Machine Learning - MT 2016
9 & 10. Support Vector Machines

Varun Kanade

University of Oxford
November 7 & 9, 2016

Announcements

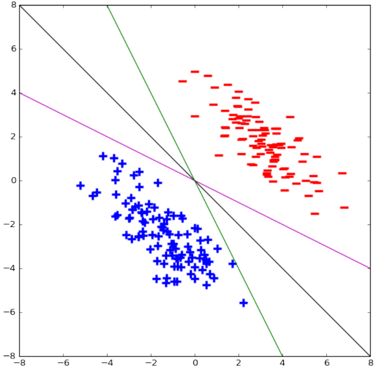
- Problem Sheet 3 due this Friday by noon
- Practical 2 next week
- (Optional) Reading a paper

Outline

This week we'll discuss classification using support vector machines.

- No clear probabilistic interpretation
- Maximum Margin Formulation
- Optimisation problem using Hinge Loss
- Dual Formulation
- Kernel Methods for non-linear classification

Binary Classification

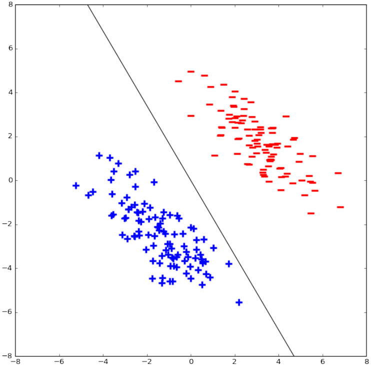


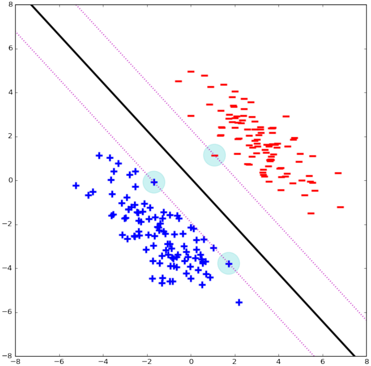
Goal: Find a linear separator

Data is linearly separable if there exists a linear separator that classifies all points correctly

Which separator should be picked?

Maximum Margin Principle



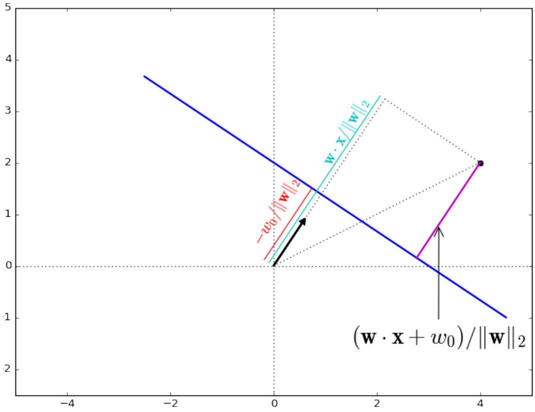


Maximise the distance of the closest point from the decision boundary

Points that are closest to the decision boundary are support vectors

Geometry Review

Given a hyperplane: $H \equiv \mathbf{w} \cdot \mathbf{x} + w_0 = 0$ and a point $\mathbf{x} \in \mathbb{R}^D$, how far is $\mathbf{x}$ from H?



Geometry Review

- Consider the hyperplane: $H \equiv \mathbf{w} \cdot \mathbf{x} + w_0 = 0$
- The distance of point $\mathbf{x}$ from H is given by:

$$\frac{|\mathbf{w} \cdot \mathbf{x} + w_0|}{\|\mathbf{w}\|_2}$$

- All points on one side of the hyperplane satisfy

$$\mathbf{w} \cdot \mathbf{x} + w_0 > 0$$

  and points on the other side satisfy

$$\mathbf{w} \cdot \mathbf{x} + w_0 < 0$$

SVM Formulation : Separable Case

Let $\mathcal{D} = \langle (\mathbf{x}_i, y_i) \rangle_{i=1}^{N}$ with $y_i \in \{-1, 1\}$

