Kernel Methods in Machine Learning

Lecture 2: Regularized learning and kernels

Juho Rousu

ICS-E4030 Kernel Methods in Machine Learning

21. September, 2016

Supervised learning

- Ingredients:
  - $\mathcal{X}$: a space of inputs and $\mathcal{Y}$: a space of outputs
  - Set of models $\mathcal{G} = \{g : \mathcal{X} \mapsto \mathcal{Y}\}$ mapping input to output,
  - (Training) data $\{(x_i, y_i)\}_{i=1}^N, x_i \in \mathcal{X}, y_i \in \mathcal{Y}$ sampled from an underlying unknown distribution $(x, y) \sim \mathcal{D}$
  - A loss function $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$, measuring the discrepancy between the model's predicted outputs and the true output
- Goal: to find a model g that minimises the expected loss on future instances

$$\min_{g \in \mathcal{G}} \mathbb{E}_{(x,y)\sim D} \ell(g(x), y)$$

Minimizing loss

- How to minimise the expected loss:

$$\min_{g \in \mathcal{G}} \mathbb{E}_{(x,y)\sim D} \ell(g(x), y)$$

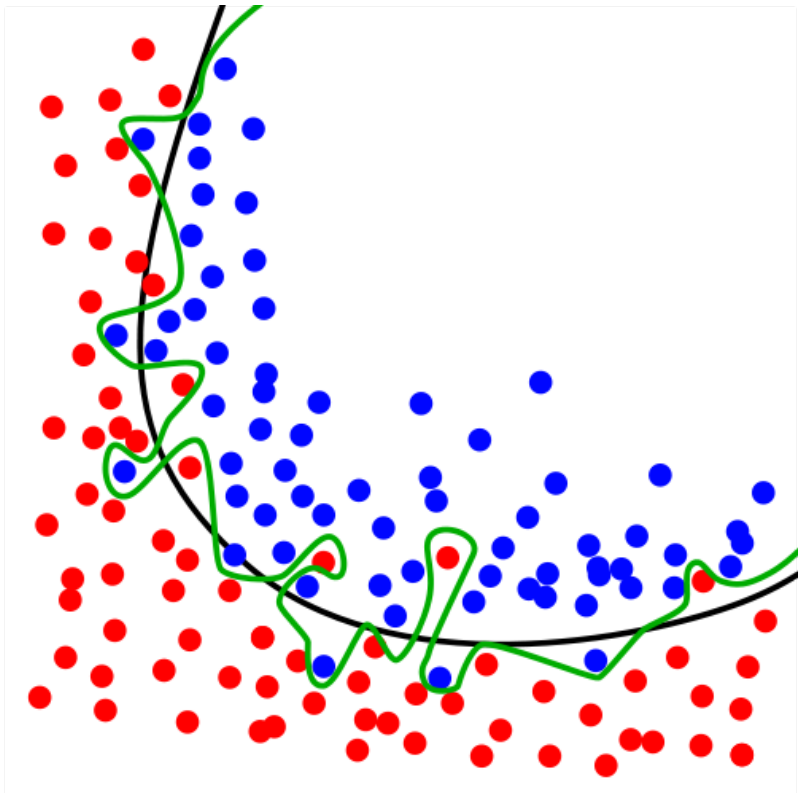
- We do not know D, cannot compute the expectation directly, but we have data
- We can estimate the average training error from a sample $S = \{(x_i, y_i)\}_{i=1}^{N}$

$$\mathbb{E}_{(x,y)\sim S} \ell(g(x), y) = \frac{1}{m} \sum_{\ell} (g(x_i), y)$$

- Minimizing training error may not be good. Q: Why is that?

Overfitting

- Generally in machine learning, it is easy to overfit to training data $\Longrightarrow$ low (even zero) error on training, high error on testing
- This is especially true with high-dimensional feature spaces and kernels
- Need to control the flexibility of the models (model selection problem)



(Image:

http://commons.wikimedia.org/wiki/File:Overfitting.svg)

Regularized learning

Regularized learning considers optimising functionals of the form

$$\min_{g \in \mathcal{G}} \frac{1}{N} \sum_{i=1}^{N} \ell(g(x_i), y_i) + \lambda \Omega(g)$$

- First term is the average error or loss that the model g makes on training data
- Second term is a regularizer that controls the complexity of the model g
  - Complex model $g \implies$ high value of $\Omega(g)$
- $\lambda \geq 0$ controls the balance between training error and model complexity, needs to be set separately
- We can apply regularised learning with kernels (dual view) as well as with features (primal view)

