TDA 231
Support Vector Machines and Kernel methods

Devdatt Dubhashi
dubhashi@chalmers.se

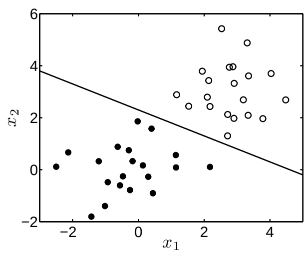
Dept. of Computer Science and Engg.
Chalmers University

The margin

- We have seen several algorithms where we find the parameters that optimise something:
  - Minimise the loss.
  - Maximise the likelihood.
  - Maximise the posterior (MAP).
- The Support Vector Machine (SVM) is no different:
- It finds the decision boundary that maximises the margin.

Some data

- We'll 'think' in 2-dimensions.



SVM is a binary classifier. N data points, each with attributes $\mathbf{x} = [x_1, x_2]^\mathsf{T}$ and target $t = \pm 1$

- A linear decision boundary can be represented as a straight line:

$$\mathbf{w}^\mathsf{T}\mathbf{x} + b = 0$$

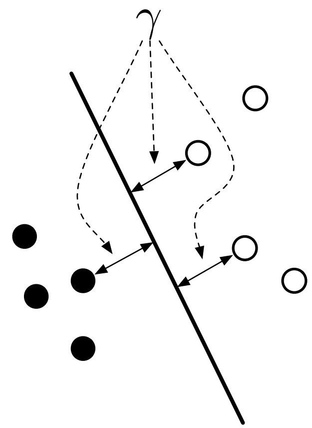
- Our task is to find $\mathbf{w}$ and b
- Once we have these, classification is easy:

$$\begin{aligned} \mathbf{w}^\mathsf{T}\mathbf{x}_{\text{new}} + b > 0 &\quad : \quad t_{\text{new}} = 1 \\ \mathbf{w}^\mathsf{T}\mathbf{x}_{\text{new}} + b < 0 &\quad : \quad t_{\text{new}} = -1 \end{aligned}$$

- i.e. $t_{\text{new}} = \text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x}_{\text{new}} + b)$

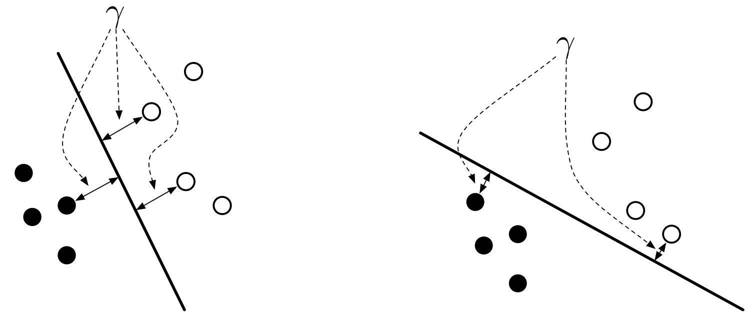
The margin

- How do we choose $\mathbf{w}$ and b?
- Need a quantity to optimise!
- Use the **margin**, γ
- Maximise it!



Perpendicular distance from the decision boundary to the closest points on each side.

Why maximise the margin?



- Maximum margin decision boundary (left) seems to better reflect the data characteristics than other boundary (right).
- Note how margin is much smaller on right and closest points have changed.
- There is going to be one 'best' boundary (w.r.t margin)
- Statistical theory justifying the choice.

