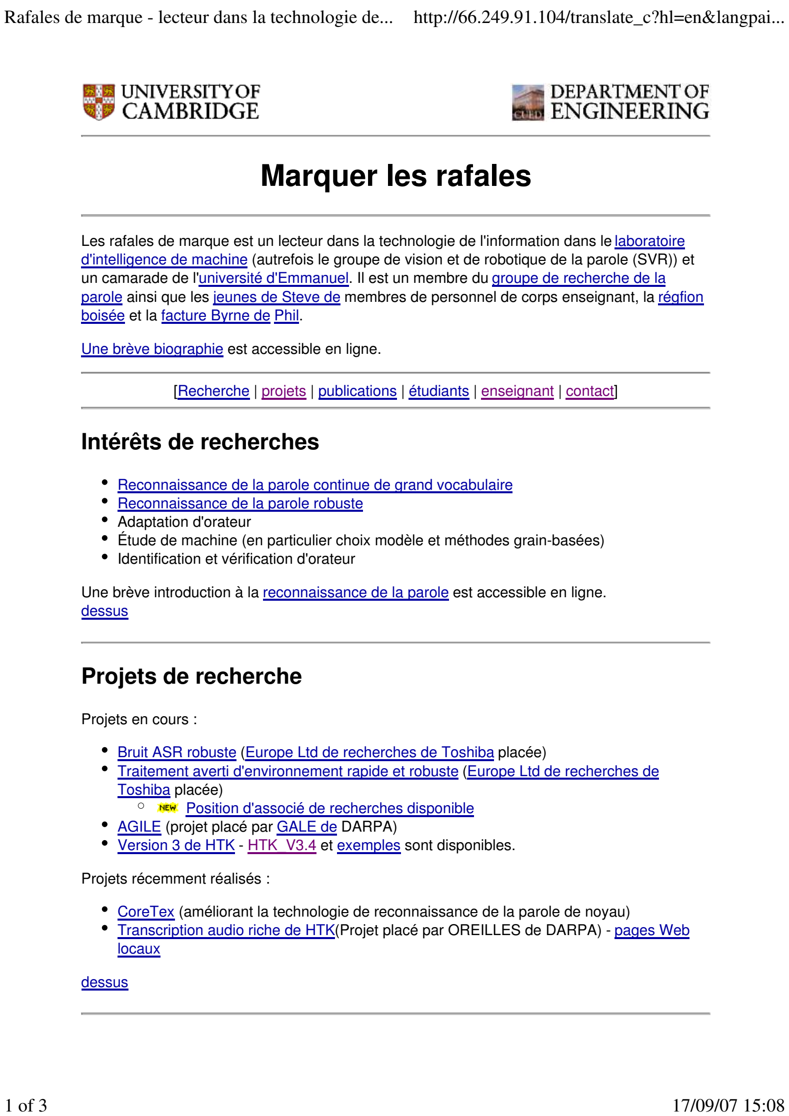
Natural Language Processing Applications

- Natural language processing becoming increasingly popular (important)

- Many possible applications:
  - spam email detection;
  - named-entity recognition;
  - machine translation;
  - relation extraction;
  - information retrieval;
  - sentiment analysis;

- Generally need to structure and annotate vast quantities of text data
  - sometimes used in combination with speech and image processing

Machine Translation



Rafales de marque - lecteur dans la technologie de... http://66.249.91.104/translate_c?hl=en&langpai...

UNIVERSITY OF CAMBRIDGE DEPARTMENT OF ENGINEERING

Marquer les rafales

Les rafales de marque est un lecteur dans la technologie de l'information dans le laboratoire d'intelligence de machine (autrefois le groupe de vision et de robotique de la parole (SVR)) et un camarade de l'université d'Emmanuel. Il est un membre du groupe de recherche de la parole ainsi que les jeunes de Steve de membres de personnel de corps enseignant, la régfion boisée et la facture Byrne de Phil.

Une brève biographie est accessible en ligne.

[Recherche | projets | publications | étudiants | enseignant | contact]

Intérêts de recherches

- Reconnaissance de la parole continue de grand vocabulaire
- Reconnaissance de la parole robuste
- Adaptation d'orateur
- Étude de machine (en particulier choix modèle et méthodes grain-basées)
- Identification et vérification d'orateur

Une brève introduction à la reconnaissance de la parole est accessible en ligne.
dessus

Projets de recherche

Projets en cours :

- Bruit ASR robuste (Europe Ltd de recherches de Toshiba placée)
- Traitement averti d'environnement rapide et robuste (Europe Ltd de recherches de Toshiba placée)
  - NEW Position d'associé de recherches disponible
- AGILE (projet placé par GALE de DARPA)
- Version 3 de HTK - HTK_V3.4 et exemples sont disponibles.

Projets récemment réalisés :

- CoreTex (améliorant la technologie de reconnaissance de la parole de noyau)
- Transcription audio riche de HTK(Projet placé par OREILLES de DARPA) - pages Web locaux

dessus

1 of 3 17/09/07 15:08

- Interesting application
  - statistical approaches work well
- Translation of my web-page (in 2007)
  - Mark Gales becomes To mark the gusts
  - Phil *Woodland*, Steve *Young*, *Bill* Byrne
  - (translates correctly in 2009)
- Part of this MPhil course.