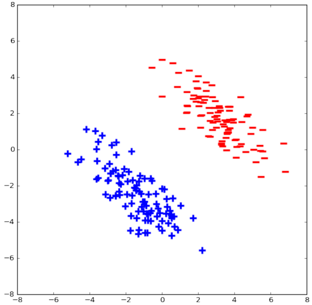
Ignoring the max-margin for now

Find $\mathbf{w}$, w_0, such that

$y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1$

for $i = 1, \ldots, N$

This is simply a linear program!



For any $\mathbf{w}$, w_0 satisfying the above, the smallest margin is at least $\frac{1}{\|\mathbf{w}\|_2}$

In order to obtain a maximum-margin condition, we minimise $\|\mathbf{w}\|_2^2$ subject to the above constraints

This results in a quadratic program!

SVM Formulation : Separable Case

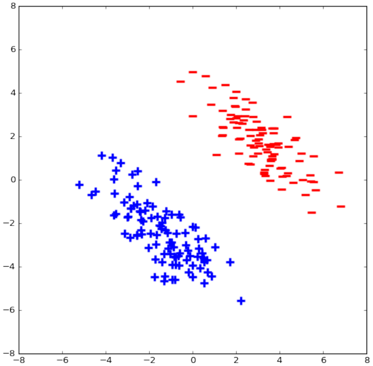
minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2$

subject to:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1$$

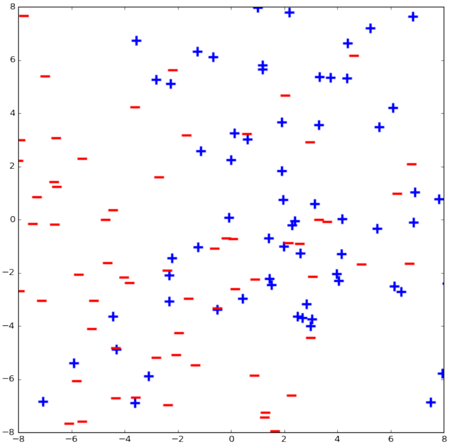
for $i = 1, \ldots, N$

Here $y_i \in \{-1, 1\}$



If data is separable, then we find a classifier with no classification error on the training set

Non-separable Data



- Quadratic program on previous slide has no feasible solution
- Which linear separator should we try to find?
- Minimising the number of misclassifications is NP-hard

SVM Formulation : Non-Separable Case

Penalty for slack terms

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2 + \quad C\sum_{i=1}^{N}\zeta_i$

subject to:

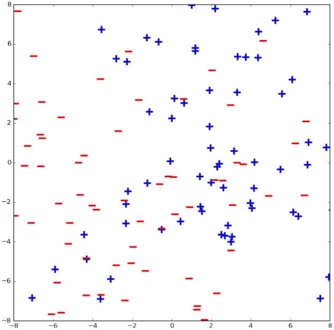
$y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq 1 - \zeta_i$

$\zeta_i \geq 0$

for $i = 1, \ldots, N$

Here $y_i \in \{-1, 1\}$

Slack to violate constraints



SVM Formulation : Loss Function

minimise: $\underbrace{\frac{1}{2}\|\mathbf{w}\|_2^2}_{\text{Regularizer}} + \underbrace{C\sum_{i=1}^{N}\zeta_i}_{\text{Loss Function}}$

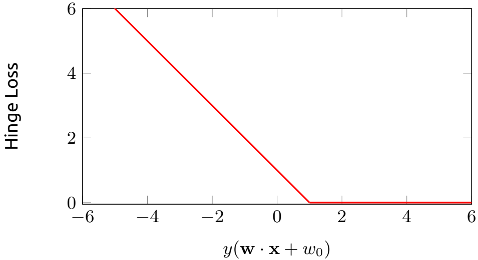
subject to:

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq 1 - \zeta_i$$

$$\zeta_i \geq 0$$

for $i = 1, \dots, N$

Here $y_i \in \{-1, 1\}$



Note that for the optimal solution, $\zeta_i = \max\{0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0)\}$

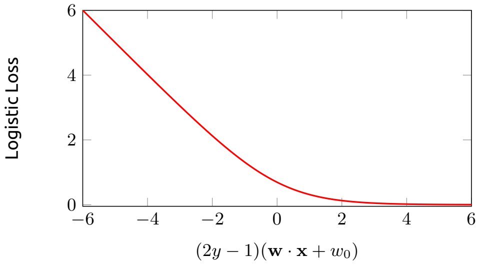
Thus, SVM can be viewed as minimizing the hinge loss with regularization

Logistic Regression: Loss Function

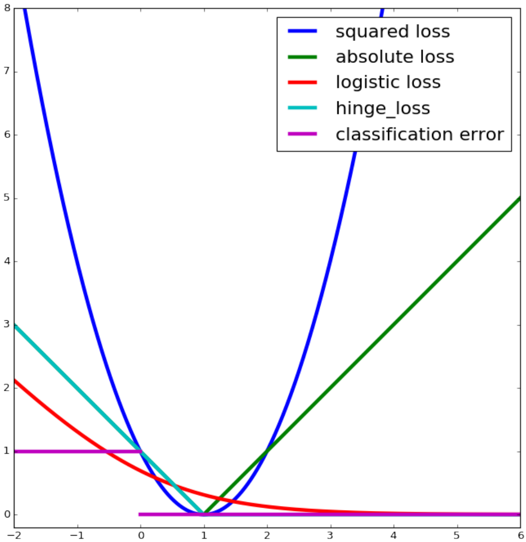
Here $y_i \in \{0, 1\}$, so to compare effectively to SVM, let $z_i = (2y_i - 1)$:

- $z_i = 1$ if $y_i = 1$
- $z_i = -1$ if $y_i = 0$

$$\begin{aligned} \mathrm{NLL}(y_i; \mathbf{w}, \mathbf{x}_i) &= -\left(y_i \log \left(\frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}_i}} \right) + (1 - y_i) \log \left(\frac{1}{1 + e^{\mathbf{w} \cdot \mathbf{x}_i}} \right) \right) \\ &= \log \left(1 + e^{-z_i (\mathbf{w} \cdot \mathbf{x}_i)} \right) = \log \left(1 + e^{-(2y_i - 1)(\mathbf{w} \cdot \mathbf{x}_i)} \right) \end{aligned}$$



Loss Functions



Outline

Multiclass Classification with SVMs (and beyond)

It is possible to have a mathematical formulation of the max-margin principle when there are more than two classes

In practice, one of the following approaches is far more common

One-vs-One:

- Train $\binom{K}{2}$ different classifiers for all pairs of classes
- At test time, choose the most commonly occurring label

One-vs-Rest:

- Train K different classifiers, one class vs the rest $K-1$
- At test time, ties may be broken by value of $\mathbf{w} \cdot \mathbf{x}_{\text{new}} + w_0$

Multiclass Classification with SVMs (and beyond)

One-vs-One

- Training roughly $K^2/2$ classifiers
- Each training procedure only uses on average $2/K$ portion of the training data
- Resulting learning problems are more likely to be "natural"

One-vs-Rest

- Training only K classifiers
- Each training procedure only uses average all the training data
- Resulting learning problems are less likely to be "natural"

For a more efficient method read the paper posted on the website

Reducing Multiclass to Binary. *E. Allwein, R. Schapire, Y. Singer*

Outline

Measuring Performance

We've encountered a few different loss functions used by learning algorithms during training time

For regression problems, it made sense to use the same loss function to measure performance (though not always necessary)

For classification problems, the natural measure of performance is classification error, number of misclassified datapoints