Computing the margin

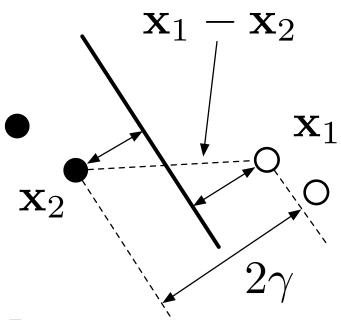
$$2\gamma = \frac{1}{||\mathbf{w}||}\mathbf{w}^{\mathsf{T}}(\mathbf{x}_1 - \mathbf{x}_2)$$

Fix the scale such that:

$$\begin{aligned} \mathbf{w}^{\mathsf{T}}\mathbf{x}_1 + b &= 1 \\ \mathbf{w}^{\mathsf{T}}\mathbf{x}_2 + b &= -1 \end{aligned}$$

Therefore:

$$\begin{aligned} (\mathbf{w}^{\mathsf{T}}\mathbf{x}_1 + b) - (\mathbf{w}^{\mathsf{T}}\mathbf{x}_2 + b) &= \\ \mathbf{w}^{\mathsf{T}}(\mathbf{x}_1 - \mathbf{x}_2) &= \\ \gamma = \frac{1}{||\mathbf{w}||} \end{aligned}$$



Maximising the margin

- We want to maximise $\gamma = \frac{1}{||\mathbf{w}||}$
- Equivalent to minimising $||\mathbf{w}||$
- Equivalent to minimising $\frac{1}{2}||\mathbf{w}||^2 = \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w}$
- There are some constraints:
  - For $\mathbf{x}_n$ with $t_n = 1$: $\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b \geq 1$
  - For $\mathbf{x}_n$ with $t_n = -1$: $\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b \leq -1$
- Which can be expressed more neatly as:

$$t_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b) \geq 1$$

- (This is why we use $t_n = \pm 1$ and not $t_n = \{0, 1\}$.)

Maximising the margin

- We have the following optimisation problem:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w}$$

$$\text{Subject to: } t_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b) \geq 1$$

- Can put the constraints into the minimisation using Lagrange multipliers:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} - \sum_{n=1}^{N} \alpha_n(t_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b) - 1)$$

$$\text{Subject to: } \alpha_n \geq 0$$

What now?

- Let's think about what happens at the solution (we'll see why...)
- We know that $\frac{\partial}{\partial \mathbf{w}} = 0$ and $\frac{\partial}{\partial b} = 0$.

$$\frac{\partial}{\partial \mathbf{w}} = \mathbf{w} - \sum_n \alpha_n t_n \mathbf{x}_n = 0$$

$$\frac{\partial}{\partial b} = -\sum_n \alpha_n t_n = 0$$

- From which we can infer that:

$$\mathbf{w} = \sum_n \alpha_n t_n \mathbf{x}_n$$

$$\sum_n \alpha_n t_n = 0$$

- Substitute these back into our optimisation problem:

$$\frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} - \sum_n \alpha_n (t_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b) - 1)$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$= \sum_n \alpha_n - \frac{1}{2}\sum_{n,m} \alpha_n \alpha_m t_n t_m \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_m$$

- Instead of minimising the previous expression, we can maximise this one (for reasons we won't go into).
- Subject to:

$$\alpha_n \geq 0$$

$$\sum_n \alpha_n t_n = 0$$

- Decision function was $\text{sign}(\mathbf{w}^{\mathsf{T}}\mathbf{x}_{\text{new}} + b)$ and is now:

$$t_{\text{new}} = \text{sign}\left(\sum_{n=1}^{N} \alpha_n t_n \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_{\text{new}} + b\right)$$

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

Primal and Dual

Primal

$$\underset{\mathbf{w}}{\text{argmin}}\ \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w}$$

Subject to: $t_n(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b) \geq 1$

Dual

$$\underset{\boldsymbol{\alpha}}{\text{argmax}} \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n.m=1}^{N} \alpha_n \alpha_m t_n t_m \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_m$$

$$\text{subject to} \quad \sum_{n=1}^{N} \alpha_n t_n = 0, \quad \alpha_n \geq 0$$

- This is a standard optimisation problem (quadratic programming)
- Has a single, global solution. This is very useful!
- Many algorithms around to solve it.

