Support Vector Machines

Danushka Bollegala Lecture 12



Linear Separability

- Consider binary classification of two dimensional feature vectors
 - e.g. features = {good, bad}
 - classes = {positiveSentiment, negativeSentiment}
- If we can find a straight line that can separate all positive instances (reviews) from all negative instances (reviews) then we call such a dataset to be *linearly separable*







Higher Dimensions

- Reviews contain more than two features (words)
- In N-dimensional space, we must find (n-1) dimensional hyperplane that separates the two classes (if they are linearly separable)
- n=2 (two dimensional feature space), we had straight lines (n=1 dimensional hyperplanes)
- Hyperplane that separates the two classes might not be unique (as we saw in our previous example)

Large Margin Classifiers

- Find two hyperplanes that separates the positive class and the negative class
- Try to maximise the minimum separation (distance) between the two hyperplanes
 - The distance between the hyperplanes is called the margin
- Maximising the margin minimises the risk of misclassifying an instance at test time
 - reduces overfitting

Support Vector Machines

- Support Vector Machines (SVMs) are one of the many large margin classification methods
- Uses a constrained convex optimisation method
- Can handle non-linear separable datasets using
 - slack variables
 - kernel functions



Distance to a straight line

• Given a straight line l ax+by+c=0 show the perpendicular distance d to l from a point (α , β) is

Home Work I



Distance to a hyperplane

• A hyperplane can be expressed as the inner-product between a weight vector (coefficients) and a feature vector (variables corresponding to the dimensions)

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

 Then the distance to this hyperplane from a point p, given by the vector p can be computed as

$$rac{m{w}^{ op}m{p}}{||m{w}||}$$

- where $||\mathbf{w}||$ is the norm (L2 length) of the vector \mathbf{w}
- Observe that this formula reduces to the one we derived in the two-dimensional case in the previous slide

SVM background

- Let us assume we are given a training dataset (t_n, x_n) of n=1,...,N instances
 - target labels $t_n = \{-1, +1\}$ for binary classification
- The feature vector for the instance x is represented by $\phi(x)$
- Our classification decision of x is made according to the score y(x) given by
 - $y(x) = \mathbf{w}^{\mathsf{T}} \mathbf{\Phi}(x) + b$
- Here, **w** is the weight vector and b is the bias (scalar) term that adjust any fixed bias from the 0 threshold
 - If y(x) > 0 then we classify x to be positive and
 - otherwise negative

SVM Derivation

- If a point (instance) is correctly classified by the hyperplane then
 - $t_n y(x_n) > 0$
- The distance from a correctly classified point to the hyperplane is given by

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

SVM Derivation

 We need to find the weight vector w and bias term b such that this margin is maximised for all the training instances in our train dataset

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

This is a difficult optimisation problem involving min-max. Moreover, it is scale-invariant meaning that by setting $w \rightarrow kw$ and $b \rightarrow kb$ the term inside min does not change!

Simplification!

Scale the parameters such that a point on the decision hyperplane satisfies

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

- All correctly classified data points will then satisfy $t_n \left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) + b \right) \ge 1, \qquad n = 1, \dots, N.$
- This is called the *canonical representation* of the decision hyperplane

SVM Derivation 3

• Now the margin becomes

$$\frac{t_n y(x_n)}{||\boldsymbol{w}||} = \frac{t_n(\boldsymbol{w}^\top \phi(x_n) + b)}{||\boldsymbol{w}||} = \frac{1}{||\boldsymbol{w}||}$$

- Great!
- Now our final objective becomes to find w and b such that we maximise the margin subjected to the set of constraints that ensures our train data instances are correctly classified
- Maximising margin = minimising the norm $||\mathbf{w}||$

SVM Optimisation Problem

- Find **w** and b such that
 - minimise $\min \frac{1}{2}||w||^2$
 - subjected to

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

Constrained Optimisation

- Find x that minimises f(x)
 - unconstrained optimisation
- Find x that minimises f(x) subjected to g(x) = 0
 - constrained optimisation

unconstrained vs. constrained

• minimise
$$f(x,y) = x^2 + y^2$$

• such that g(x,y) = y - x - 1 = 0

$$f(x,y) = x^{2} + (x+1)^{2} = 2x^{2} + 2x + 1$$

 $\frac{\partial f}{\partial x} = 4x + 2 = 0$ x = -1/2y = x + 1 = 1/2

 $\min f(x,y) = (-0.5)^2 + (0.5)^2 = 0.5$



Constrained Optimisation

• Optimisation problem

minimise f(x)subject to $f_i(x) \le 0$ i = 1, ..., m $h_j(x) = 0$ j = 1, ..., p

• Lagrangian

$$L(x,\lambda,\nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

 λ_i , v_i are called Lagrange multipliers. In particular, $\lambda_i \ge 0$. (v_i can be anything)

If a point x satisfies equality and inequality constraints then it is said to be a **feasible** point. We are looking for the feasible point x* that has the smallest f(x*) value

Dual Function

 The minimum value of the Lagrangian w.r.t. x is called the Lagrange dual function, g(λ, v).

$$g(\lambda,\nu) = \inf_{x\in\mathcal{D}} L(x,\lambda,\nu) = \inf_{x\in\mathcal{D}} \left(f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x) \right)$$

- *g* is simply a sum of linear terms (Lagrange multipliers) because x is fixed.
- *g* is convex even if *f* is nonconvex because *g* is the infimum of a family of affine (linear) functions.
- g gives a lower bound for L.
- We would like to find the maximum lower bound of *f* by maximising g.