However, not all mistakes are equally problematic

- Mistakenly blocking a legitimate comment vs failing to mark abuse on online message boards
- Failing to detect medical risk is worse than inaccurately predicting chance of risk

Measuring Performance

For binary classification, we have:

Prediction	Actual Labels: yes	no
yes	true positive	false positive
no	false negative	true negative

For multi-class classification, it is common to write confusion matrix

Prediction	Actual Labels: 1	2	$\cdots$	K
1	N_{11}	N_{12}	$\cdots$	N_{1K}
2	N_{21}	N_{22}	$\cdots$	N_{2K}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
K	N_{K1}	N_{K2}	$\cdots$	N_{KK}

Measuring Performance

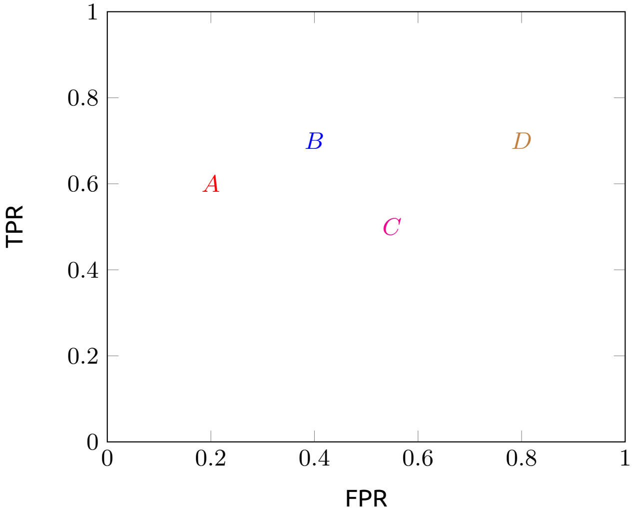
For binary classification, we have:

Prediction	Actual Labels: yes	Actual Labels: no
yes	true positive	false positive
no	false negative	true negative

False positive errors are also called Type I errors, false negative errors are called Type II errors

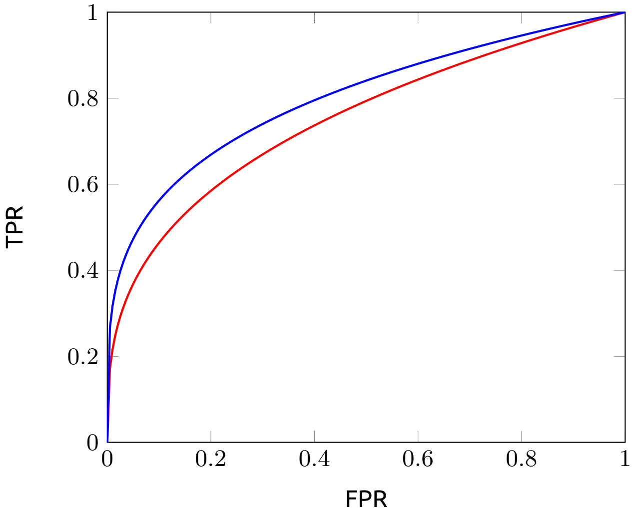
- True Positive Rate: $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$, aka sensitivity or recall
- False Positive Rate: $\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$
- Precision: $\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$

Receiver Operating Characteristic



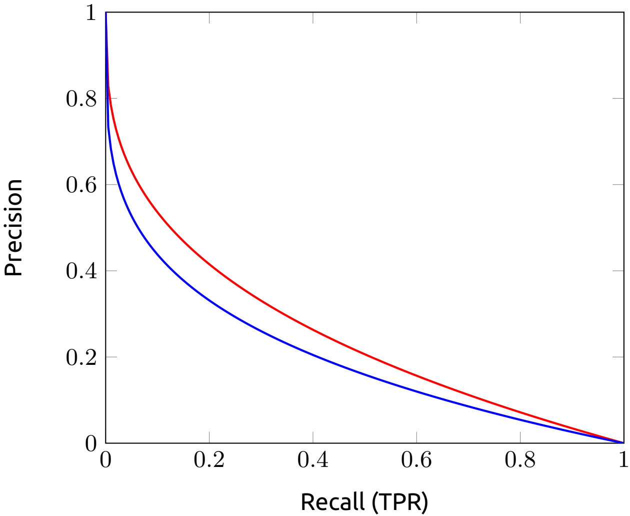
Which classifier would you pick?

