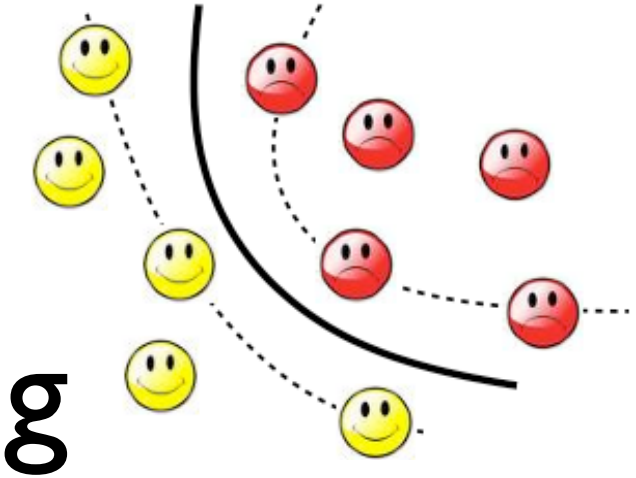


Machine Learning and Data Mining

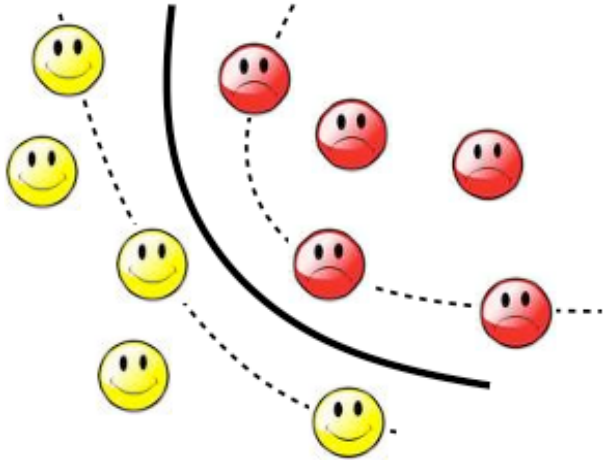
(COMP 5318)

Logistic Regression and SVMs

Fabio Ramos
Roman Marchant

Announcements

- Assignment 1 will be out this week
  Groups of 2 or 3.
- This lecture is based on:
  - Murphy's book: Chapters 8, 14
  - Bishop's book: Chapters 4, 7
  - Ullman's book: Chapter 12



Quick Review

Supervised learning

THE UNIVERSITY OF SYDNEY

- Learn a mapping function *f* from *x* to *t*

$$t = f(\mathbf{x})$$

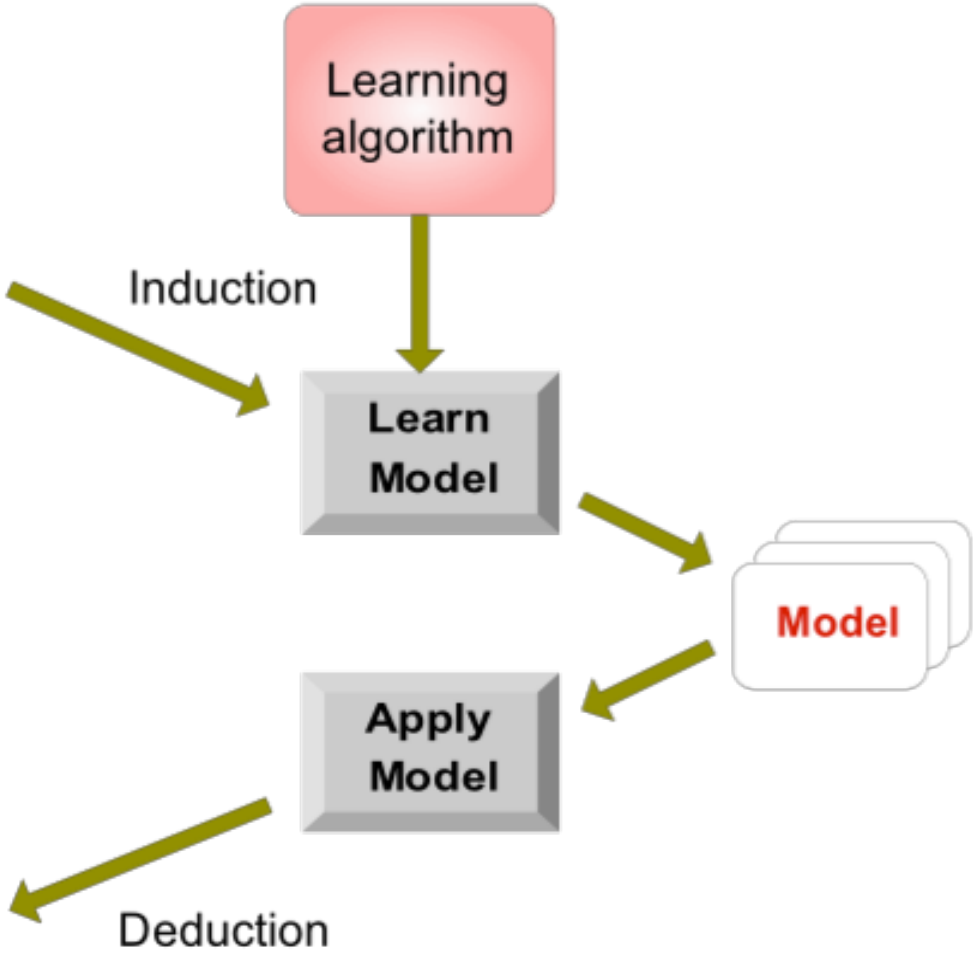
- If $t \in \{1, 2, 3, \ldots, C\}$ the problem is called classification
- If $t \in \mathbb{R}$ the problem is called regression

Illustrating Classification Task

THE UNIVERSITY OF SYDNEY

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

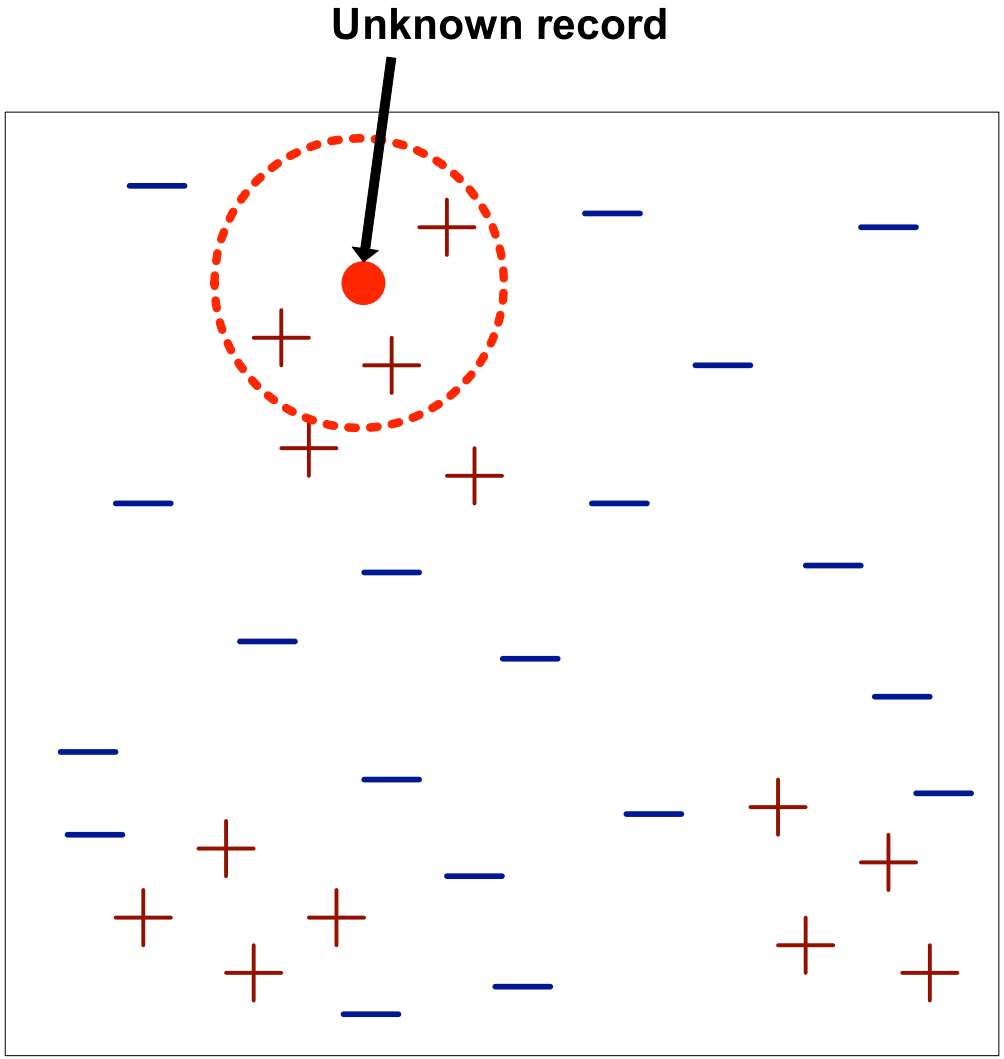
Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

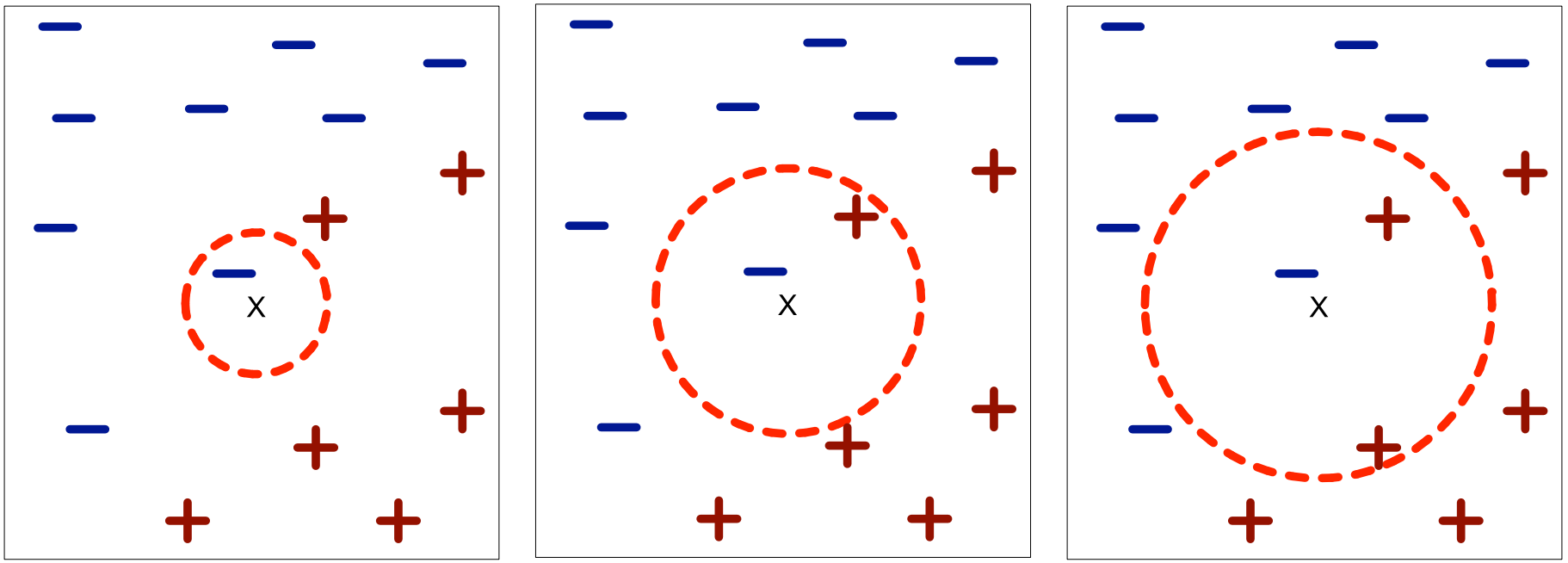
Nearest Neighbour Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbours to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbours
  - Use class labels of nearest neighbours to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbour



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbours of a record x are data points that have the k smallest distance to x

Bayesian Classifiers

THE UNIVERSITY OF SYDNEY

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes' theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - Choose value of C that maximises
    $P(C \mid A_1, A_2, \ldots, A_n)$
  - Equivalent to choosing value of C that maximises
    $P(A_1, A_2, \ldots, A_n \mid C)\, P(C)$
- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?