Sentiment Analysis

- Alternative application
  - statistical approaches again work well
- "Twitter Sentiment" for `angry`
  - lots of tweets about `angry birds` game
  - sentiment accuracy reasonable

Natural Language Processing

Why is natural language processing an interesting machine learning task?

- “Standard” machine learning tasks are of the form
  - clearly defined set of observations $\boldsymbol{x}$
  - “reasonable” number of classes $\omega_1, \ldots, \omega_K$
- Consider statistical machine translation with source vocabulary $V_{\rm s}$ target vocabulary $V_{\rm t}$
  - for target sentence of 10 words $V_{\rm t}^{10}$ possible sentences
  - $V_{\rm s}$ word features, $V_{\rm s}^2$ word-pair features, $V_{\rm s}^3$ word-tuple features, ...
  - vast number of possible classes, vast number of possible features
- These initial 2 lectures will not address these problems directly
  - standard machine learning described
  - language processing extensions to will be described in future lectures

Basic Discrete Probability

- Discrete random variable x takes one value from the set, with probabilities

$$\mathcal{X} = \omega_1, \ldots, \omega_K; \quad p_j = \Pr(x = \omega_j), \quad j = 1, \ldots, K$$

Probability mass function, $P(x)$, describes the set of probabilities, satisfies

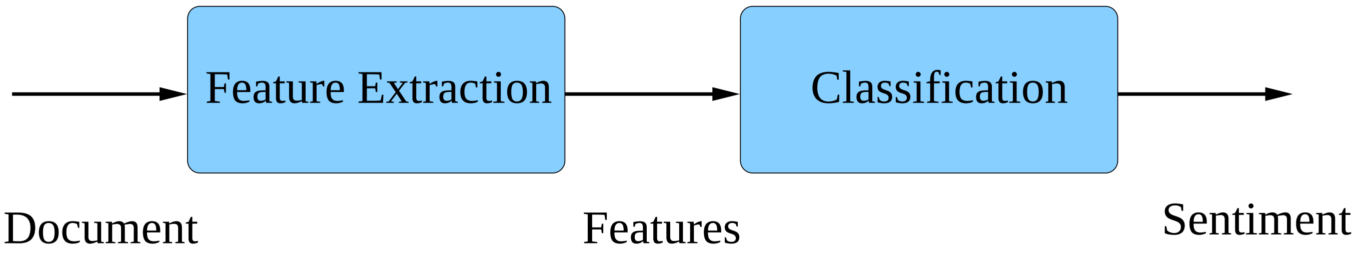
$$\sum_{x \in \mathcal{X}} P(x) = 1, \quad P(x) \geq 0$$

Probability density function, $p(x)$, equivalent for continuous random variables

- For random variables x, y, z need

conditional distribution: $P(x|y) = \frac{P(x,y)}{P(y)}$
joint distribution $P(x, y)$
marginal distribution $P(x) = \sum_{y \in \mathcal{Y}} P(x, y)$
chain rule $P(x, y, z) = P(x|y, z)\, P(y|z)\, P(z)$

Machine Learning Framework



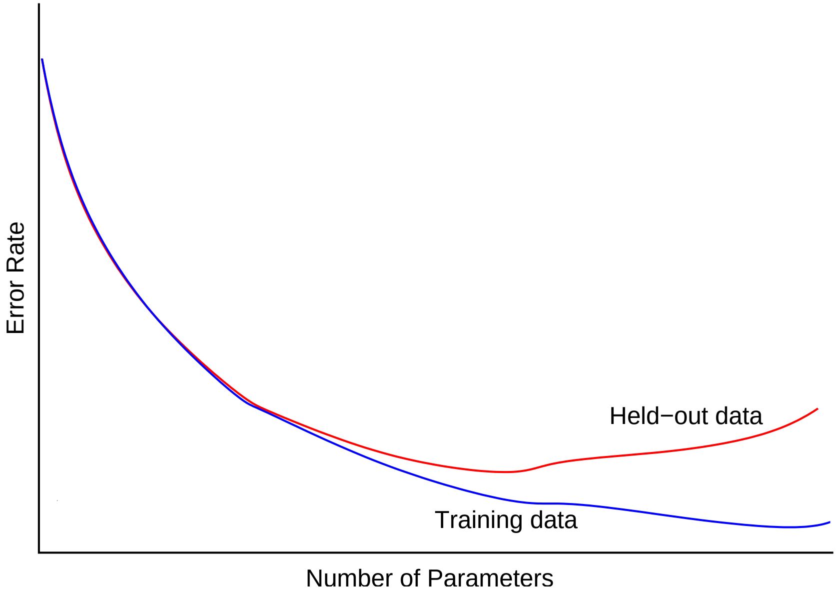
- There are two stages in a pattern recognition framework:
  - feature extraction: a feature vector, $\boldsymbol{x}$, is derived from the "observations";
  - classification: a class ω is identified given the feature vector $\boldsymbol{x}$:
- Example: sentiment analysis
  - $\boldsymbol{w}$ is the document (words)
  - $\boldsymbol{x}$ is the a binary vector whether a particular word is in the document
  - ω is the sentiment (e.g. angry)
- Need to design a suitable feature vector and classifier for task.