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

Using Dual for Prediction

$$\underset{\boldsymbol{\alpha}}{\text{argmax}} \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n.m=1}^{N} \alpha_n \alpha_m t_n t_m \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_m$$

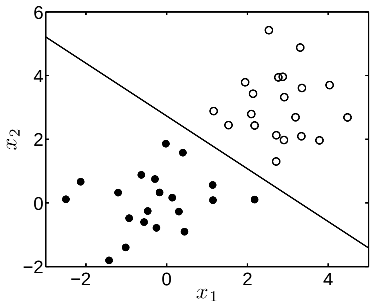
$$\text{subject to} \quad \sum_{n=1}^{N} \alpha_n t_n = 0, \quad \alpha_n \geq 0$$

- This is a standard optimisation problem (quadratic programming)
- Has a single, global solution. This is very useful!
- Many algorithms around to solve it.
- e.g. `quadprog` in Matlab...
- Once we have α_n:

$$t_{\text{new}} = \text{sign}\left(\sum_{n=1}^{N} \alpha_n t_n \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_{\text{new}} + b\right)$$

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

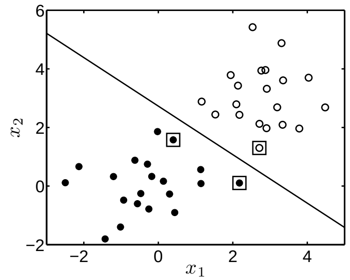
Optimal boundary



- Optimisation gives us $\alpha_1, \ldots, \alpha_N$
- Compute $\mathbf{w} = \sum_n \alpha_n t_n \mathbf{x}_n$
- Compute $b = t_n - \mathbf{w}^\mathsf{T}\mathbf{x}$ (for one of the closest points)
  - Recall that we defined $\mathbf{w}^\mathsf{T}\mathbf{x} + b = \pm 1 = t_n$ for closest points.
- Plot $\mathbf{w}^\mathsf{T}\mathbf{x} + b = 0$

Support Vectors

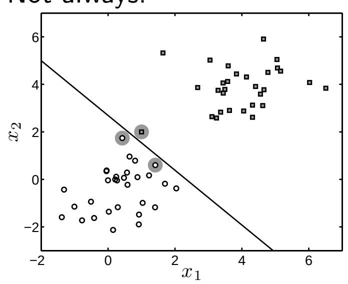
- At the optimum, only 3 non-zero α values (squares).



- $t_{\text{new}} = \text{sign}\left(\sum_n \alpha_n t_n \mathbf{x}_n^\mathsf{T}\mathbf{x}_{\text{new}} + b\right)$
- Predictions only depend on these data-points!
- We knew that – margin is only a function of closest points.
- These are called Support Vectors
- Normally a small proportion of the data:
  - Solution is sparse.

Is sparseness good?

- Not always:



- Why does this happen?

$$t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) \geq 1$$

- All points must be on correct side of boundary.
- This is a hard margin

Soft margin

- We can relax the constraints:

$$t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) \geq 1 - \xi_n, \;\; \xi_n \geq 0$$

- Our optimisation becomes:

$$\underset{\mathbf{w}}{\text{argmin}} \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} + C\sum_{n=1}^{N}\xi_n$$
$$\text{subject to} \quad t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) \geq 1 - \xi_n$$

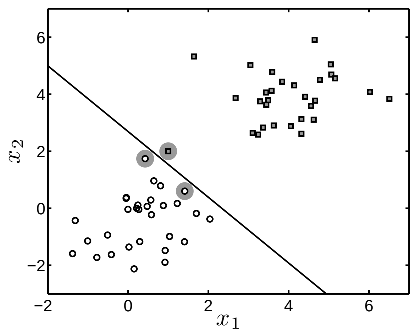
- And when we add Lagrange etc:

$$\underset{\boldsymbol{\alpha}}{\text{argmax}} \sum_{n=1}^{N}\alpha_n - \frac{1}{2}\sum_{n.m=1}^{N}\alpha_n\alpha_m t_n t_m \mathbf{x}_n^\mathsf{T}\mathbf{x}_m$$
$$\text{subject to} \quad \sum_{n=1}^{N}\alpha_n t_n = 0, \quad 0 \leq \alpha_n \leq C$$

- The **only** change is an upper-bound on α_n!