KKT Conditions

- Karush-Khan-Tucker (KKT) Conditions
 - When f is convex, KKT are necessary and sufficient conditions for the constrained convex optimisation problem (f is convex, f_i are convex and h_i are affine)

Primal feasibility

$$f_i(x^*) \le 0$$
 $i = 1, ..., m$
 $h_j(x^*) = 0$ $j = 1, ..., p$

22

Dual feasibility

 $\lambda_i^* \ge 0 \quad i = 1, \dots, m$

Complementary slackness $\lambda_i^* f_i(x^*) = 0$ i = 1, ..., m

Stability
$$\nabla_x f_0(x) + \sum_i \lambda_i \nabla_x f_i(x) + \sum_j \nu_j \nabla_x h_j(x) = \mathbf{0}$$

Idea

Minimising a two variable function f(x,y) w.r.t. x and y means that we are drawing the contours for f(x,y).

Minimising while satisfying g(x,y) = chappens when the two curves touch $f(x,y) = d_2$ each other.

figure from Wikipedia

 $f(x,y) = d_1$

g(x,y) = c

At this point the two gradients must be parallel and in opposite directions

Home Work

Use Lagrangian multiplier method to solve the optimisation problem in slide 19

Back to SVM Derivation

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



SVM

Plugging these back to the Lagrangian function we get the dual function

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

We differentiate the dual, *L̃(a)*, w.r.t. *a_n* and set to zero to find the maximiser. We get a set of N *linear* simultaneous equations that we can solve to obtain the Lagrange multipliers subject to the following constraints

$$a_n \ge 0, \qquad n = 1, \dots, N,$$

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

Observations

- We must find Lagrange multipliers a_n (collectively denoted by the vector **a**) such that L(**a**) is minimised.
- We have the inner-product between two instances
 x_n and x_m appearing in the objective function
 - $\mathbf{k}(\mathbf{x}_n, \mathbf{x}_m) = \mathbf{\Phi}(\mathbf{x}_n)^{\mathsf{T}} \mathbf{\Phi}(\mathbf{x}_m)$
 - Only the inner products matter. We do not need the explicit form of feature vectors $\phi(x)$
 - Can be *kernalised* using numerous kernel functions to overcome the non-linear separability issue.

Observations

- Note that if the Lagrange multiplier a_n = 0, then the n-th instance has no effect on the objective function L
- The instances that correspond to non-zero
 Lagrange multipliers are the ones that we need to store in our final model
 - Support Vectors
 - The instances that appear on top of the decision hyperplanes and determine its shape

Classification with SVMs

 During test time, to classify a test instance x, we simply compute the inner-product between x and each of the support vectors x_n

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

We still do not need the explicit representation of x or x_n and can work with the values returned by the kernel function

Home Work

 Using the decision function y(x) = w^Tφ(x) + b and the result we obtained for w in slide 23, derive the classification function for SVMs in the kernel form as shown in slide 27.

Kernel Functions

Linear Kernel

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \boldsymbol{x}_n^{\top} \boldsymbol{x}_m$$

- Does not use any transformations
- Polynomial Kernel

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = (\boldsymbol{x}_n^{\top} \boldsymbol{x}_m + c)^d$$

- Quadratic (d=2), and Cubic (q=3) are widely used.
- Can account for the combinations of features such as bigrams in text mining tasks
- Sigmoid Kernel

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \tanh(\boldsymbol{x}_n^{\top} \boldsymbol{x}_m + c)$$

• Exponential Radial Basis Function (RBF) Kernel

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||(\boldsymbol{x}_n - \boldsymbol{x}_m)||}{2\sigma^2}\right)$$

• Subsumes all possible kernel functions

Slack variables

- Sometimes it is easy to *shift* some of the training instances (especially around the decision hyperplane) so that the dataset becomes linearly separable
- Doing this too much will change the train data significantly and we will not learn the concept expressed by our train data
- Try to minimise the amount of shifting we do for train instances to make the problem linearly separable
 - Each train instance is associated with a slack variable that is set to a non-zero value such that the corresponding training instance is moved sufficiently to the correct side of the decision hyperplane

SVMs and slack variables



 $\xi_n > 1$ misclassification $t_n y(\mathbf{x}_n) \ge 1 - \xi_n, \qquad n = 1, \dots, N$ slacked version of the constraint $C\sum_{n=1}^{N} \xi_n + \frac{1}{2} \|\mathbf{w}\|^2$ objective function

C: cost-parameter

Higher values of C impose heavier penalties of slacking, whereas smaller C values will change the train data significantly. In practice use cross-validation to set C.

SVM slack version

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} a_n \{t_n y(\mathbf{x}_n) - 1 + \xi_n\} - \sum_{n=1}^{N} \mu_n \xi_n$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n)$$

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{n=1}^{N} a_n t_n = 0$$

$$\frac{\partial L}{\partial \xi_n} = 0 \quad \Rightarrow \quad a_n = C - \mu_n.$$

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

Same Lagrangian as before!

SVM Implementations

- LIBSVM
 - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
 - Available in a large number of programming languages
- SVM Light
 - http://svmlight.joachims.org/
 - can do ranking SVMs

Home Work

- Use LIBSVM to train a binary sentiment classifier using the train data provided in the Assignment
 1
- Compare the performance with the perceptron classifier that you implemented