Naïve Bayes Classifier

THE UNIVERSITY OF SYDNEY

- Assume independence among attributes A_i when class is given:
  - $P(A_1, A_2, \ldots, A_n | C) = P(A_1 | C_j)\, P(A_2 | C_j) \ldots P(A_n | C_j)$
  - Can estimate $P(A_i | C_j)$ for all A_i and C_j.
  - New point is classified to C_j if $P(C_j) \prod P(A_i | C_j)$ is maximal.

How to Estimate Probabilities from Data?

THE UNIVERSITY OF SYDNEY

#	Refund	Status	Salary	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class: $P(C) = N_c/N$
  - e.g., $P(No) = 7/10$, $P(Yes) = 3/10$
- For discrete attributes:

$$P(A_i \mid C_k) = |A_{ik}| / N_{c^k}$$

  - where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
  - Examples:
    $P(Status=Married|No) = 4/7$
    $P(Refund=Yes|Yes) = 0$

Metrics for Performance Evaluation

THE UNIVERSITY OF SYDNEY

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Cost-Sensitive Measures

THE UNIVERSITY OF SYDNEY

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F - measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

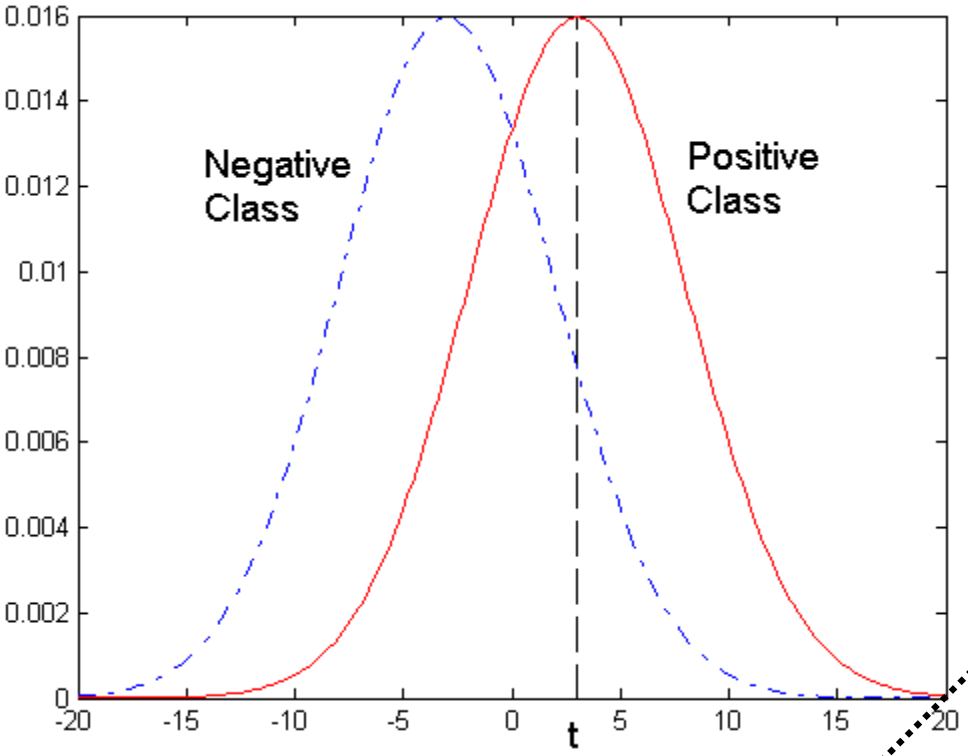
d: TN (true negative)

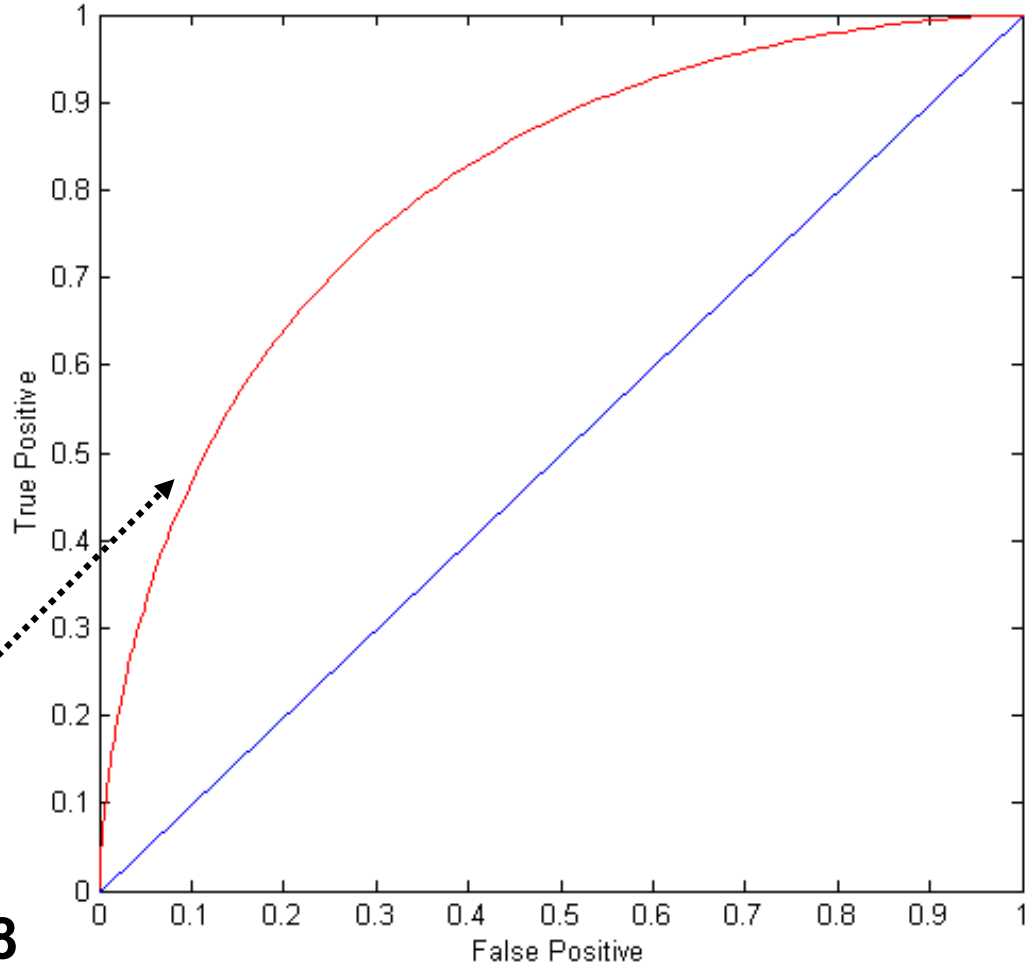
- Precision is biased towards C(Yes | Yes) & C(Yes | No)
- Recall is biased towards C(Yes | Yes) & C(No | Yes)
- F-measure is biased towards all except C(No | No)

$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$

ROC Curve

- 1-dimensional data set containing 2 classes (positive and negative)
- any points located at x > t is classified as positive





At threshold t:

TP=0.5, FN=0.5, FP=0.12, FN=0.88

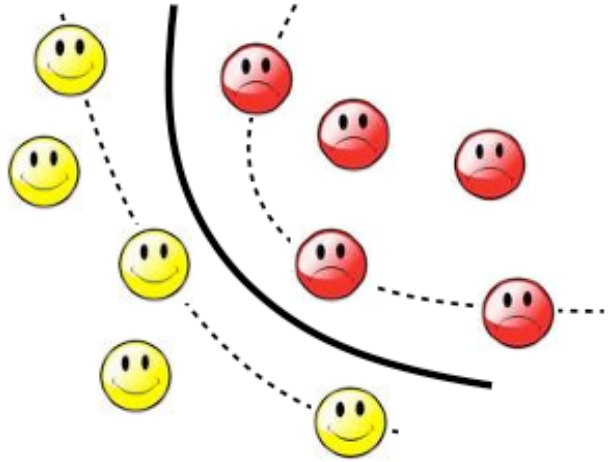
How to construct a ROC curve

THE UNIVERSITY OF SYDNEY

Instance	$P(+\|A)$	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

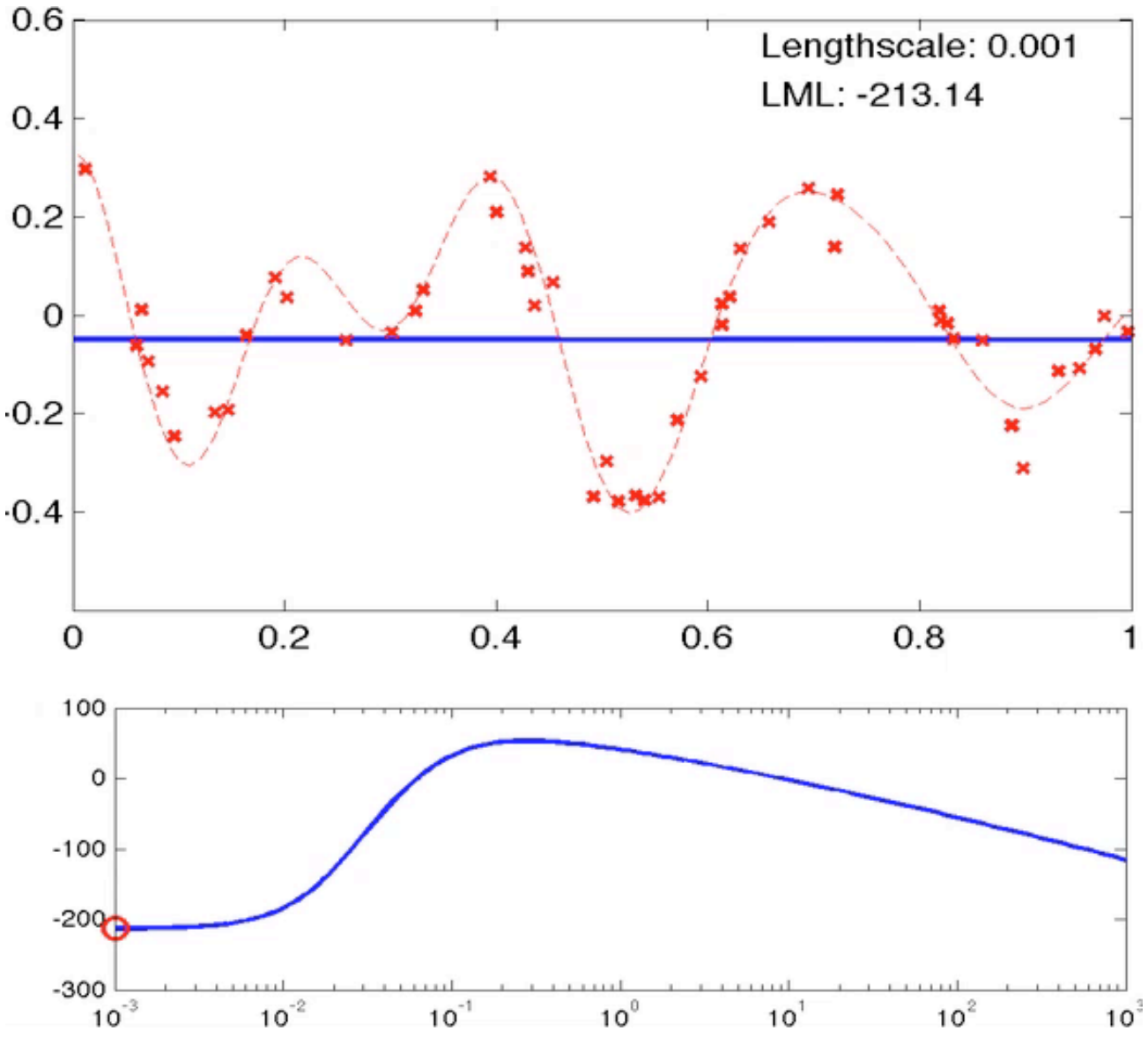
- Use classifier that produces posterior probability for each test instance $P(+ \mid A)$
- Sort the instances according to $P(+ \mid A)$ in decreasing order
- Apply threshold at each unique value of $P(+ \mid A)$
- Count the number of TP, FP, TN, FN at each threshold
- TP rate, TPR = TP/(TP+FN)
- FP rate, FPR = FP/(FP + TN)

THE UNIVERSITY OF SYDNEY



Classification II

Learning as optimisation



Loss functions

THE UNIVERSITY OF SYDNEY

- Squared error, 0-1 Loss

$$L(y, \hat{y}) = (y - \hat{y})^2$$

$$L(y, \hat{y}) = I(y \neq \hat{y})$$

- Minimise risk, (expected loss, empirical loss)

$$R(\hat{f}) = E_{\mathbf{x},y} L(f(\mathbf{x}), \hat{f}(\mathbf{x}))$$

$$\hat{R}(\hat{f}) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{f}(\mathbf{x}_i))$$

Loss for density estimation

- Suppose output is $\hat{p}(y|\mathbf{x})$, truth is $p(y|\mathbf{x})$
- Use KL (Kullback-Leibler) Loss

$$L(p(y|\mathbf{x}), \hat{p}(y|\mathbf{x})) = KL(p(y|\mathbf{x}), \hat{p}(y|\mathbf{x})) = \sum_y p(y|\mathbf{x}) \log \frac{p(y|\mathbf{x})}{\hat{p}(y|\mathbf{x})}$$

- Risk is expected negative log likelihood

$$R(\hat{p}) = -E_{\mathbf{x}} \sum_y p(y|\mathbf{x}) \log \hat{p}(y|\mathbf{x}) = -E_{\mathbf{x},y} \log \hat{p}(y|\mathbf{x})$$

Estimation vs Inference

- Learning as optimisation (frequentist): Given D, choose $\hat{f}$ to approximate f as closely as possible, so as to minimise (future) expected loss
- Usually compute parameter estimate $\hat{\theta}$
- Learning as inference (Bayesian): Given D, compute posterior over functions $p(f|D)$
- Or posterior over parameters

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

- Decision theory demonstrates that one of the best ways to minimise frequentist risk is to be Bayesian.

Generative vs Discriminative

THE UNIVERSITY OF SYDNEY

- Generative approach:
  - Model $p(t, \mathbf{x}) = p(\mathbf{x}|t)p(t)$
  - Use Bayes' theorem $p(t|\mathbf{x}) = \dfrac{p(\mathbf{x}|t)p(t)}{p(\mathbf{x})}$

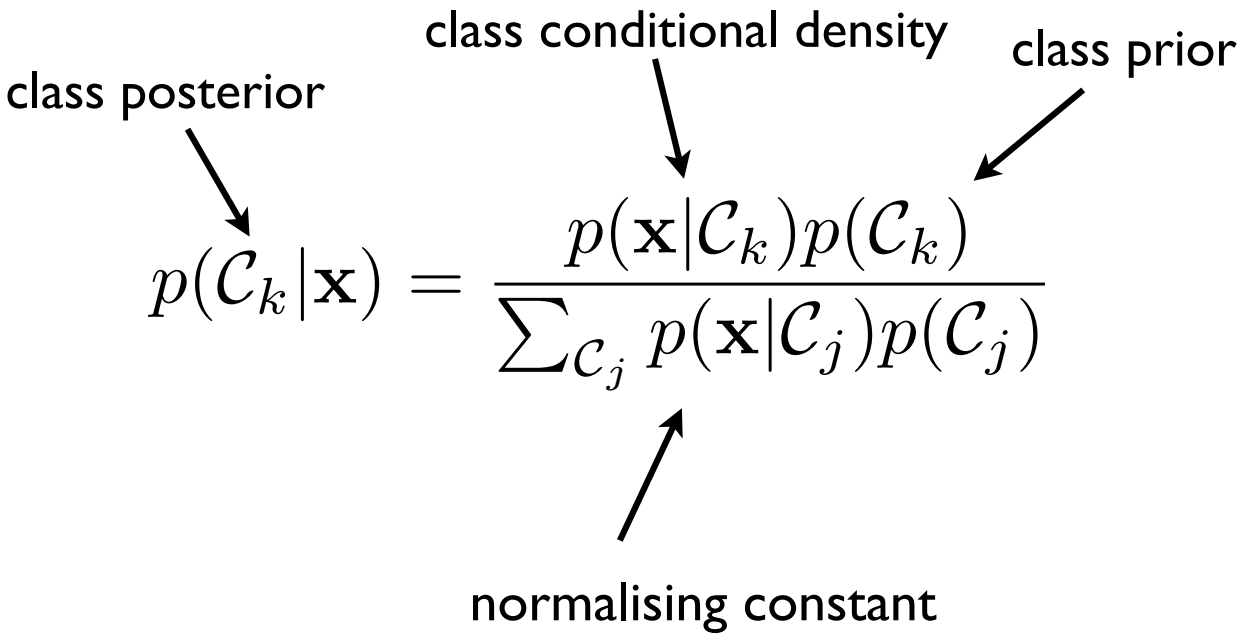
- Discriminative approach:
  - Model $p(t|\mathbf{x})$ directly

Naïve Bayes Classifier

THE UNIVERSITY OF SYDNEY

Generative model:

$$p(\mathbf{x}, \mathcal{C}_k) = p(\mathcal{C}_k)p(\mathbf{x}_n|\mathcal{C}_k) = \pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)$$



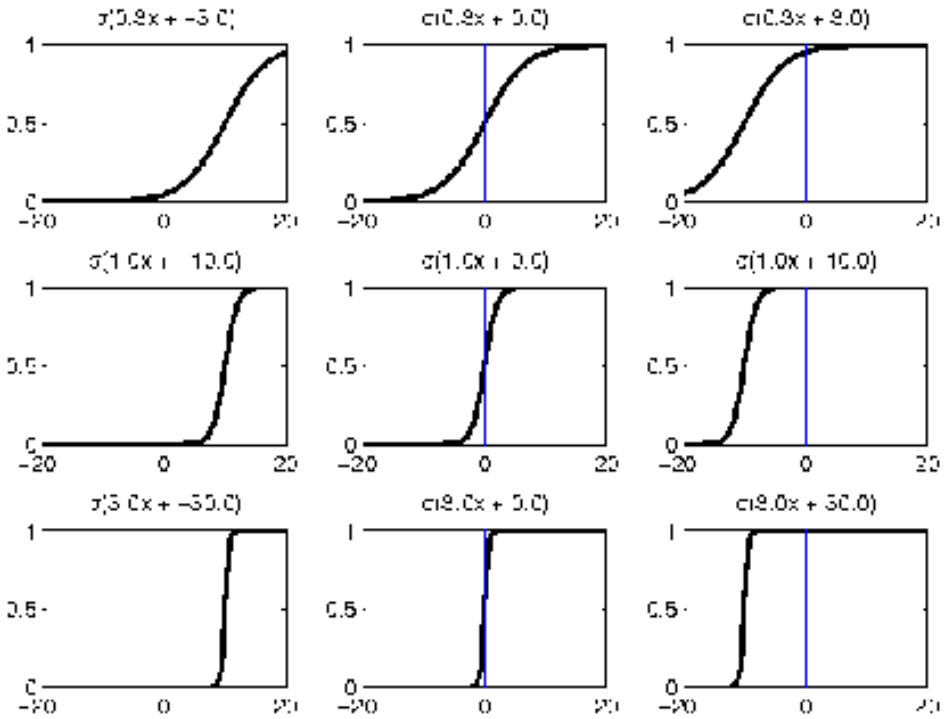
THE UNIVERSITY OF SYDNEY

Logistic Regression

Logistic Regression

- Discriminative model for binary classification

$$
\begin{aligned}
p(y|\mathbf{x}, \mathbf{w}) &= \text{Ber}(y|\sigma(\eta)) = \sigma(\eta)^y (1 - \sigma(\eta))^{1-y} \\
\eta &= \mathbf{w}^T \mathbf{x} \\
\sigma(\eta) &\stackrel{\text{def}}{=} \frac{1}{1 + \exp(-\eta)} = \frac{e^\eta}{e^\eta + 1}
\end{aligned}
$$