Soft margins

- Here's our problematic data again:



- α_n for the 'bad' square is 3.5.
- So, if we set $C < 3.5$, we should see this point having less influence and the boundary moving to somewhere more sensible...

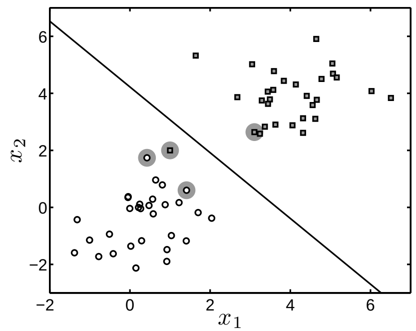
Introduction

D. Dubhashi

The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

Soft margins

- Try $C = 1$



- We have an extra support vector.
- And a better decision boundary.

Introduction

D. Dubhashi

The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

Soft margins

- The choice of C is very important.
- Too high and we *over-fit* to noise.
- Too low and we *underfit*
  - ...and lose any sparsity.
- Choose it using cross-validation.

Introduction

D. Dubhashi

The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

SVMs – some observations

- In our example, we started with 3 parameters:

$$\mathbf{w} = [w_1, w_2]^{\mathsf{T}}, \quad b$$

- In general: D+1.
- We now have N: $\alpha_1, \ldots, \alpha_N$
- Sounds harder?
- Depends on data dimensionality:
  - Typical Microarray dataset:
  - $D \sim 3000$, $N \sim 30$.
  - In some cases $N \ll D$

Inner products

- Here's the optimisation problem:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_n \alpha_n - \frac{1}{2}\sum_{n,m} \alpha_n \alpha_m t_n t_m \mathbf{x}_n^\mathsf{T} \mathbf{x}_m$$

- Here's the decision function:

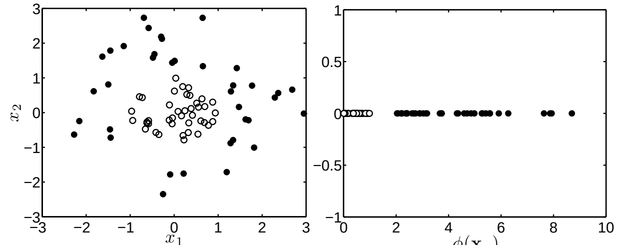
$$t_{\text{new}} = \operatorname{sign}\left(\sum_n \alpha_n t_n \mathbf{x}_n^\mathsf{T} \mathbf{x}_{\text{new}} + b\right)$$

- Data ($\mathbf{x}_n, \mathbf{x}_m, \mathbf{x}_{\text{new}}$, etc) only appears as inner (dot) products:

$$\mathbf{x}_n^\mathsf{T}\mathbf{x}_m, \;\; \mathbf{x}_n^\mathsf{T}\mathbf{x}_{\text{new}}, \text{etc}$$

Projections

- Our SVM can find linear decision boundaries.
- What if the data requires something nonlinear?



- We can transform the data e.g.:

$$\phi(\mathbf{x}_n) = x_{n1}^2 + x_{n2}^2$$

- So that it can be separated with a straight line.
- And use $\phi(\mathbf{x}_n)$ instead of $\mathbf{x}_n$ in our optimisation.

Projections

- Our optimisation is now:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_n \alpha_n - \frac{1}{2}\sum_{n,m} \alpha_n \alpha_m t_n t_m \phi(\mathbf{x}_n)^\mathsf{T} \phi(\mathbf{x}_m)$$

- And predictions:

$$t_{\text{new}} = \operatorname{sign}\left(\sum_n \alpha_n t_n \phi(\mathbf{x}_n)^\mathsf{T} \phi(\mathbf{x}_{\text{new}}) + b\right)$$

- In this case:

$$\phi(\mathbf{x}_n^\mathsf{T})\phi(\mathbf{x}_m) = (x_{n1}^2 + x_{n2}^2)(x_{m1}^2 + x_{m2}^2) = k(\mathbf{x}_n, \mathbf{x}_m)$$

- We can think of the dot product in the projected space as a function of the original data.