Receiver Operating Characteristic



- For many classifiers, it is possible to tradeoff the FPR vs TPR
- Often summarised by the area under the curve (AUC)

Precision Recall Curves



- For many classifiers, we can tradeoff the Precision vs Recall (TPR)
- More useful when number of false negatives is very large

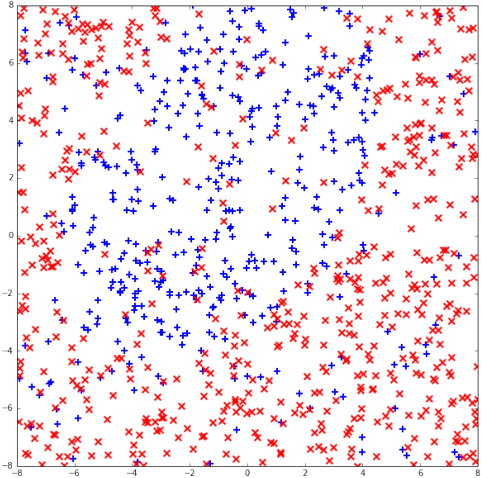
How to tune classifiers to satisfy these criteria?

- Some classifiers like logistic regression output the probability of a label being 1, *i.e.*, $p(y \mid \mathbf{x}, \mathbf{w})$
- In generative models, the actual prediction is based on the ratio of conditional probabilities,
$$\frac{p(y = 1 \mid \mathbf{x}, \boldsymbol{\theta})}{p(y = 0 \mid \mathbf{x}, \boldsymbol{\theta})}$$
- We can choose a threshold other than 1/2 (for logistic) or 1 (for generative models), to prefer one type of errors over the other
- For classifiers like SVM, it is harder (though possible) to have a probabilistic interpretation
- It is possible to reweight the training data to prefer one type of errors over the other

Outline

SVM Formulation: Non-Separable Case

What if your data looks like this?



SVM Formulation : Constrained Minimisation

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\zeta_i$

subject to:

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - (1-\zeta_i) \geq 0$$

$$\zeta_i \geq 0$$

for $i = 1, \ldots, N$

Here $y_i \in \{-1, 1\}$

Contrained Optimisation with Inequalities

Primal Form

$$
\begin{aligned}
\text{minimise} \quad & F(\mathbf{z}) & \\
\text{subject to} \quad & g_i(\mathbf{z}) \geq 0 & i = 1, \ldots, m \\
& h_j(\mathbf{z}) = 0 & j = 1, \ldots, l
\end{aligned}
$$

Lagrange Function

$$
\Lambda(\mathbf{z}; \alpha, \boldsymbol{\mu}) = F(\mathbf{z}) - \sum_{i=1}^{m} \alpha_i g_i(\mathbf{z}) - \sum_{j=1}^{l} \mu_j h_j(\mathbf{z})
$$

For convex problems, i.e., F is convex, all g_i are convex and h_i are affine, necessary and sufficient conditions for a critical point of Λ to be the minimum of the original constrained optimisation problem are given by the Karush-Kuhn-Tucker (or KKT) conditions

For non-convex problems, they are necessary but not sufficient

KKT Conditions

Lagrange Function

$$\Lambda(\mathbf{z}; \boldsymbol{\alpha}, \boldsymbol{\mu}) = F(\mathbf{z}) - \sum_{i=1}^{m} \alpha_i g_i(\mathbf{z}) - \sum_{j=1}^{l} \mu_j h_j(\mathbf{z})$$