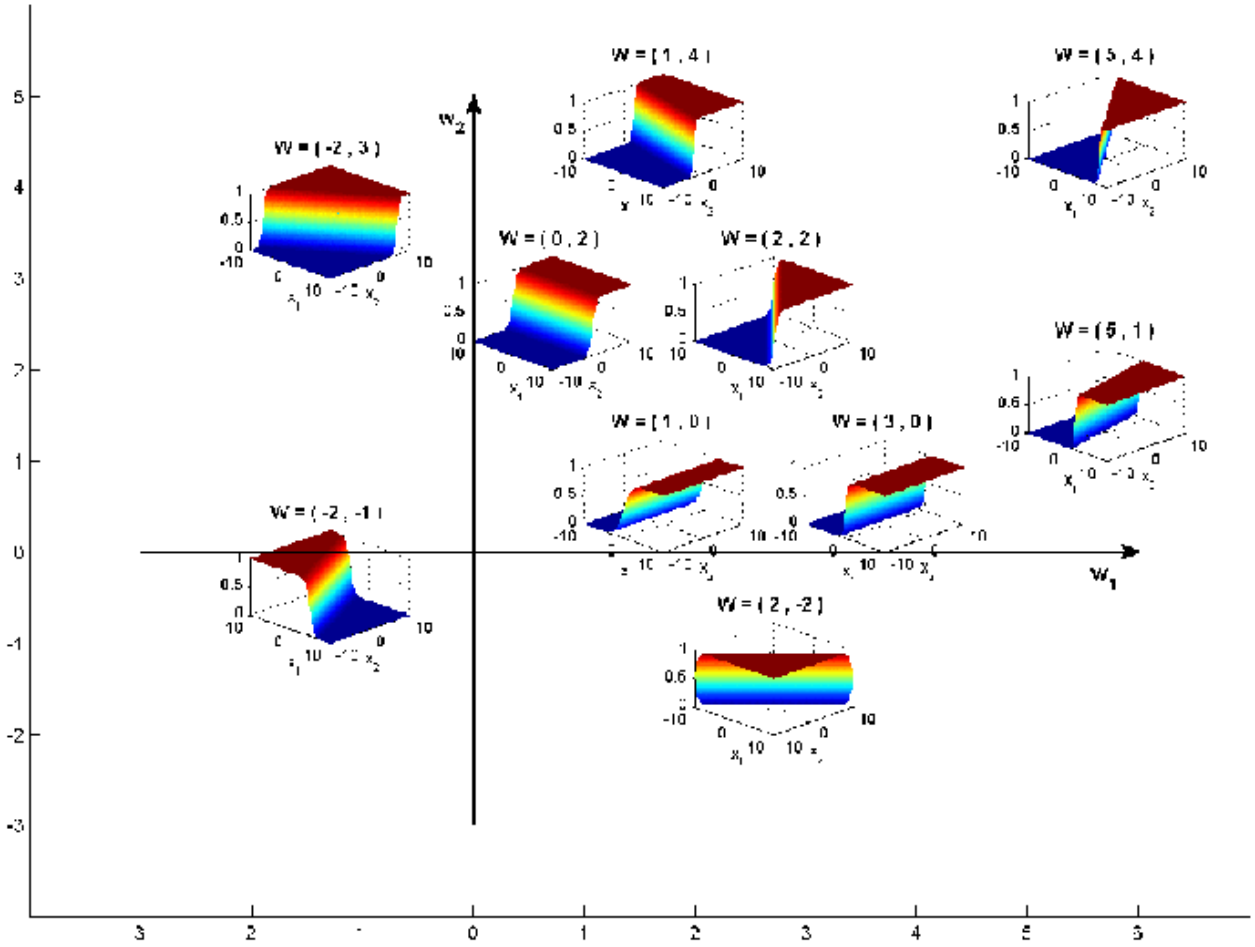


Sigmoid or Logistic function

Decision boundary

- Logistic Regression in 2D

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x})$$



Logistic Regression

- Assumes a parametric form for directly estimating P($Y \mid X$). For binary concepts, this is:

$$P(Y=1 \mid X) = \frac{1}{1+\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

$$P(Y=0 \mid X) = 1 - P(Y=1 \mid X)$$

$$= \frac{\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1+\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

- Equivalent to a one-layer backpropagation neural net.
- Logistic regression is the source of the sigmoid function used in backpropagation.
- Objective function for training is somewhat different.

Logistic Regression as a Log-Linear Model

- Logistic regression is basically a linear model, which is demonstrated by taking logs.

$$\text{Assign label } Y = 0 \text{ iff } 1 < \frac{P(Y=0 \mid X)}{P(Y=1 \mid X)}$$

$$1 < \exp(w_0 + \sum_{i=1}^{n} w_i X_i)$$

$$0 < w_0 + \sum_{i=1}^{n} w_i X_i$$

$$\text{or equivalently} \quad w_0 > \sum_{i=1}^{n} -w_i X_i$$

- Also called a **maximum entropy model** (**MaxEnt**) because it can be shown that standard training for logistic regression gives the distribution with maximum entropy that is consistent with the training data.

Logistic Regression Training

- Weights are set during training to maximise the **conditional data likelihood** :

$$W \leftarrow \underset{W}{\operatorname{argmax}} \prod_{d \in D} P(Y^d \mid X^d, W)$$

where D is the set of training examples and Y^d and X^d denote, respectively, the values of Y and X for example d.

- Equivalently viewed as maximising the **conditional log likelihood** (CLL)

$$W \leftarrow \underset{W}{\operatorname{argmax}} \sum_{d \in D} \ln P(Y^d \mid X^d, W)$$

Logistic Regression Training

- Like neural-nets, can use standard gradient descent to find the parameters (weights) that optimise the CLL objective function.
- Many other more advanced training methods are possible to speed up convergence.
  - Conjugate gradient
  - Generalised Iterative Scaling (GIS)
  - Modified Iterative Scaling (MIS)
  - Limited-memory quasi-Newton (L-BFGS)
  - Stochastic gradient descent

Logistic Regression Training

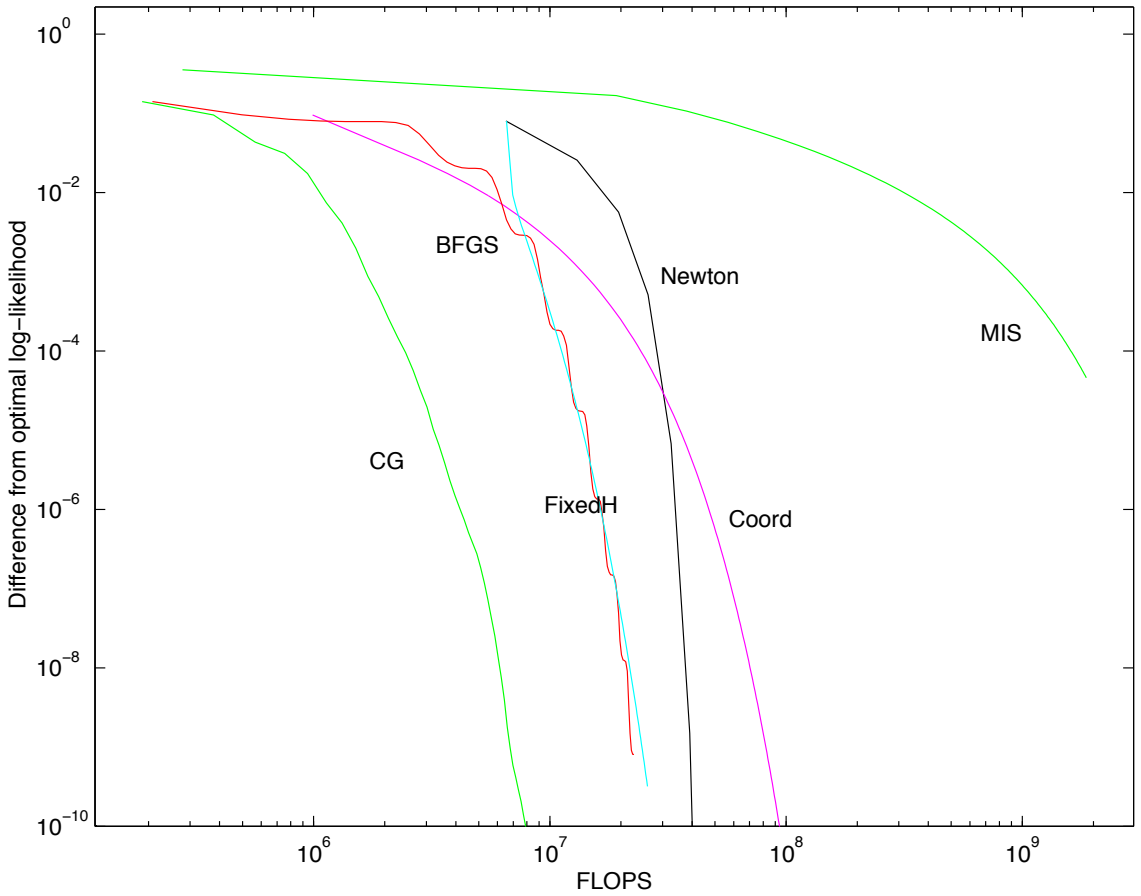
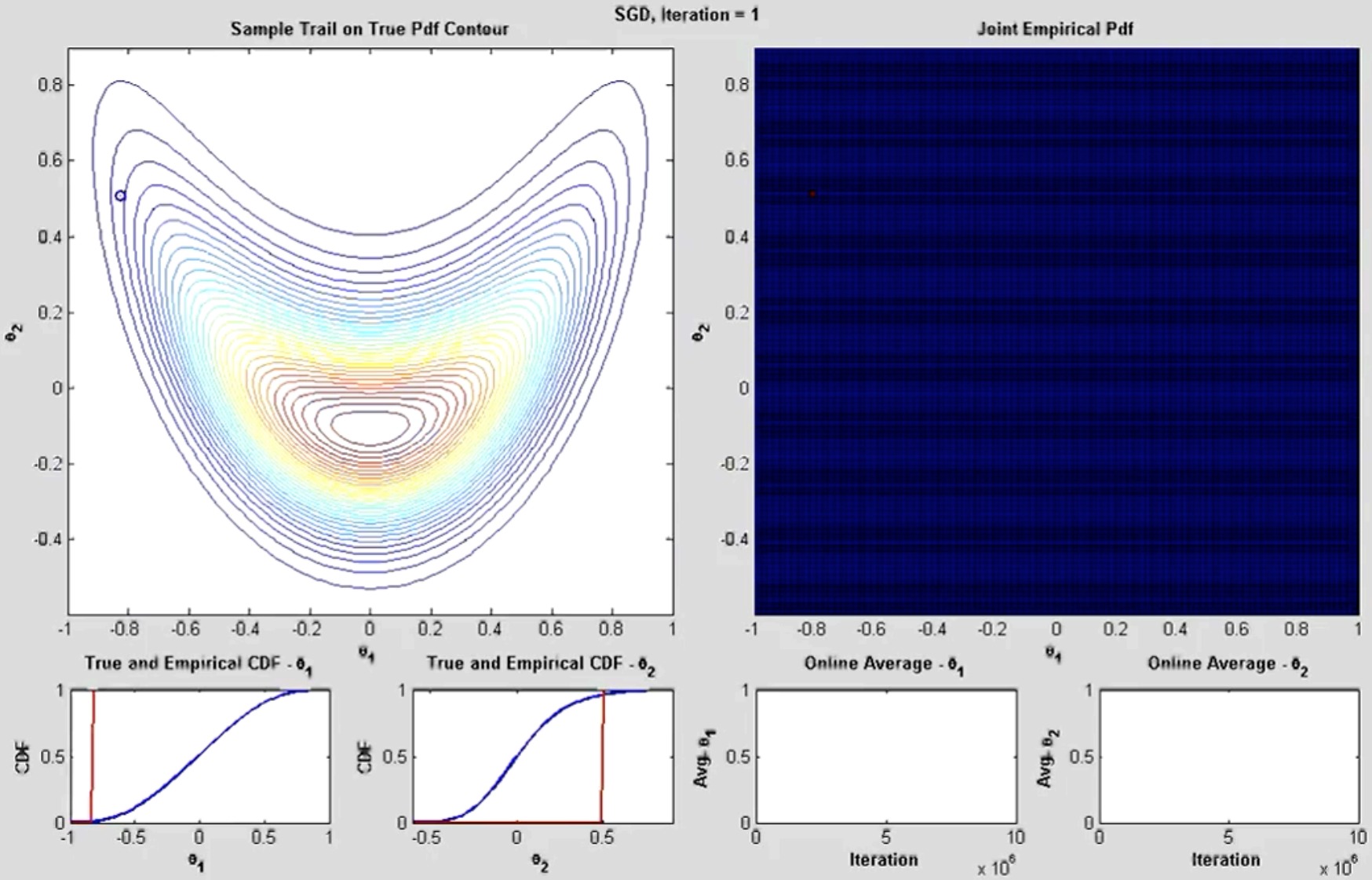


Figure 1: Cost vs. performance of six logistic regression algorithms. The dataset had 300 points in 100 dimensions. "CG" is conjugate gradient (section 4), "Coord" is coordinate-wise Newton, "FixedH" is Fixed-Hessian, and "MIS" is modified iterative scaling. CG also has the lowest actual time in Matlab.