Projections

- We needn't directly think of projections at all.
- Can just think of functions $k(\mathbf{x}_n, \mathbf{x}_m)$ that *are dot products in some space*.
- Called *kernel* functions.
- Don't ever need to actually project the data – just use the kernel function to compute what the dot product would be if we did project.
- Optimisation task:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_n \alpha_n - \frac{1}{2}\sum_{n,m} \alpha_n \alpha_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

- Predictions:

$$t_{\text{new}} = \operatorname{sign}\left(\sum_n \alpha_n t_n k(\mathbf{x}_n, \mathbf{x}_{\text{new}}) + b\right)$$

Kernels

- Plenty of off-the-shelf kernels that we can use:
- Linear:
$$k(\mathbf{x}_n, \mathbf{x}_m) = \mathbf{x}_n^\mathsf{T}\mathbf{x}_m$$
- Gaussian:
$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left\{-\beta(\mathbf{x}_n - \mathbf{x}_m)^\mathsf{T}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$
- Polynomial:
$$k(\mathbf{x}_n, \mathbf{x}_m) = (1 + \mathbf{x}_n^\mathsf{T}\mathbf{x}_m)^\beta$$
- These all correspond to $\phi(\mathbf{x}_n)^\mathsf{T}\phi(\mathbf{x}_m)$ for some transformation $\phi(\mathbf{x}_n)$.
- Don't know what the projections $\phi(\mathbf{x}_n)$ are – don't need to know!

Kernels

- Our algorithm is still only finding linear boundaries....
- ...but we're finding linear boundaries in some other space.
- The optimisation is just as simple, regardless of the kernel choice.
  - Still a quadratic program.
  - Still a single, global optimum.
- We can find very complex decision boundaries with a linear algorithm!

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

A technical point

- Our decision boundary was defined as $\mathbf{w}^\mathsf{T}\mathbf{x} + b = 0$.
- Now, $\mathbf{w}$ is defined as:
$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n t_n \phi(\mathbf{x}_n)$$
- We don't know $\phi(\mathbf{x}_n)$.
- We **only know** $\phi(\mathbf{x}_n)^\mathsf{T}\phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$
- So, we can't compute $\mathbf{w}$ or the boundary!
- But we can evaluate the predictions on a grid of $\mathbf{x}_{\text{new}}$ and use Matlab to draw a contour:
$$\sum_{n=1}^{N} \alpha_n t_n k(\mathbf{x}_n, \mathbf{x}_{\text{new}}) + b$$

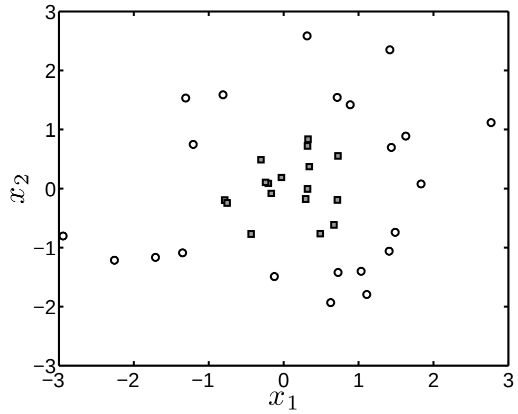
Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

Aside: kernelising other algorithms

- **Many** algorithms can be kernelised.
  - Any that can be written with data only appearing as inner products.
- Simple algorithms can be used to solve very complex problems!
- Class exercise:
  - KNN requires the distance between $\mathbf{x}_{\text{new}}$ and each $\mathbf{x}_n$:
$$(\mathbf{x}_{\text{new}} - \mathbf{x}_n)^\mathsf{T}(\mathbf{x}_{\text{new}} - \mathbf{x}_n)$$
  - Can we kernelise it?