Training and Evaluation Data

- The basic machine learning framework has two sets of data:
  1. Training data: is used to train the classifier - data may be:
     - supervised: the correct classes of the training data are known.
     - unsupervised: the correct classes of the training data are not known
     - reinforcement learning: don't learn a model - directly learn an action!
  2. Test data: held-out data for evaluating the classifier

  Supervised training data will be mostly considered in this course
- It is important that the training and test data do not overlap
  - performance on training data better than on held-out data
  - becomes more important as the classifiers become more complex
  - development data sometimes used to tune parameters
- Aim to build a classifier that performs well on held-out data, generalise.

Generalisation



- Performance typically goes as left
  - increasing model parameters,
  - improves training data performance
  - but not always test data performance
- What complexity of classifier to use?

- Consider various regions of the graph

1. Too Simple: The models used are relatively simple. The performance on the training and test data is about the same as the models are "well" trained.
2. Optimal: This is where the error rate on the test data is at a minimum. This is where we want to be.
3. Too Complex: The models perform very well on the training data. However the models perform badly on the test data.

Machine Learning Based Decisions

- Need a systematic well-motivated approach for making decisions - consider
  - classifier: form and how to train the classifier
  - decision rule given a classifier: how to use it
- Consider a system where
  - observation: feature vector of dimension d, $\boldsymbol{x}$
  - class labels: there are K classes, denoted by ω_1, $\omega_2, ...,$ ω_K.
- Classifiers for making decisions can be broadly split as:
  - Generative models: a model of the joint distribution of observations and classes is trained, $P(\boldsymbol{x}, \omega_j)$.
  - Discriminative models: a model of the posterior distribution of the class given the observation is trained, $P(\omega_j|\boldsymbol{x})$.
  - Discriminant functions: a mapping from an observation $\boldsymbol{x}$ to class ω_j is directly trained. No posterior probability, $P(\omega_j|\boldsymbol{x})$, generated just class labels.

Bayes' Decision Theory

- Given a classifier create a decision rule to minimise average probability of error.

$$P(\text{error}) = \sum_{\boldsymbol{x}\in\mathcal{X}} P(\text{error}, \boldsymbol{x}) = \sum_{\boldsymbol{x}\in\mathcal{X}} P(\text{error}|\boldsymbol{x})P(\boldsymbol{x})$$

  – for a two class problem, the conditional probability of error can be written

$$P(\text{error}|\boldsymbol{x}) \quad = \quad \begin{cases} P(\omega_1|\boldsymbol{x}) & \text{if we decide } \omega_2 \\ P(\omega_2|\boldsymbol{x}) & \text{if we decide } \omega_1 \end{cases}$$

- Minimising $P(\text{error}|\boldsymbol{x})$ for each $\boldsymbol{x}$ minimises the average probability of error.
  – this gives Bayes' decision rule, which for a two class problem is

$$\text{Decide} \begin{cases} \text{Class } \omega_1 & \text{if } P(\omega_1|\boldsymbol{x}) > P(\omega_2|\boldsymbol{x}) \\ \text{Class } \omega_2 & \text{otherwise} \end{cases}, \qquad \frac{P(\omega_1|\boldsymbol{x})}{P(\omega_2|\boldsymbol{x})} \mathop{\gtrless}_{\omega_2}^{\omega_1} 1$$

  – for multiple classes select according to $\hat{\omega} = \operatorname{argmax}_{\omega} \{P(\omega|\boldsymbol{x})\}$

Generative Models

- For generative models the joint distribution is found - often expressed as

$$P(\boldsymbol{x}, \omega_j) = P(\boldsymbol{x}|\omega_j)P(\omega_j)$$

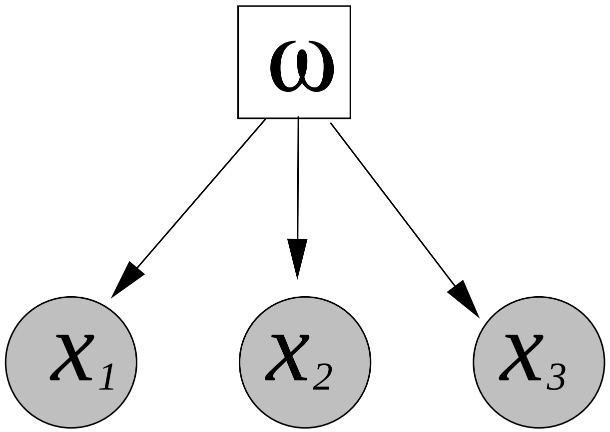
- Form of classifier considered has two parts
  - prior probabilities: an idea of how frequent each class is, $P(\omega_1), \ldots, P(\omega_K)$.
  - class-conditional (likelihood) probability: the PMF of the feature vector for each class $P(\boldsymbol{x}|\omega_1), \ldots, P(\boldsymbol{x}|\omega_K)$.