See: "A comparison of numerical optimizers for logistic regression" by Thomas Minka, 2003.

Gradient Descent

THE UNIVERSITY OF SYDNEY



Preventing Overfitting in Logistic Regression

- To prevent overfitting, one can use **regularisation** (a.k.a. smoothing) by penalising large weights by changing the training objective:

$$W \leftarrow \underset{W}{\operatorname{argmax}} \sum_{d \in D} \ln P(Y^d \mid X^d, W) - \frac{\lambda}{2}\|W\|^2$$

Where λ is a constant that determines the amount of smoothing

- This can be shown to be equivalent to assuming a Gaussian prior for W with zero mean and a variance related to $1/\lambda$.

Multinomial Logistic Regression

- Logistic regression can be generalised to multi-class problems (where Y has a multinomial distribution).
- Effectively constructs a linear classifier for each category.

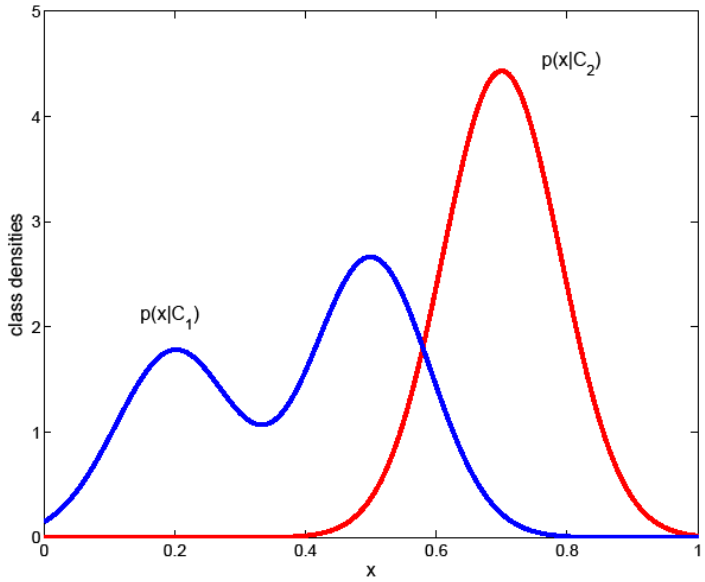
Relation Between NB and Logistic Regression

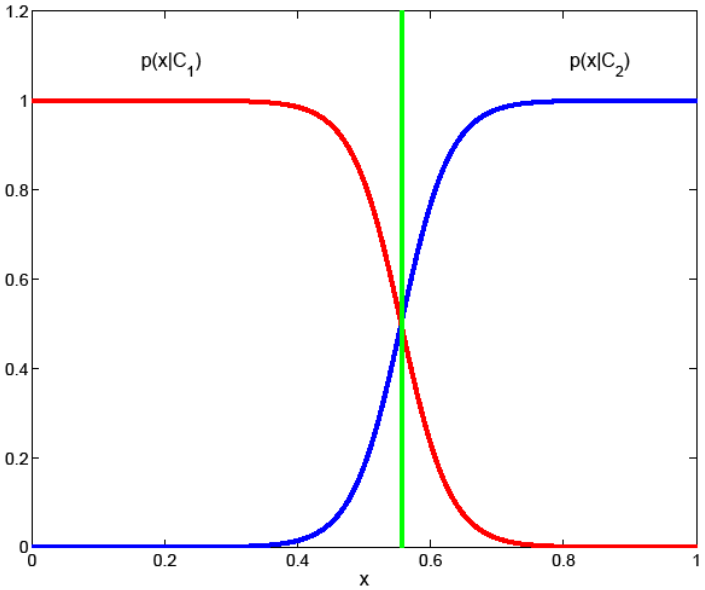
- Naïve Bayes with Gaussian distributions for features (GNB), can be shown to give the same functional form for the conditional distribution $P(Y \mid X)$.
  - But converse is not true, so Logistic Regression makes a weaker assumption.
- Logistic regression is a **discriminative** rather than generative model, since it models the conditional distribution $P(Y \mid X)$ and directly attempts to fit the training data for predicting Y from X. Does not specify a full joint distribution.

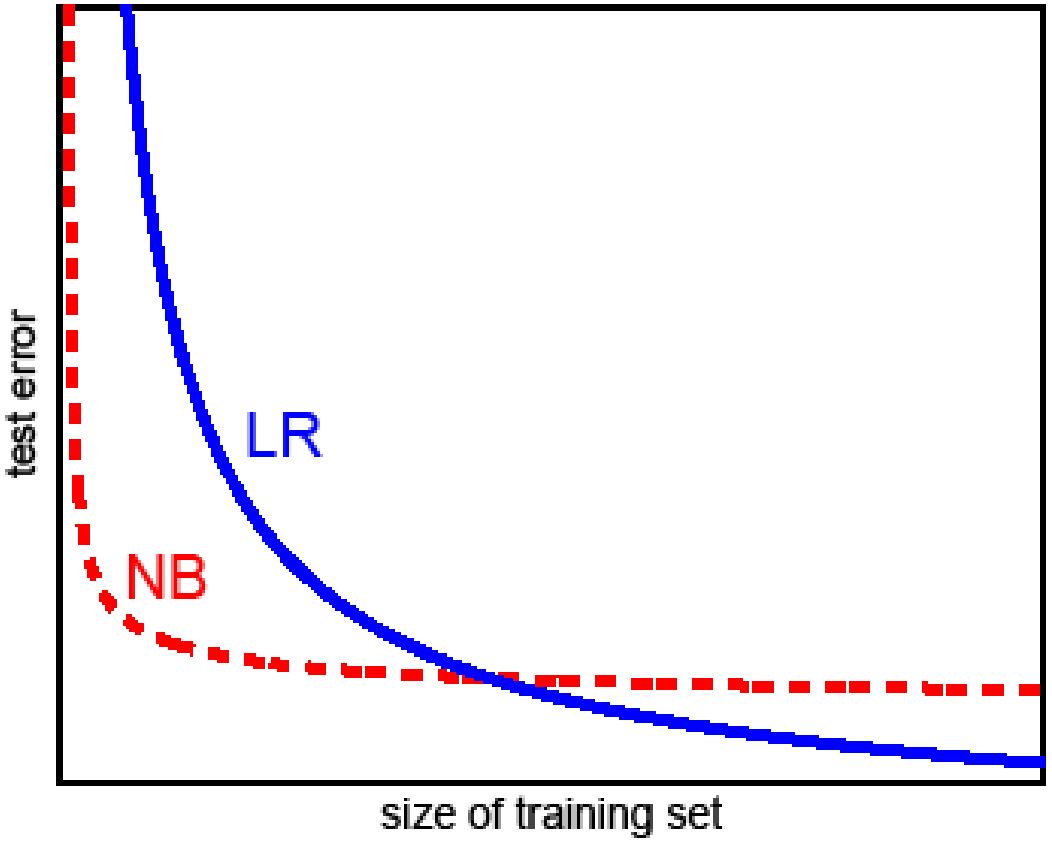
Relation Between NB and Logistic Regression (continued)

- When conditional independence is violated, logistic regression gives better generalisation if it is given sufficient training data.
- GNB converges to accurate parameter estimates faster ($O(\log n)$ examples for n features) compared to Logistic Regression ($O(n)$ examples).
- Experimentally, GNB is better when training data is scarce, logistic regression is better when it is plentiful.

Logistic Regression vs NB



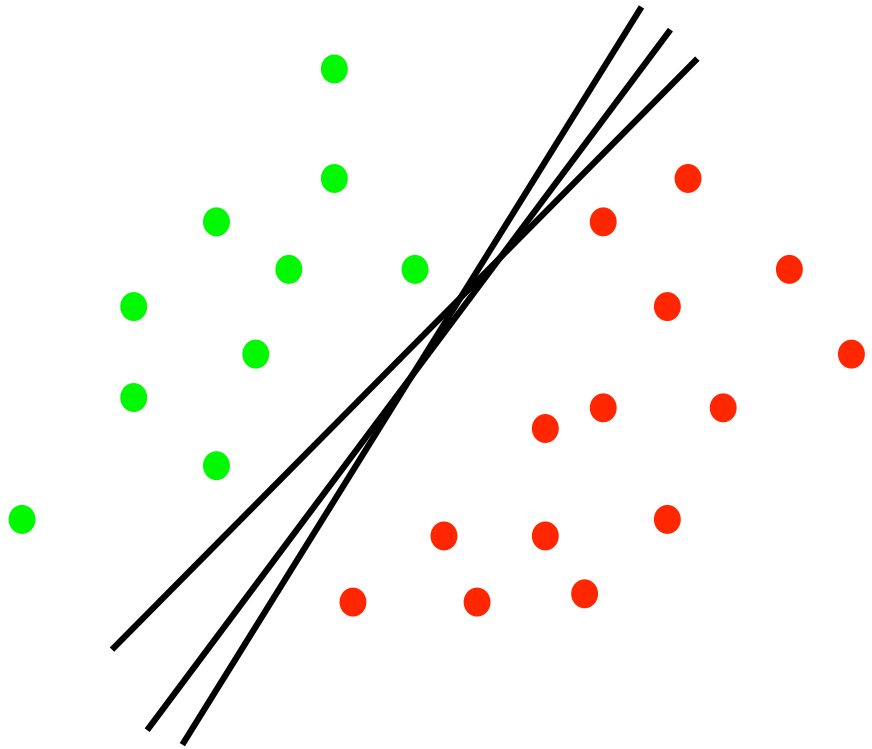




THE UNIVERSITY OF SYDNEY

Support Vector Machines

Linear classifiers



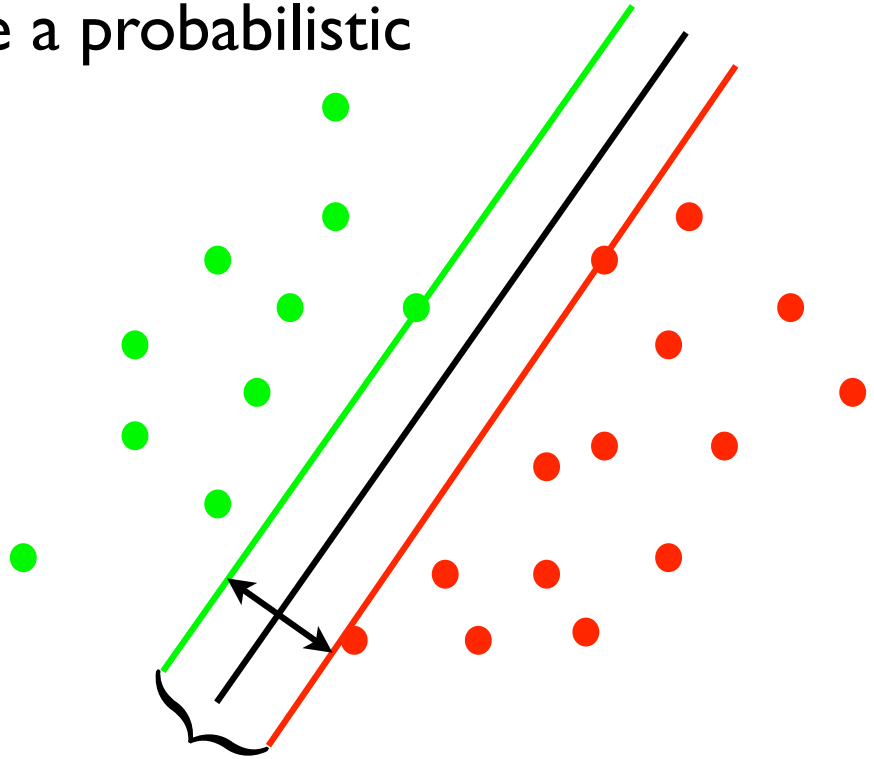
Which line is better?