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

Example – nonlinear data

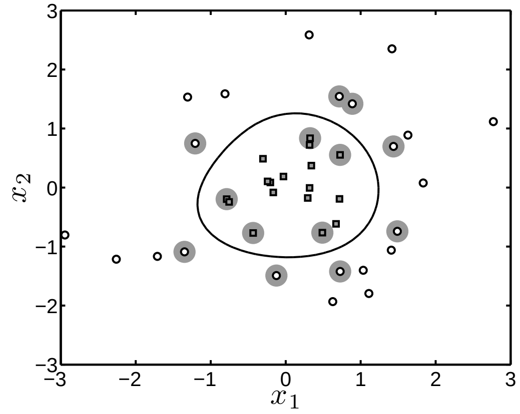


- We'll use a Gaussian kernel:

$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left\{-\beta(\mathbf{x}_n - \mathbf{x}_m)^{\mathsf{T}}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$

- And vary β ($C = 10$).

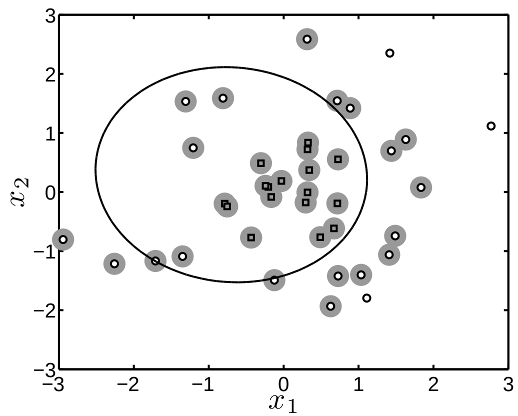
Examples



- $\beta = 1$.

$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left\{-\beta(\mathbf{x}_n - \mathbf{x}_m)^{\mathsf{T}}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$

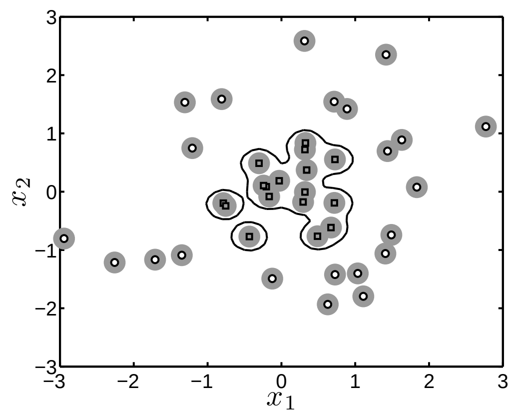
Examples



- $\beta = 0.01$.

$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left\{-\beta(\mathbf{x}_n - \mathbf{x}_m)^{\mathsf{T}}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$

Examples



- $\beta = 50$.

$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left\{-\beta(\mathbf{x}_n - \mathbf{x}_m)^{\mathsf{T}}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$

The Gaussian kernel

- β controls the *complexity* of the decision boundaries.
- $\beta = 0.01$ was too simple:
  - Not flexible enough to surround just the square class.
- $\beta = 50$ was too complex:
  - *Memorises* the data.
- $\beta = 1$ was about right.
- Neither $\beta = 50$ or $\beta = 0.01$ will *generalise* well.
- Both are also non-sparse (lots of support vectors).

Choosing kernel function, parameters and C

- Kernel function and parameter choice is data dependent.
- Easy to overfit.
- Need to set C too
- C and β are *linked*
  - C too high – overfitting.
  - C too low – underfitting.
- Cross-validation!
- Search over β and C
  - SVM scales with N^3 (naive implementation)
  - For large N, cross-validation over many C and β values is infeasible.

Summary - SVMs

- Described a classifier that is optimised by maximising the *margin*.
- Did some re-arranging to turn it into a quadratic programming problem.
- Saw that data only appear as inner products.
- Introduced the idea of kernels.
- Can fit a linear boundary in some other space without explicitly projecting.
- Loosened the SVM constraints to allow points on the wrong side of boundary.
- Other algorithms can be kernelised...we'll see a clustering one in the future.