- For an unknown observation $\boldsymbol{x}$, Bayes' rule allows the calculation of posterior probability of class membership.

$$P(\omega_j|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\omega_j)P(\omega_j)}{\sum_{k=1}^{K} P(\boldsymbol{x}|\omega_k)\ P(\omega_k)}, \quad j = 1, 2, ..., K$$

Naive Bayes' Classifier

- Simple form of generative model:
  - joint distribution: $P(\boldsymbol{x}, \omega_j) = P(\omega_j) \prod_{i=1}^{d} P(x_i|\omega_j)$
  - classification: $P(\omega_j|\boldsymbol{x}) \propto P(\omega_j) \prod_{i=1}^{d} P(x_i|\omega_j)$
- Elements of the feature vector conditionally independent given the class



- write as a Bayesian Network (BN)
  - shaded observed variable
  - unshaded unobserved variable
  - circle continuous variable
  - square discrete variable

- More on Bayesian Networks (and Graphical Models) later in the module

Probability Distributions

- For generative models need to decide form of conditional distribution $P(\boldsymbol{x}|\omega_j)$
  - (d-dimensional) feature vector may be discrete or continuous

- Discrete distributions (probability mass functions) - primary interest here
  - Multivariate-Bernoulli distribution: $x_i \in \{0, 1\}$,

$$P(\boldsymbol{x}|\omega_j) = \prod_{i=1}^{d} p_{ji}^{x_i}(1 - p_{ji})^{1-x_i}; \qquad 0 \leq p_{ji} \leq 1$$

  - Multinomial distribution: $x_i \in \{0, \ldots, n\}$

$$P(\boldsymbol{x}|\omega_j) = \frac{n!}{\prod_{i=1}^{d} x_i!} \prod_{i=1}^{d} p_{ji}^{x_i}, \qquad n = \sum_{i=1}^{d} x_i, \quad \sum_{i=1}^{d} p_{ji} = 1, \quad p_{ji} \geq 0$$

- Continuous distribution, $x_i \in [-\infty, \infty]$, less interest on this module

Maximum Likelihood Training

- The class-conditional distribution $P(\boldsymbol{x}|\omega_j)$ needs to be trained
  - only the data from the class of interest used
  - for class ω_j with n training examples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$

$$\hat{\boldsymbol{\lambda}}_j = \operatorname*{argmax}_{\boldsymbol{\lambda}} \left\{ \prod_{\tau=1}^{n} P(\boldsymbol{x}_\tau|\boldsymbol{\lambda}) \right\} = \operatorname*{argmax}_{\boldsymbol{\lambda}} \left\{ \sum_{\tau=1}^{n} \log\left(P(\boldsymbol{x}_\tau|\boldsymbol{\lambda})\right) \right\}$$

- For the multivariate Bernoulli distribution: $\boldsymbol{\lambda}_j = \{p_{j1}, \ldots, p_{jd}\}$

$$\hat{\boldsymbol{\lambda}}_j = \operatorname*{argmax}_{\boldsymbol{\lambda}_j} \left\{ \sum_{\tau=1}^{n} \sum_{i=1}^{d} x_{\tau i} \log(p_{ji}) + (1 - x_{\tau i}) \log(1 - p_{ji}) \right\}$$

Differentiating wrt $\boldsymbol{\lambda}_j$ and equating to zero yields: $p_{ji} = \frac{1}{n} \sum_{\tau=1}^{n} x_{\tau i}$

Improving the Basic Model

- Incorporating a Prior: What happens if a count is zero?
  - simplest solution to initialise counts with a constant α: for Bernoulli

$$p_{ji} = \frac{1}{\alpha + n}\left(\alpha + \sum_{\tau=1}^{n} x_{\tau i}\right)$$

  - more details on this topic in discussion of language models
- Mixture Model: more "powerful" distribution combining multiple distributions:

$$P(\boldsymbol{x}|\omega_j) = \sum_{m=1}^{M} P(c_m|\omega_j)P(\boldsymbol{x}|c_m, \omega_j)$$

  - component c_m has prior, $P(c_m|\omega_j)$ and probability distribution, $P(\boldsymbol{x}|c_m, \omega_j)$
  - more details on this topic in the lectures on graphical models

Limitations of Generative Models

- An obvious question is:

  If Bayes' decision rule minimises average probability of error and we can train generative models - why do anything else?

  Plus often able to estimate parameters simply by counting!