For convex problems, Karush-Kuhn-Tucker (KKT) conditions give necessary and sufficient conditions for a solution (critical point of Λ) to be optimal

$$\begin{array}{rrl} \text{Dual feasibility:} & \alpha_i \geq 0 & \text{for } i = 1, \ldots m \\ \text{Primal feasibility:} & g_i(\mathbf{z}) \geq 0 & \text{for } i = 1, \ldots m \\ & h_j(\mathbf{z}) = 0 & \text{for } j = 1, \ldots l \\ \text{Complementary slackness:} & \alpha_i g_i(\mathbf{z}) = 0 & \text{for } i = 1, \ldots m \end{array}$$

SVM Formulation

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\zeta_i$

subject to:

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - (1 - \zeta_i) \geq 0$$

$$\zeta_i \geq 0$$

for $i = 1, \ldots, N$

Here $y_i \in \{-1, 1\}$

Lagrange Function

$$\Lambda(\mathbf{w}, w_0, \boldsymbol{\zeta}; \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\zeta_i - \sum_{i=1}^{N}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - (1 - \zeta_i)) - \sum_{i=1}^{N}\mu_i\zeta_i$$

SVM Dual Formulation

Lagrange Function

$$\Lambda(\mathbf{w}, w_0, \boldsymbol{\zeta}; \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\zeta_i - \sum_{i=1}^{N}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) - (1-\zeta_i)) - \sum_{i=1}^{N}\mu_i\zeta_i$$

We write derivatives with respect to $\mathbf{w}$, w_0 and ζ_i,

$$\frac{\partial\Lambda}{\partial w_0} = -\sum_{i=1}^{N}\alpha_i y_i$$

$$\frac{\partial\Lambda}{\partial \zeta_i} = C - \alpha_i - \mu_i$$

$$\nabla_{\mathbf{w}}\Lambda = \mathbf{w} - \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i$$

For (KKT) dual feasibility constraints, we require $\alpha_i \geq 0$, $\mu_i \geq 0$

SVM Dual Formulation

Setting the derivatives to 0, substituting the resulting expressions in Λ (and simplifying), we get a function $g(\boldsymbol{\alpha})$ and some constraints

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

Constraints

$$0 \leq \alpha_i \leq C \qquad\qquad i = 1, \ldots, N$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

Finding critical points of Λ satisfying the KKT conditions corresponds to finding the maximum of $g(\boldsymbol{\alpha})$ subject to the above constraints

SVM: Primal and Dual Formulations

Primal Form

minimise: $\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\zeta_i$

subject to:

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq (1-\zeta_i)$$

$$\zeta_i \geq 0$$

for $i = 1, \ldots, N$

Dual Form

maximise $\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j$

subject to:

$$\textstyle\sum_{i=1}^{N}\alpha_i y_i = 0$$

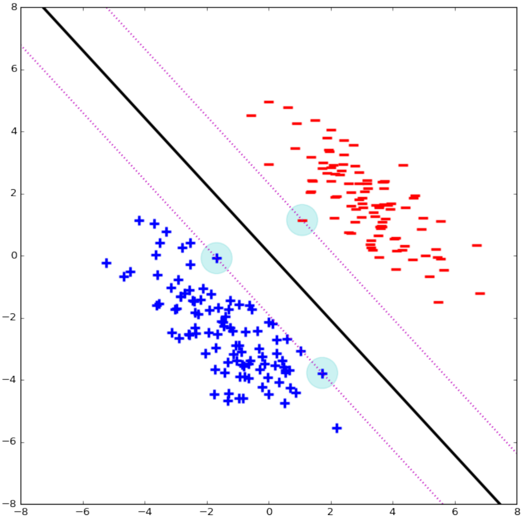
$$0 \leq \alpha_i \leq C$$

for $i = 1, \ldots, N$

KKT Complementary Slackness Conditions

- For all i, $\alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) - (1 - \zeta_i)\right) = 0$
- If $\alpha_i > 0$, $\; y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) = 1 - \zeta_i$
- Recall the form of the solution: $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$
- Thus, only those datapoints $\mathbf{x}_i$ for which $\alpha_i > 0$, determine the solution
- This is why they are called support vectors