Performance evaluation

- We've seen 4 classification algorithms.
- How do we choose?
  - Which algorithm?
  - Which parameters?
- Need performance indicators.
- We'll cover:
  - 0/1 loss.
  - ROC analysis (sensitivity and specificity)
  - Confusion matrices

0/1 loss

- 0/1 loss: proportion of times classifier is wrong.
- Consider a set of predictions $t_1, \ldots, t_N$ and a set of true labels $t_1^*, \ldots, t_N^*$.
- Mean loss is defined as:

$$\frac{1}{N}\sum_{n=1}^{N}\delta(t_n \neq t_n^*)$$

- ($\delta(a)$ is 1 if a is true and 0 otherwise)
- Advantages:
  - Can do binary or multiclass classification.
  - Simple to compute.
  - Single value.

0/1 loss

Disadvantage: Doesn't take into account class imbalance:

- We're building a classifier to detect a rare disease.
- Assume only 1% of population is diseased.
- Diseased: $t = 1$
- Healthy: $t = 0$
- What if we always predict healthy? ($t = 0$)
- Accuracy 99%
- But classifier is rubbish!

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

Sensitivity and specificity

- We'll stick with our disease example.
- Need to define 4 quantities. The numbers of:
- **True positives (TP)** – the number of objects with $t_n^* = 1$ that are classified as $t_n = 1$ (diseased people diagnosed as diseased).
- **True negatives (TN)** – the number of objects with $t_n^* = 0$ that are classified as $t_n = 0$ (healthy people diagnosed as healthy).
- **False positives (FP)** – the number of objects with $t_n^* = 0$ that are classified as $t_n = 1$ (healthy people diagnosed as diseased).
- **False negatives (FN)** – the number of objects with $t_n^* = 1$ that are classified as $t_n = 0$ (diseased people diagnosed as healthy).

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

Sensitivity

$$S_e = \frac{TP}{TP + FN}$$

- The proportion of diseased people that we classify as diseased.
- The higher the better.
- In our example, $S_e = 0$.

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

Specificity

$$S_p = \frac{TN}{TN + FP}$$

- The proportion of healthy people that we classify as healthy.
- The higher the better.
- In our example, $S_p = 1$.

Optimising sensitivity and specificity

- We would like both to be as high as possible.
- Often increasing one will decrease the other.
- Balance will depend on application:
- e.g. diagnosis:
  - We can probably tolerate a decrease in specificity (healthy people diagnosed as diseased)....
  - ...if it gives us an increase in sensitivity (getting diseased people right).

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

ROC analysis

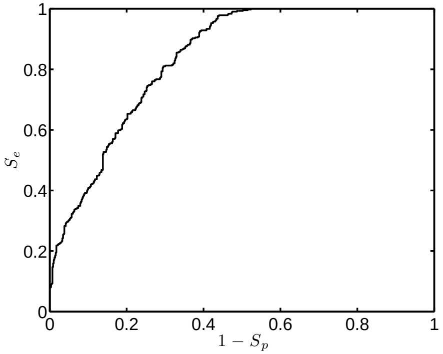
- Many classification algorithms involve setting a threshold.
- e.g. SVM:

$$t_{\text{new}} = \text{sign}\left(\sum_{n=1}^{N} t_n \alpha_n k(\mathbf{x}_n, \mathbf{x}_{\text{new}}) + b\right)$$

- Implies a threshold of zero (sign function)
- However, we could use any threshold we like....
- The *Receiver Operating Characteristic (ROC) curve* shows how S_e and $1 - S_p$ vary as the threshold changes.