- The classifier is the minimum error classifier only if
  - form of the class-conditional distribution is correct
  - training sample set is infinite
  - training algorithm finds the correct parameters
  - correct prior is used

  None of these are usually true!, but still used for some tasks

Discriminative Models

- Classification requires the class-posterior $P(\omega_j|\boldsymbol{x})$
  - can just directly model the posterior distribution
  - avoids the complexity of modelling the joint distribution $P(\boldsymbol{x}, \omega_j)$
- Form of model called a discriminative model
- Many debates of generative versus discriminative models:
  - discriminative model criterion more closely related to classification process
  - not dependent on generative process being correct
  - joint distribution can be very complicated to accurately model
  - only final posterior distribution needs to be a valid distribution
- Initially consider classifiers that yield linear decision boundaries

Linear Decision Boundaries

- Decision boundaries partition the feature-space into regions
  - associated with each region is a class label
  - linear decision boundaries use:
    lines ($d = 2$); planes ($d = 3$); hyper-planes ($d > 3$) to determine regions
- Initially only binary (2-class) classification tasks will be considered

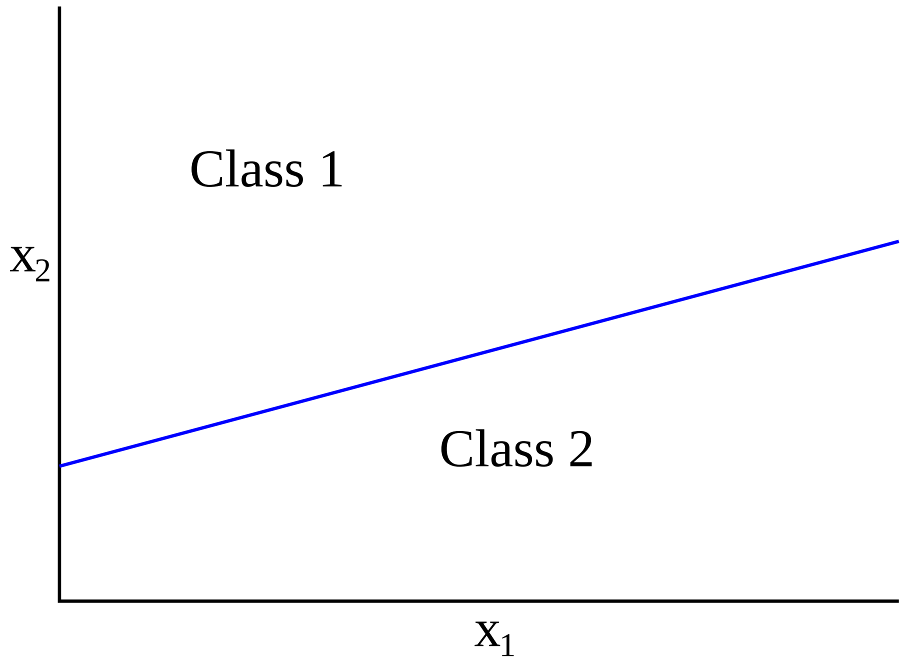
Consider linear decision boundary where

$$g(\boldsymbol{x}) = \mathbf{w}^{\mathsf{T}}\boldsymbol{x} + b$$

Decisions are made based on

$$\text{Decide} \begin{cases} \text{class } \omega_1 & g(\boldsymbol{x}) > 0 \\ \text{class } \omega_2 & g(\boldsymbol{x}) < 0 \end{cases}$$

Parameters of decision boundary b & $\mathbf{w}$



Logistic Regression/Classification

- For binary classification - logistic regression classification is a simple form
  - class posterior probability a function of the perpendicular distance to the decision boundary

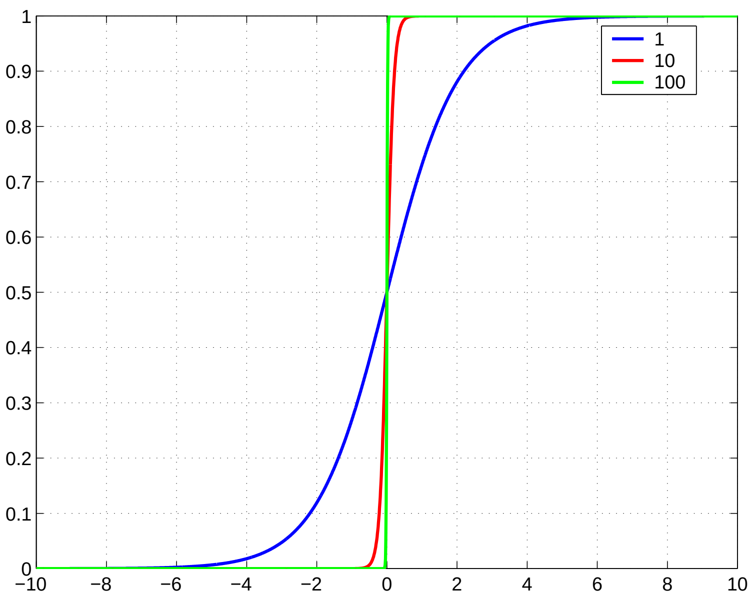
$$P(\omega_1|\boldsymbol{x}) = \frac{1}{1+\exp(-(\mathbf{w}^\mathsf{T}\boldsymbol{x}+b))}; \quad P(\omega_2|\boldsymbol{x}) = \frac{\exp(-(\mathbf{w}^\mathsf{T}\boldsymbol{x}+b))}{1+\exp(-(\mathbf{w}^\mathsf{T}\boldsymbol{x}+b))}$$