Maximum margin

THE UNIVERSITY OF SYDNEY

Things to note

- The boundary is defined by only a few points (support vectors)
- Difficult to define a probabilistic explanation



Linear Basis Function Models

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

where $\phi_j(\mathbf{x})$ are known as *basis functions*.

Typically, $\phi_0(\mathbf{x}) = 1$, so that w_0 acts as a bias.

In the simplest case, we use linear basis functions :
$\phi_d(\mathbf{x}) = x_d$

Maximum margin classifiers (1)

Given a training set

Inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N$

Targets

$t_1, \ldots, t_N$ where $t_n \in \{-1, 1\}$

Linear classifier

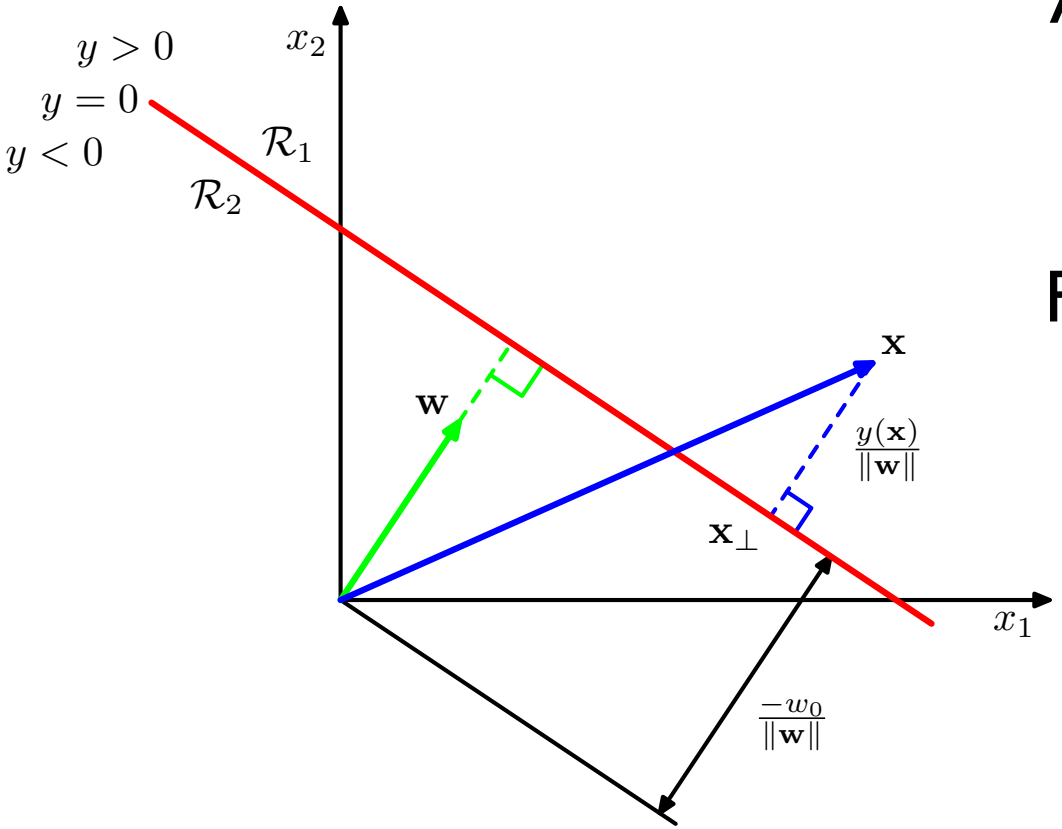
$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}) + b$$

Output sign of $y(\mathbf{x})$

so that $t_n y(\mathbf{x}_n) > 0$

for all points in the training set

Maximum margin classifiers(2)



Assuming $\phi(\mathbf{x}) = \mathbf{x}$

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

For points in the boundary

$$y(\mathbf{x}) = 0 \quad \text{and}$$

$$\frac{\mathbf{w}^{\mathrm{T}}\mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$

For arbitrary $\mathbf{x}$

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|} \quad \text{and} \quad r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}$$

Maximum margin classifiers(3)

THE UNIVERSITY OF SYDNEY

For all points correctly classified $t_n y(\mathbf{x}_n) > 0$, $b = \omega_0$

and $$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$$

Maximum margin is found by

$$\arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n \left[t_n \left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b \right) \right] \right\}$$

However, this is too difficult to optimise !

Maximum margin classifiers(4)

Note that if $\mathbf{w} \to \kappa\mathbf{w}$ $\quad b \to \kappa b$ $\quad t_n y(\mathbf{x}_n)/\|\mathbf{w}\|$ is unchanged

Setting $t_n \left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b\right) = 1$ for the support vectors:

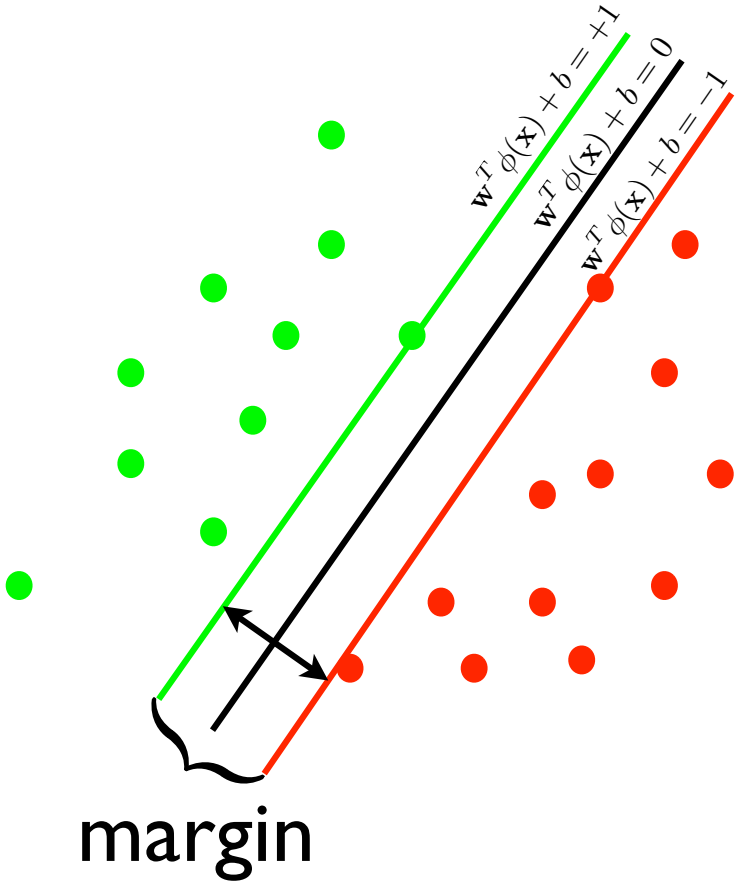
Quadratic programming

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

s.t.

$$t_n \left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b\right) \geqslant 1, \;\; n = 1, \ldots, N$$

Support Vector Machines



Quadratic programming

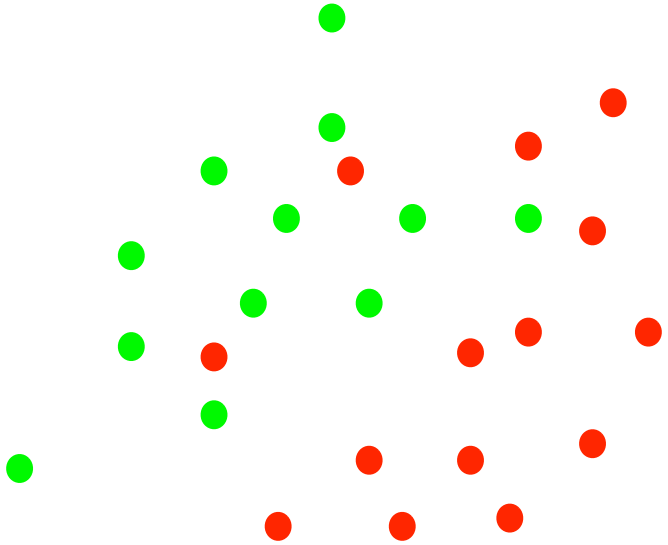
$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

s.t.

$$t_n\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)+b\right)\geqslant 1,\ \ n=1,\ldots,N$$

- Solve efficiently by quadratic programming (QP)
- Hyperplane defined by support vectors

What happens if the data is not linearly separable?



Quadratic programming

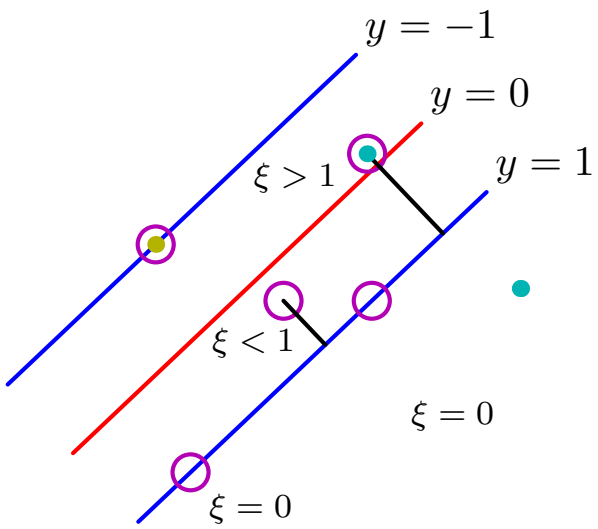
$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

s.t.

$$t_n\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)+b\right) \geqslant 1, \quad n=1,\ldots,N$$

If margin ≥ 1, don't care
If margin < 1, pay linear penalty

Soft Margin Classification



Quadratic programming

$$\arg\min_{\mathbf{w},b} C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|\mathbf{w}\|^2$$

s.t.

$$t_n y(\mathbf{x}_n) \geqslant 1 - \xi_n \qquad n = 1, \ldots, N$$

$$\xi_n \geqslant 0$$

The Dual SVM formulation

THE UNIVERSITY OF SYDNEY

Quadratic programming

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

s.t.

$$t_n\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)+b\right) \geqslant 1, \quad n=1,\ldots,N$$

Lagrangian function

$$L(\mathbf{w},b,\mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n\left\{t_n(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)+b)-1\right\}$$

setting the derivatives w.r.t. $\mathbf{w}$ and b equal to zero:

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n)$$

$$0 = \sum_{n=1}^{N} a_n t_n.$$

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n,\mathbf{x}_m)$$

Kernel trick

$$k(\mathbf{x},\mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}')$$

s.t.

$$a_n \geqslant 0, \qquad n=1,\ldots,N,$$

$$\sum_{n=1}^{N} a_n t_n = 0.$$

Kernel SVMs

THE UNIVERSITY OF SYDNEY

- Solve the QP problem

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

with respect to $\mathbf{a}$ subject to the constraints

$$\begin{aligned} a_n &\geqslant 0, \qquad n = 1, \ldots, N, \\ \sum_{n=1}^{N} a_n t_n &= 0. \end{aligned}$$

- Support vectors satisfy

$$t_n \left(\sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1$$

- For query points compute

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b.$$

Think about kernels not features

Polynomial kernel

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= \left(1 + \mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2 = (1 + x_1 z_1 + x_2 z_2)^2 \\ &= 1 + 2x_1 z_1 + 2x_2 z_2 + x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1 x_2, x_2^2)(1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, \sqrt{2}z_1 z_2, z_2^2)^{\mathrm{T}} \\ &= \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{z}). \end{aligned}$$

Gaussian kernel

$$k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma\|\mathbf{x} - \mathbf{z}\|^2)$$

Infinite dimensional feature space!