Support Vectors



SVM Dual Formulation

maximise $\displaystyle \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\mathsf{T} \mathbf{x}_j$

subject to:

$$0 \le \alpha_i \le C \quad i = 1, \dots, N$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

- Objective depends only between dot products of training inputs
- Dual formulation particularly useful if inputs are high-dimensional
- Dual constraints are much simpler than primal ones
- To make a new prediction only need to know dot product with support vectors
  - Solution is of the form $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$
  - And so $\mathbf{w} \cdot \mathbf{x}_{\text{new}} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x}_{\text{new}}$

Outline

Gram Matrix

If we put the inputs in matrix $\mathbf{X}$, where the i^{th} row of $\mathbf{X}$ is $\mathbf{x}_i^\top$.

$$\mathbf{K} = \mathbf{X}\mathbf{X}^\top = \begin{bmatrix} \mathbf{x}_1^\top \mathbf{x}_1 & \mathbf{x}_1^\top \mathbf{x}_2 & \cdots & \mathbf{x}_1^\top \mathbf{x}_N \\ \mathbf{x}_2^\top \mathbf{x}_1 & \mathbf{x}_2^\top \mathbf{x}_2 & \cdots & \mathbf{x}_2^\top \mathbf{x}_N \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_N^\top \mathbf{x}_1 & \mathbf{x}_N^\top \mathbf{x}_2 & \cdots & \mathbf{x}_N^\top \mathbf{x}_N \end{bmatrix}$$

- The matrix $\mathbf{K}$ is positive definite if $D > N$ and $\mathbf{x}_i$ are linearly independent
- If we perform basis expansion

$$\phi : \mathbb{R}^D \rightarrow \mathbb{R}^M$$

  then replace entries by $\phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$
- We only need the ability to compute inner products to use SVM

Kernel Trick

Suppose, $\mathbf{x} \in \mathbb{R}^2$ and we perform degree 2 polynomial expansion, we could use the map:

$$\psi(\mathbf{x}) = \left[1, x_1, x_2, x_1^2, x_2^2, x_1 x_2\right]^{\mathsf{T}}$$

But, we could also use the map:

$$\phi(\mathbf{x}) = \left[1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2\right]^{\mathsf{T}}$$

If $\mathbf{x} = [x_1, x_2]^{\mathsf{T}}$ and $\mathbf{x}' = [x_1', x_2']^{\mathsf{T}}$, then

$$\begin{aligned} \phi(\mathbf{x})^{\mathsf{T}}\phi(\mathbf{x}') &= 1 + 2x_1x_1' + 2x_2x_2' + x_1^2(x_1')^2 + x_2^2(x_2')^2 + 2x_1x_2x_1'x_2' \\ &= (1 + x_1x_1' + x_2x_2')^2 = (1 + \mathbf{x} \cdot \mathbf{x}')^2 \end{aligned}$$

Instead of spending $\approx D^d$ time to compute inner products after degree d polynomial basis expansion, we only need $O(D)$ time

Kernel Trick

We can use a symmetric positive semi-definite matrix (Mercer Kernels)

$$\mathbf{K} = \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \kappa(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_N) \\ \kappa(\mathbf{x}_2, \mathbf{x}_1) & \kappa(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_N, \mathbf{x}_1) & \kappa(\mathbf{x}_N, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

Here $\kappa(\mathbf{x}, \mathbf{x}')$ is some measure of similarity between $\mathbf{x}$ and $\mathbf{x}'$

The dual program becomes

maximise $\displaystyle\sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K_{i,j}$

subject to : $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$

To make prediction on new $\mathbf{x}_{\text{new}}$, only need to compute $\kappa(\mathbf{x}_i, \mathbf{x}_{\text{new}})$ for support vectors $\mathbf{x}_i$ (for which $\alpha_i > 0$)