  - linear decision boundary - need to find $\boldsymbol{\lambda} = \{\mathbf{w}, b\}$.
- Often parameters combined into single vectors

$$\tilde{\boldsymbol{x}} = \left[\begin{array}{c}\boldsymbol{x}\\1\end{array}\right], \quad \tilde{\mathbf{w}} = \left[\begin{array}{c}\mathbf{w}\\b\end{array}\right], \qquad \tilde{\mathbf{w}}^\mathsf{T}\tilde{\boldsymbol{x}} = \mathbf{w}^\mathsf{T}\boldsymbol{x} + b$$

Form of Posterior

- The posterior distribution of class 1 and perpendicular distance to the decision boundary



- a sigmoid function

$$P(\omega_1|\boldsymbol{x}) = \frac{1}{1 + \exp(-\rho z)}$$

where $\rho z = \mathbf{w}^{\mathsf{T}}\boldsymbol{x} + b$

- diagram shows variations with ρ

- Posterior is only a function of perpendicular distance to the decision boundary
- Altering the value of the priors simply moves the sigmoid to the right or left

MaxEnt Model

- A multi-class generalisation of logistic regression is the MaxEnt model

$$P(\omega_j|\boldsymbol{x}) = \frac{1}{Z}\exp\left(\mathbf{w}_j^{\mathsf{T}}\boldsymbol{x} + b_j\right); \quad Z = \sum_{k=1}^{K}\exp\left(\mathbf{w}_k^{\mathsf{T}}\boldsymbol{x} + b_k\right)$$

Z is the normalisation term

- A more general form of MaxEnt model considers class-specific features
  - can use a feature function of the observation and class, $f(\boldsymbol{x}, \omega_j)$

$$P(\omega_j|\boldsymbol{x}) = \frac{1}{Z}\exp\left(\sum_{i=1}^{D}\lambda_i f_i(\boldsymbol{x}, \omega_j)\right); \quad Z = \sum_{k=1}^{K}\exp\left(\sum_{i=1}^{D}\lambda_i f_i(\boldsymbol{x}, \omega_k)\right)$$

  - D is the size of the combined feature-vector for all classes

MaxEnt Model (cont)

- Feature-functions can represent the simple linear form

$$\boldsymbol{\lambda} = \begin{bmatrix} \mathbf{w}_1 \\ b_1 \\ \vdots \\ \mathbf{w}_K \\ b_K \end{bmatrix}; \quad \mathbf{f}(\boldsymbol{x}, \omega_1) = \begin{bmatrix} \boldsymbol{x} \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}; \quad \mathbf{f}(\boldsymbol{x}, \omega_k) = \begin{bmatrix} 0 \\ \vdots \\ \boldsymbol{x} \\ 1 \\ 0 \\ \vdots \end{bmatrix}$$

- Relationship to logistic regression (note binary case, restricted feature function)

$$\begin{aligned} P(\omega_1|\boldsymbol{x}) &= \frac{\exp\left(\mathbf{w}_1^\mathsf{T}\boldsymbol{x} + b_1\right)}{\exp\left(\mathbf{w}_1^\mathsf{T}\boldsymbol{x} + b_1\right) + \exp\left(\mathbf{w}_2^\mathsf{T}\boldsymbol{x} + b_2\right)} \\ &= \frac{1}{1 + \exp\left((\mathbf{w}_2 - \mathbf{w}_1)^\mathsf{T}\boldsymbol{x} + (b_2 - b_1)\right)} \end{aligned}$$

Discriminative Model Training

- Similar criterion to ML-training of generative models
  - maximise posterior of the correct class (rather than generating the data)

$$\hat{\boldsymbol{\lambda}} = \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \left\{ \sum_{\tau=1}^{N} \log \left(P(y_\tau | \boldsymbol{x}_\tau, \boldsymbol{\lambda}) \right) \right\}$$

  where $y_\tau \in \{\omega_1, \ldots, \omega_K\}$ and training data $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$

- Useful properties
  - multi-class training of parameters $\boldsymbol{\lambda}$
  - all training examples used to derive all the model parameters
  - different features may be used for different classes
    (generative model requires same feature-space)

Training MaxEnt Models

- Training MaxEnt model is a convex optimisation problem
  - one solution to train parameters is generalised iterative scaling

$$\lambda_i^{(l+1)} = \lambda_i^{(l)} + \frac{1}{C}\log\left(\frac{\sum_{\tau=1}^{N} f_i(\boldsymbol{x}_\tau, y_\tau)}{\sum_{\tau=1}^{N}\sum_{k=1}^{K} P(\omega_k|\boldsymbol{x}_\tau, \boldsymbol{\lambda}^{(l)}) f_i(\boldsymbol{x}_\tau, \omega_k)}\right)$$

  - iterative approach (parameters at iteration l are $\boldsymbol{\lambda}^{(l)}$)