What Functions are Kernels?

THE UNIVERSITY OF SYDNEY

- For some functions $K(\mathbf{x}_i,\mathbf{x}_j)$ checking that $K(\mathbf{x}_i,\mathbf{x}_j)= \phi(\mathbf{x}_i)^{\mathrm{T}}\phi(\mathbf{x}_j)$ can be cumbersome.
- Mercer's theorem:

  Every semi-positive definite symmetric function is a kernel

- Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix:

$K=$

$K(\mathbf{x}_1,\mathbf{x}_1)$	$K(\mathbf{x}_1,\mathbf{x}_2)$	$K(\mathbf{x}_1,\mathbf{x}_3)$	…	$K(\mathbf{x}_1,\mathbf{x}_n)$
$K(\mathbf{x}_2,\mathbf{x}_1)$	$K(\mathbf{x}_2,\mathbf{x}_2)$	$K(\mathbf{x}_2,\mathbf{x}_3)$		$K(\mathbf{x}_2,\mathbf{x}_n)$
…	…	…	…	…
$K(\mathbf{x}_n,\mathbf{x}_1)$	$K(\mathbf{x}_n,\mathbf{x}_2)$	$K(\mathbf{x}_n,\mathbf{x}_3)$	…	$K(\mathbf{x}_n,\mathbf{x}_n)$

Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$, where $\boldsymbol{\phi}(\mathbf{x})$ is $\mathbf{x}$ itself
- Polynomial of power p: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^p$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$, where $\boldsymbol{\phi}(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions
- Gaussian (radial-basis function): $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}}$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$, where $\boldsymbol{\phi}(\mathbf{x})$ is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian); combination of functions for support vectors is the separator.
- Higher-dimensional space still has *intrinsic* dimensionality d (the mapping is not *onto*), but linear separators in it correspond to *non-linear* separators in original space.

Overfitting

- Huge feature space with kernels, what about overfitting???
  - Maximising margin leads to sparse set of support vectors
  - Some interesting theory says that SVMs search for simple hypothesis with large margin
  - Often robust to overfitting

Summary

- SVM advantages
  - Efficient QP learning
  - Sparse
- SVM disadvantages
  - Not probabilistic
  - Do not directly handle multi-class problems
  - QP might struggle in very large datasets

An Empirical Comparison of Supervised Learning Algorithms

Rich Caruana CARUANA@CS.CORNELL.EDU
Alexandru Niculescu-Mizil ALEXN@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

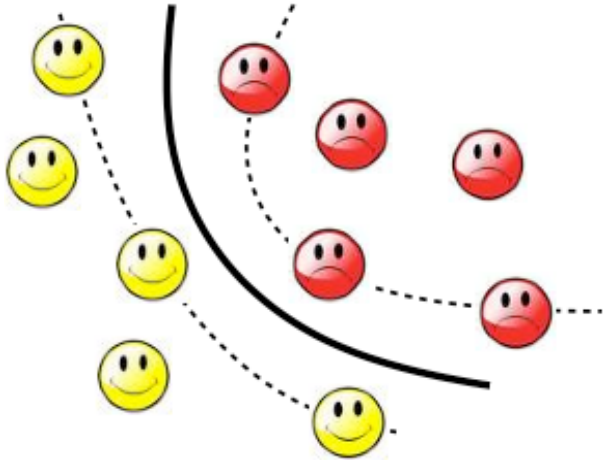
Abstract

A number of supervised learning methods have been introduced in the last decade. Unfortunately, the last comprehensive empirical evaluation of supervised learning was the Statlog Project in the early 90's. We present a large-scale empirical comparison between ten supervised learning methods: SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps. We also examine the effect that calibrating the models via Platt Scaling and Isotonic Regression has on their performance. An important aspect of our study is the use of a variety of performance criteria to evaluate the learning methods.

This paper presents results of a large-scale empirical comparison of ten supervised learning algorithms using eight performance criteria. We evaluate the performance of SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps on eleven binary classification problems using a variety of performance metrics: accuracy, F-score, Lift, ROC Area, average precision, precision/recall break-even point, squared error, and cross-entropy. For each algorithm we examine common variations, and thoroughly explore the space of parameters. For example, we compare ten decision tree styles, neural nets of many sizes, SVMs with many kernels, etc.

Because some of the performance metrics we examine interpret model predictions as probabilities and models such as SVMs are not designed to predict probabilities, we compare the performance of each algorithm both before and after calibrating its predictions with

MODEL	CAL	COVT	ADULT	LTR.P1	LTR.P2	MEDIS	SLAC	HS	MG	CALHOUS	COD	BACT	MEAN
BST-DT	PLT	**.938**	.857	**.959**	**.976**	.700	.869	**.933**	.855	**.974**	**.915**	.878*	**.896***
RF	PLT	.876	.930	.897	.941	**.810**	.907*	.884	.883	.937	.903*	.847	.892
BAG-DT	–	.878	.944*	.883	.911	.762	.898*	.856	**.898**	.948	.856	**.926**	.887*
BST-DT	ISO	.922*	.865	.901*	.969	.692*	.878	.927	.845	.965	.912*	.861	.885*
RF	–	.876	.946*	.883	.922	.785	.912*	.871	.891*	.941	.874	.824	.884
BAG-DT	PLT	.873	.931	.877	.920	.752	.885	.863	.884	.944	.865	.912*	.882
RF	ISO	.865	.934	.851	.935	.767*	**.920**	.877	.876	.933	.897*	.821	.880
BAG-DT	ISO	.867	.933	.840	.915	.749	.897	.856	.884	.940	.859	.907*	.877
SVM	PLT	.765	.886	.936	.962	.733	.866	.913*	.816	.897	.900*	.807	.862
ANN	–	.764	.884	.913	.901	.791*	.881	.932*	.859	.923	.667	.882	.854
SVM	ISO	.758	.882	.899	.954	.693*	.878	.907	.827	.897	.900*	.778	.852
ANN	PLT	.766	.872	.898	.894	.775	.871	.929*	.846	.919	.665	.871	.846
ANN	ISO	.767	.882	.821	.891	.785*	.895	.926*	.841	.915	.672	.862	.842
BST-DT	–	.874	.842	.875	.913	.523	.807	.860	.785	.933	.835	.858	.828
KNN	PLT	.819	.785	.920	.937	.626	.777	.803	.844	.827	.774	.855	.815
KNN	–	.807	.780	.912	.936	.598	.800	.801	.853	.827	.748	.852	.810
KNN	ISO	.814	.784	.879	.935	.633	.791	.794	.832	.824	.777	.833	.809
BST-STMP	PLT	.644	**.949**	.767	.688	.723	.806	.800	.862	.923	.622	.915*	.791
SVM	–	.696	.819	.731	.860	.600	.859	.788	.776	.833	.864	.763	.781
BST-STMP	ISO	.639	.941	.700	.681	.711	.807	.793	.862	.912	.632	.902*	.780
BST-STMP	–	.605	.865	.540	.615	.624	.779	.683	.799	.817	.581	.906*	.710
DT	ISO	.671	.869	.729	.760	.424	.777	.622	.815	.832	.415	.884	.709
DT	–	.652	.872	.723	.763	.449	.769	.609	.829	.831	.389	.899*	.708
DT	PLT	.661	.863	.734	.756	.416	.779	.607	.822	.826	.407	.890*	.706
LR	–	.625	.886	.195	.448	.777*	.852	.675	.849	.838	.647	.905*	.700
LR	ISO	.616	.881	.229	.440	.763*	.834	.659	.827	.833	.636	.889*	.692
LR	PLT	.610	.870	.185	.446	.738	.835	.667	.823	.832	.633	.895	.685
NB	ISO	.574	.904	.674	.557	.709	.724	.205	.687	.758	.633	.770	.654
NB	PLT	.572	.892	.648	.561	.694	.732	.213	.690	.755	.632	.756	.650
NB	–	.552	.843	.534	.556	.011	.714	-.654	.655	.759	.636	.688	.481



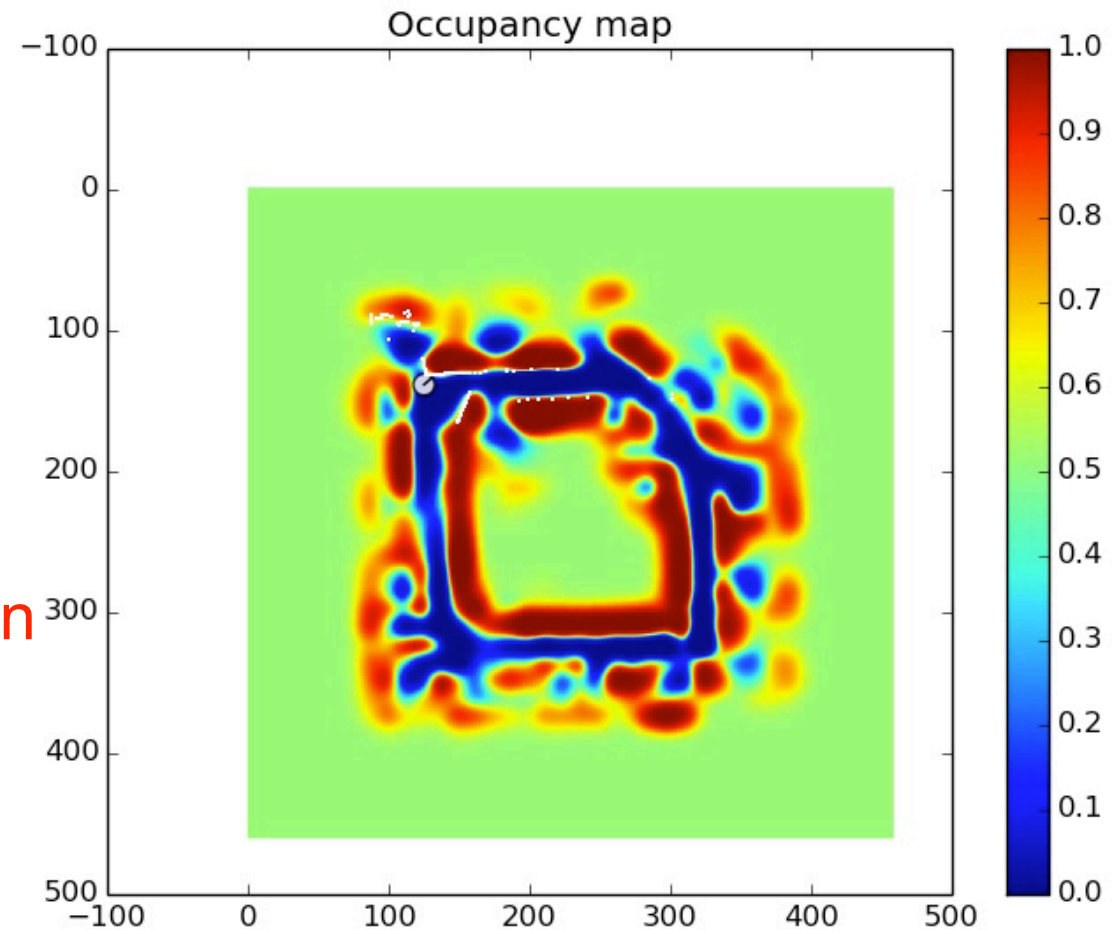
Research Topics

Hilbert Maps

THE UNIVERSITY OF SYDNEY

+ Continuous, probabilistic
+ Fast, scalable, constant cost
+ Immensely parallelisable
+ Naturally incremental
+ One pass over the data
+ Super simple to implement