Polynomial Kernels

Rather than perform basis expansion,

$$\kappa(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x} \cdot \mathbf{x}')^d$$

This gives all terms of degree up to d

If we use $\kappa(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}')^d$, we get only degree d terms

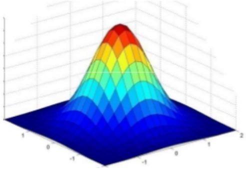
<u>Linear Kernel</u>: $\kappa(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$

All of these satisfy the Mercer or positive-definite condition

Gaussian or RBF Kernel

Radial Basis Function (RBF) or Gaussian Kernel

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$



σ^2 is known as the bandwidth

We used this with $\gamma = \frac{1}{2\sigma^2}$ when we studied kernel basis expansion for regression

Can generalise to more general covariance matrices

Results in a Mercel kernel

Kernels on Discrete Data : Cosine Kernel

For text documents: let $\mathbf{x}$ denote bag of words

Cosine Similarity

$$\kappa(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x} \cdot \mathbf{x}'}{\|\mathbf{x}\|_2 \|\mathbf{x}'\|_2}$$

Term frequency $\text{tf}(c) = \log(1 + c)$, c word count for some word w

Inverse document frequency $\text{idf}(w) = \log\left(\frac{N}{1+N_w}\right)$, N_w #docs containing w

$\text{tf-idf}(\mathbf{x})_w = \text{tf}(x_w)\text{idf}(w)$

Kernels on Discrete Data : String Kernel

Let $\mathbf{x}$ and $\mathbf{x}'$ be strings over some alphabet $\mathcal{A}$

$$\mathcal{A} = \{A, R, N, D, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V\}$$

```
IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTLESQTVQGGTV
ERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDYLQEFLGVMNTEWI
```

```
PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAERLQENLQAYRTFHVLLA
RLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLEYKIPRNEADGMLFEKK
LWGLKVLQELSQWTVRSIHDLRFISSHQTGIP
```

$\kappa(\mathbf{x}, \mathbf{x}') = \sum_s w_s \phi_s(\mathbf{x}) \phi_s(\mathbf{x}')$

$\phi_s(\mathbf{x})$ is the number of times s appears in $\mathbf{x}$ as substring

w_s is the weight associated with substring s

How to choose a good kernel?

Not always easy to tell whether a kernel function is a Mercer kernel

Mercer Condition: For any finite set of points, the Kernel matrix should be positive semi-definite

If the following hold:

- κ_1, κ_2 are Mercer kernels for points in $\mathbb{R}^D$
- $f : \mathbb{R}^D \rightarrow \mathbb{R}$
- $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^M$
- κ_3 is a Mercer kernel on $\mathbb{R}^M$

the following are Mercer kernels

- $\kappa_1 + \kappa_2$, $\kappa_1 \cdot \kappa_2$, $\alpha\kappa_1$ for $\alpha \geq 0$
- $\kappa(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})f(\mathbf{x}')$
- $\kappa_3(\phi(\mathbf{x}), \phi(\mathbf{x}'))$
- $\kappa(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x}'$ for $\mathbf{A}$ positive definite

Kernel Trick in Linear Regression

Recall the least squares objective for linear regression

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{N} (\mathbf{w}^{\mathsf{T}}\mathbf{x}_i - y_i)^2$$

and the solution $\widehat{\mathbf{w}}_{\mathsf{LS}} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}(\mathbf{X}^{\mathsf{T}}\mathbf{y})$.

We can express $\widehat{\mathbf{w}} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i$. Why?

Revisit Problem 3 on Sheet 1 (You essentially performed the 'Kernel Trick')

Next Time : Neural Networks

- Online book: Michael Nielsen http://www.michaelnielsen.org
- Draft Deep Learning Book: http://www.deeplearningbook.org