- (strictly) requires that the features add up to a constant

$$\sum_{i=1}^{D} f_i(\boldsymbol{x}_\tau, \omega_k) = C, \quad \forall \tau, k$$

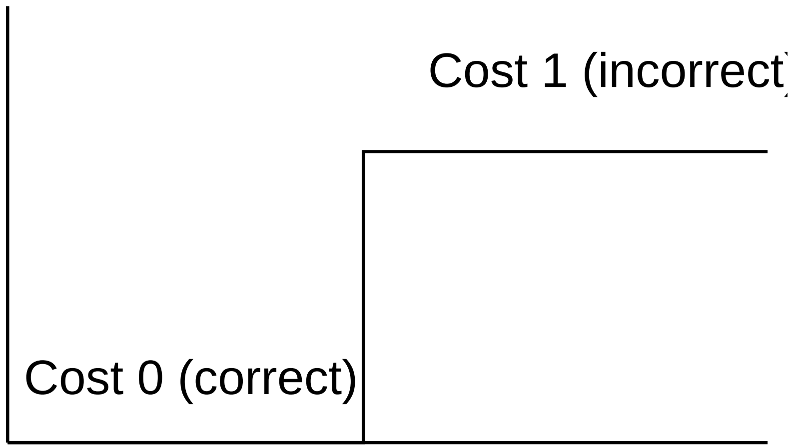
  - extensions relaxes this requirements, e.g. improved iterative scaling

Discriminative Functions

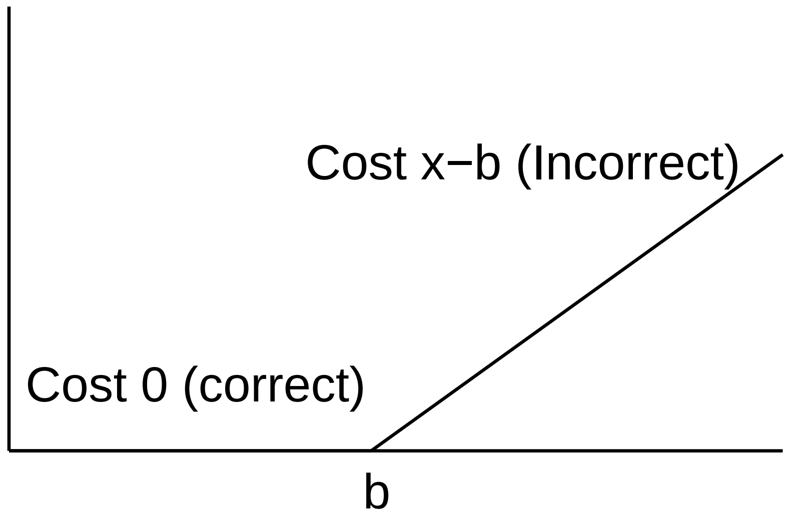
- Aim to directly minimise the error rate
  - all we care about for classification
  - don't need to determine class posteriors, $P(\omega_j|\boldsymbol{x})$, just decision boundaries
- Wide range of possible optimisation schemes/criteria (Duda, Hart and Stork)
  - perceptron algorithm described in this lecture
- Some disadvantages to not obtaining a posterior probability: issues with
  - reject option: don't quote answer if posterior below a threshold
  - compensating for class priors: priors are sometimes estimated on different data (e.g. language models)
  - combining models: sometimes build and combine multiple systems using posteriors (could use voting ...)

Cost Functions

- Need to decide on the criterion, cost function, to train the decision boundary



ideal cost function



perceptron cost function

- Would like to minimise the error rate - ideal cost function above shown above
  - differentiating this function yields a delta-function
  - awkward to train (consider gradient descent with this)
- An alternative is the perceptron algorithm, which is differentiable
  - the distance of incorrect observations from the decision boundary

Perceptron Criterion

- The perceptron criterion cost for a sample is
  - zero when correctly classified
  - perpendicular distance to the decision boundary, $g(\boldsymbol{x})$, when misclassified
- The distance to the decision boundary for misclassified samples
  - for class ω_1, $g(\boldsymbol{x}) < 0$
  - for class ω_2, $g(\boldsymbol{x}) > 0$
- To simplify training the extended training observations, $\tilde{\boldsymbol{x}}_\tau$, are normalised

$$\overline{\boldsymbol{x}}_\tau = \begin{cases} \tilde{\boldsymbol{x}}_\tau, & \text{belongs to } \omega_1 \\ -\tilde{\boldsymbol{x}}_\tau, & \text{belongs to } \omega_2 \end{cases}$$

$$\tilde{\mathbf{w}}^{\mathsf{T}}\overline{\boldsymbol{x}}_\tau > 0 \qquad\qquad \tilde{\mathbf{w}}^{\mathsf{T}}\overline{\boldsymbol{x}}_\tau < 0$$

correctly classified — misclassified

Perceptron Algorithm

- The perceptron algorithm can be used to find $\tilde{\mathbf{w}}$

1. Choose initial value $\tilde{\mathbf{w}}^{(0)}$, $l = 0$; ($\tilde{\mathbf{w}}^{(l)}$ estimate at iteration l)
2. Obtain the set of mis-classified points, $\mathcal{Y}^{(l)}$, using current estimate $\tilde{\mathbf{w}}^{(l)}$, i.e. find all the points where $\tilde{\mathbf{w}}^{(l)\mathsf{T}}\overline{\boldsymbol{x}}_\tau < 0$
3. Minimise the total perceptron criterion

$$E(\tilde{\mathbf{w}}) = \sum_{\overline{\boldsymbol{x}}_\tau \in \mathcal{Y}^{(l)}} (-\tilde{\mathbf{w}}^{\mathsf{T}}\overline{\boldsymbol{x}}_\tau)$$

update rule is

$$\tilde{\mathbf{w}}^{(l+1)} = \tilde{\mathbf{w}}^{(l)} + \sum_{\overline{\boldsymbol{x}}_\tau \in \mathcal{Y}^{(l)}} \overline{\boldsymbol{x}}_\tau$$