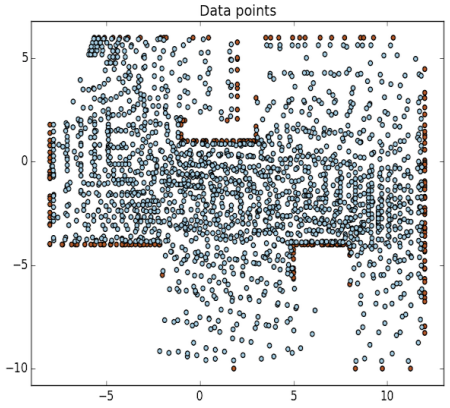
Key idea: Project the data in an *approximate* RK Hilbert space and learn a linear classifier



Intel Labs dataset

F. Ramos, L. Ott. Hilbert maps: continuous occupancy mapping with stochastic gradient descent. In Proceedings of the Robotics: Science and Systems (RSS) XI, 2015.

Overview



Data

Regularised log-likelihood

$$NLL(\mathbf{w}) = \sum_{i=1}^{N} \log\left(1 + \exp\left(-y_i \mathbf{w}^T \Phi(\mathbf{x}_i)\right)\right) + R(\mathbf{w})$$

Logistic Regression

$$p(y_* = -1 | \mathbf{x}_*, \mathbf{w}) = \frac{1}{1 + \exp(\mathbf{w}^T \Phi(\mathbf{x}_*))}$$

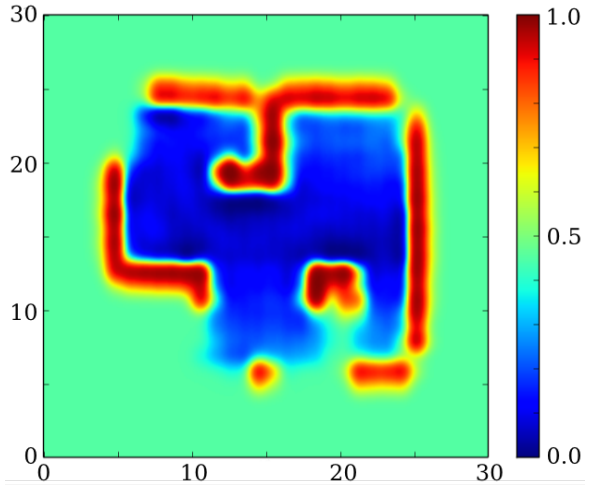
Feature Mapping

$$\Phi(\mathbf{x}) = \frac{1}{\sqrt{n}} \left[e^{-i\mathbf{s}_1 \cdot \mathbf{x}}, \ldots, e^{-i\mathbf{s}_n \cdot \mathbf{x}}\right]$$

Key: approximates RKHS with a basis function

Stochastic gradient

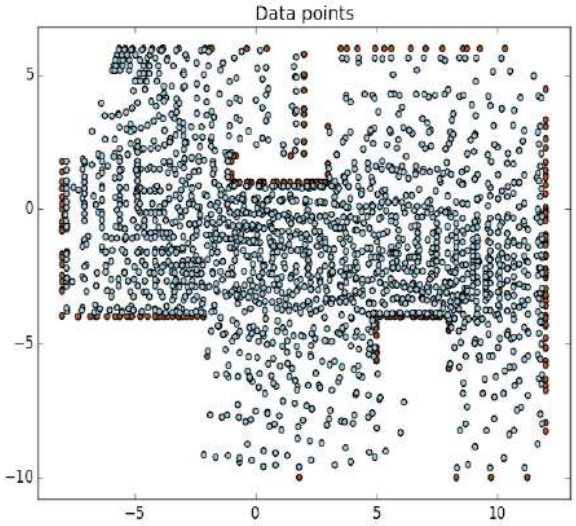
$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t A_t^{-1} \frac{\partial}{\partial \mathbf{w}} NLL(\mathbf{w})$$



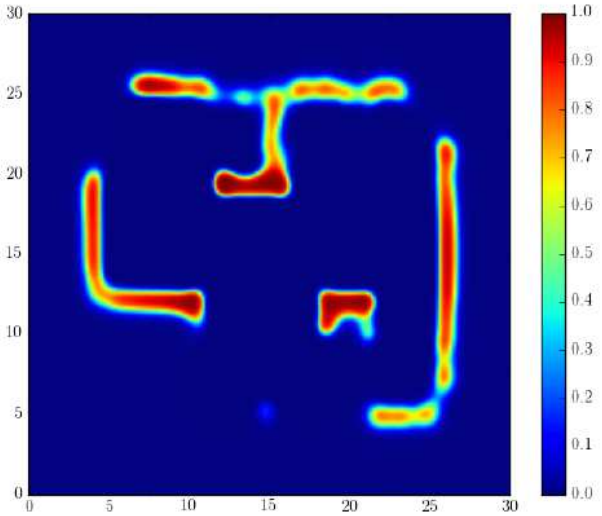
Hilbert Maps

HM with LR vs SVM

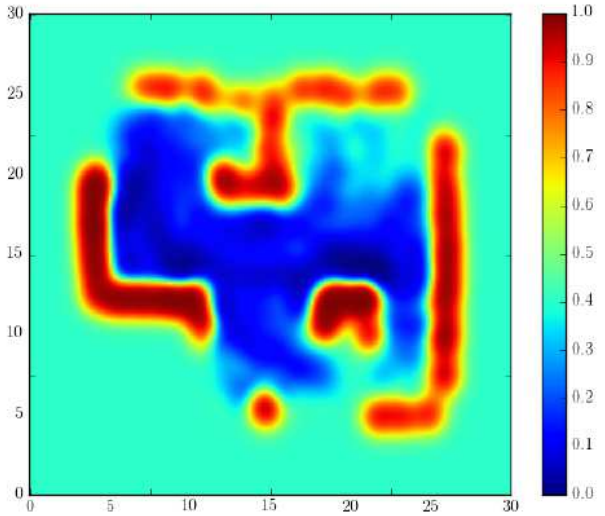
THE UNIVERSITY OF SYDNEY



Data



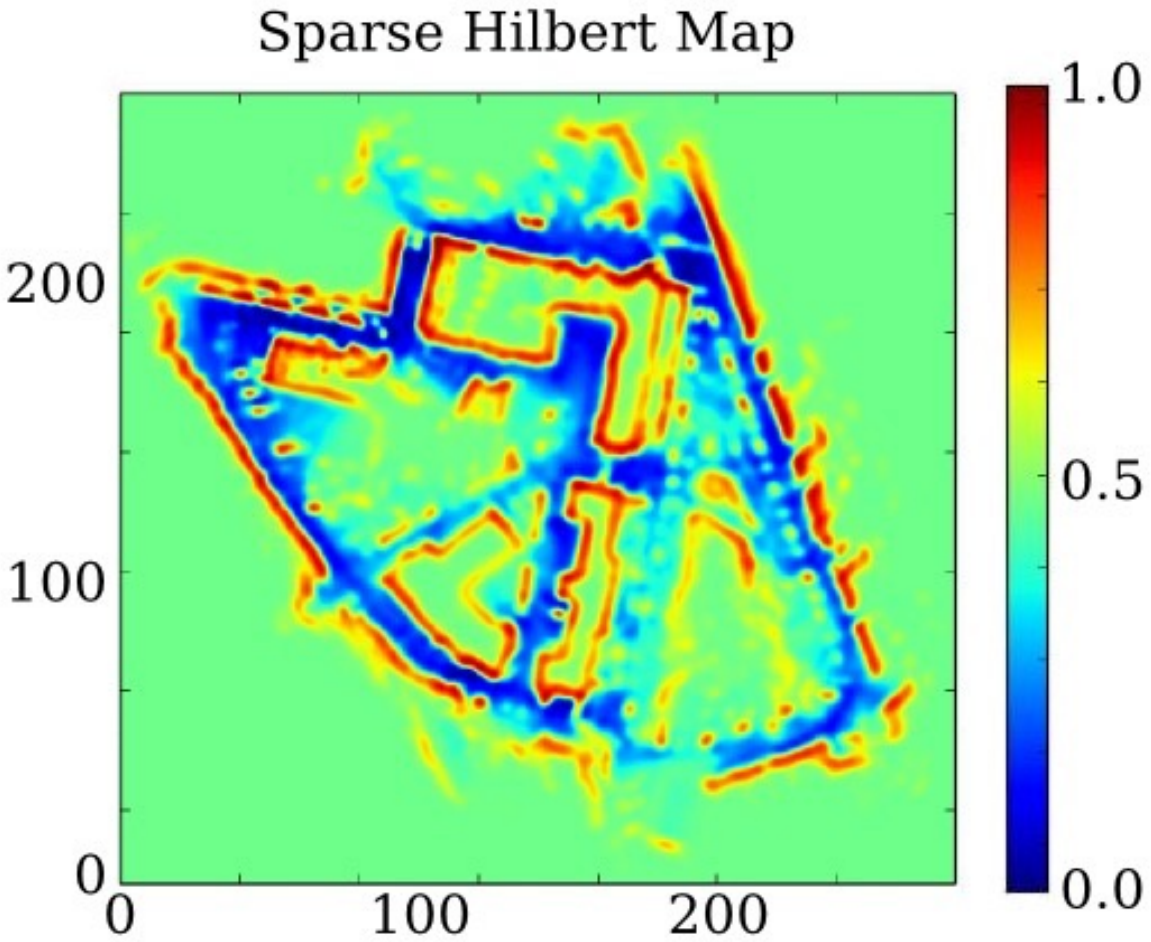
SVMs



Logistic Regression

Freiburg Campus Dataset

THE UNIVERSITY OF SYDNEY



Freiburg Campus Dataset