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

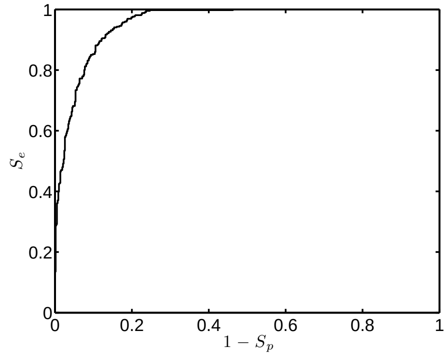
ROC curve



- SVM for nonlinear data (earlier in lecture) with $\beta = 50$.
- Each point is a threshold value.
  - Bottom left – everything classified as 0 (-1 in SVM)
  - Top right – everything classified as 1.
- Goal: get the curve to the top left corner – perfect classification ($S_e = 1, S_p = 1$).

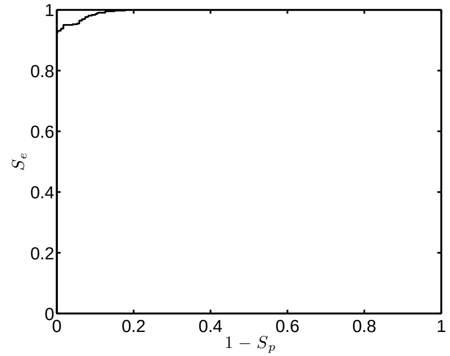
Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

ROC curve



- SVM for nonlinear data (earlier in lecture) with $\beta = 0.01$.
- Better than $\beta = 50$
  - Closer to top left corner.

ROC curve



- SVM for nonlinear data (earlier in lecture) with $\beta = 1$.
- Better still.

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

AUC

- We can quantify performance by computing the *Area Under the ROC Curve (AUC)*
- The higher this value, the better.
  - $\beta = 50$: AUC=0.8348
  - $\beta = 0.01$: AUC= 0.9551
  - $\beta = 1$: AUC=0.9936
- AUC is generally a safer measure than 0/1 loss.

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

Confusion matrices

The quantities we used to compute S_e and S_p can be neatly summarised in a table:

		True class	
		1	0
Predicted class	1	TP	FP
	0	FN	TN

- This is known as a *confusion matrix*
- It is particularly useful for multi-class classification.
- Tells us where the mistakes are being made.
- Note that normalising columns gives us S_e and S_p

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

Confusion matrices – example

- 20 newsgroups data.
- Thousands of documents from 20 classes (newsgroups)
- Use a Naive Bayes classifier ($\approx$ 50000 dimensions (words)!)
  - Details in book Chapter.
- $\approx$ 7000 independent test documents.
- Summarise results in 20×20 confusion matrix:

True class

Predicted class	...	10	11	12	13	14	15	16	18	18	19	20
1	...	4	2	0	2	10	4	7	1	12	7	47
2	...	0	0	4	18	7	8	2	0	1	1	3
3	...	0	0	1	0	1	0	1	0	0	0	0
4	...	1	0	1	28	3	0	0	0	0	0	0
⋮												
16	...	3	2	2	5	17	4	376	3	7	2	**68**
17	...	1	0	9	0	3	1	3	325	3	**95**	19
18	...	2	1	0	2	6	2	1	2	325	4	5
19	...	8	4	8	0	10	21	1	16	19	**185**	7
20	...	0	0	1	0	1	1	2	4	0	1	**92**

- Algorithm is getting ‘confused’ between classes 20 and 16, and 19 and 17.
  - 17: talk.politics.guns
  - 19: talk.politics.misc
  - 16: talk.religion.misc
  - 20: soc.religion.christian
- Maybe these should be just one class?
- Maybe we need more data in these classes?
- Confusion matrix helps us direct our efforts to improving the classifier.

Introduction
D. Dubhashi
The margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary
Assessing classifier performance
0/1 loss
ROC analysis
Confusion matrices
Summary

Summary

- SVM: a kernel classifier.
- Linear classifier – (possibly) nonlinear data transformation.
- Introduced two different performance measures:
  - 0/1 loss
  - ROC/AUC
- Introduced confusion matrices – a way of assessing the performance of a multi-class classifier.