4. If converged ($\mathcal{Y}^{(l)}$ is empty), or max iter, STOP; else $l = l + 1$ goto (2).

Perceptron Algorithm (cont)

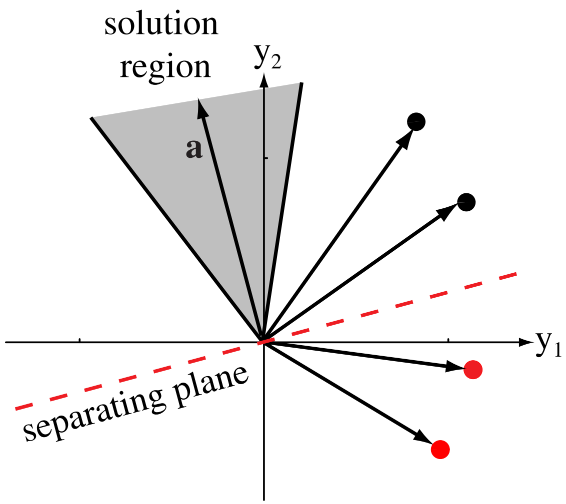
- For linearly separable data:
  - a linear decision boundary that correctly classifies all points exists
  - in this case using the algorithm guaranteed to find a solution

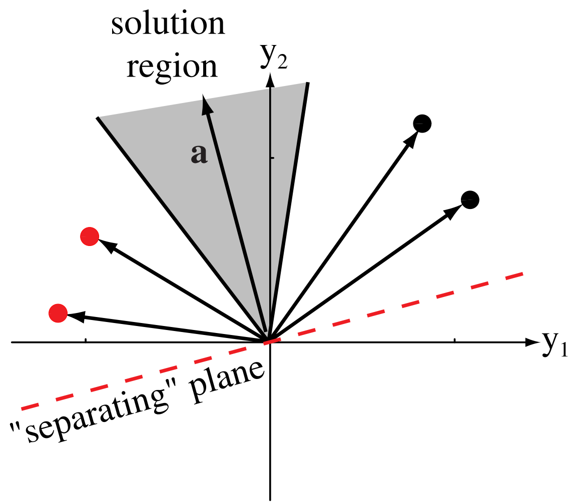
$$\tilde{\mathbf{w}}^{(l+1)} = \tilde{\mathbf{w}}^{(l)} + \sum_{\overline{\boldsymbol{x}}_\tau \in \mathcal{Y}^{(l)}} \overline{\boldsymbol{x}}_\tau$$

  - automatically stops when the set of mis-classified points is empty.
- Two forms of update may be considered:
  - batch update: all the data is presented for each iteration
    the decision boundary is only updated after all the samples have been seen
  - sequential update: data is presented one sample at a time
    the decision boundary is updated after each sample

Perceptron Solution Region

- Each sample $\tilde{\boldsymbol{x}}_\tau$ places a constraint on possible location of solution vector.
- $\tilde{\boldsymbol{w}}^\mathsf{T}\tilde{\boldsymbol{x}}_\tau = 0$ defines a hyperplane through the origin on the "weight-space" of $\tilde{\boldsymbol{w}}$ vectors with $\tilde{\boldsymbol{x}}_\tau$ as a normal vector.
- Normalised data: solution must be on the positive side of every such hyperplane
  - it is not unique (if it exists) and lies anywhere within solution region
- Figure shows solution region for un-normalised and normalised data (DHS).
  - note change of notation: $\mathbf{a} = \tilde{\mathbf{w}}$, $\boldsymbol{y} = \tilde{\boldsymbol{x}}$ (left) $\boldsymbol{y} = \overline{\boldsymbol{x}}$ (right)





Structured Data

- So far considered the case where the labels (training and test) are “scalar”
  - not always the case - consider translation

Speech recognition is difficult $\rightarrow$ Lleferydd cydnabyddiaeth yn anodd

- Consider the number of possible classes
  - every possible possible “sentence” translation - vast
- This is also an issue with the possible features to extract
  - can consider pairs of words, triples etc etc

Need to make conditional-independence assumptions

"Theory" Part of Module

- These lectures give underlying theory for the seminars and associated papers
- Graphical Models
  - Bayesian networks and graphical models (beyond naive Bayes)
  - Markov chains (including language modelling)
  - mixture models (including Gaussian mixture models)
  - hidden Markov models (including Viterbi algorithm)
  - conditional random fields
  - expectation maximisation and variational approaches
- Support Vector Machines
  - large margin training and dual representation
  - kernel methods (including sequence kernels)
- Clustering
  - unsupervised approaches (including K-means clustering)
  - latent Dirichlet allocation