Linear models

We consider linear functions as the models

$$g(x) = \langle \mathbf{w}, \phi(x) \rangle$$

- $\mathbf{w}$ is a normal vector of a hyperplane
- $g(x)$ can be seen as a projection of $\phi(x)$ onto $\mathbf{w}$
- It is sometimes useful to think of $g(x)$ as a **scoring function** for the data items
- **Note:** We often omit the intercept term typically seen in linear regression i.e. $h(x) = \mathbf{w}^T \phi(x) + b$
  - The intercept term can be modelled by adding to each feature vector a constant feature $\phi_0(x) = const$

Loss functions

$$\min_{g \in \mathcal{G}} \sum_{i=1}^{N} \ell(g(x_i), y_i) + \lambda \Omega(g)$$

- By changing the loss function, different learning problems can be tackled and prior information embedded
- Examples of loss functions:
  - Squared loss $\ell(g(x), y) = (g(x) - y)^2$: used in least squares regression
  - 0/1 loss $\ell(g(x), y)) = 1_{g(x) \neq y}$, where 1_A denotes the indicator function for event A: used in classification
  - Hinge loss: $\ell(g(x), y) = \max(0, 1 - yg(x))$, where $y \in \{-1, +1\}$, used in SVM classification
  - Logistic loss $\ell(g(x), y) = \log(1 + \exp(-yg(x)))$, used in logistic regression
- Tailored loss functions can be defined for different tasks: ranking, dimensionality reduction, multi-target, structured output

Regularizers

$$\min_{g \in \mathcal{G}} \sum_{i=1}^{N} \ell(g(x_i), y_i) + \lambda \Omega(g)$$

- By changing the regulariser, we can change how model complexity is measured
- Assuming linear function $g(x) = \langle \mathbf{w}, \phi(x) \rangle$, regularisers measure some norm of $\mathbf{w}$
  - Squared Euclidean norm (squared L^2 norm): $\Omega(g) = \|\mathbf{w}\|^2 = \langle \mathbf{w}, \mathbf{w} \rangle$; most typical choice, as squared norm induces kernels naturally
  - L^1 norm: $\Omega(g) = \|\mathbf{w}\|_1 = \sum_{j=1}^{D} |w_j|$; induces sparsity when minimised - many zeros in feature weights
  - **Note:** L^0 norm: number of non-zero features; typically not used due to NP-hardness and non-convexity of optimisation
  - More complex norms are used as well, e.g. Group sparse norms $L^{2,1} = \sum_{j=1}^{r} \|\mathbf{w}_j\|$; a few non-zero feature groups selected of total r groups; Hierachical norms; structured sparsity

Linear models: dual representation

- In order to use kernels, we wish to represent the model's prediction in terms of inner products of training points

$$g(x) = \langle \mathbf{w}, \phi(x) \rangle = \sum_{i=1}^{N} \alpha_i \langle \phi(x_i), \phi(x) \rangle$$

- This implies that the weight vector is a linear combination of training points

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi(x_i)$$

- With convex optimisation problems such a representation usually exists (this is the case for problems in this course)
- To use kernels in the model, replace the inner product with a kernel

$$g(x) = \langle \mathbf{w}, \phi(x) \rangle = \sum_{i=1}^{N} \alpha_i \mathbf{k}(x_i, x)$$

Regularized learning: Ridge regression

If we choose:

- Linear model $g(x) = \langle \mathbf{w}, \phi(x) \rangle$ as the model
- Squared loss $\ell(t, z) = (t - z)^2$ as the loss function
- Quadratic regulariser $\Omega(g) = \frac{1}{2} \|\mathbf{w}\|^2$

We obtain the so called Ridge regression model

$$\min_{\mathbf{w}} \sum_{i=1}^{N} (\langle \mathbf{w}, \phi(x_i) \rangle - y_i)^2 + \lambda \|\mathbf{w}\|^2$$

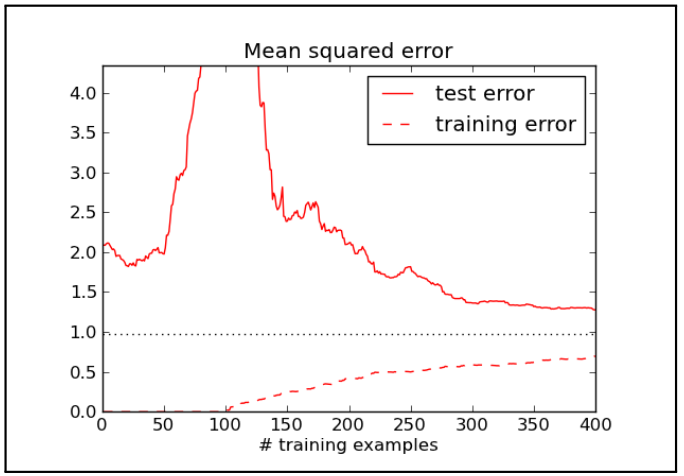
Regularized learning: Ridge regression

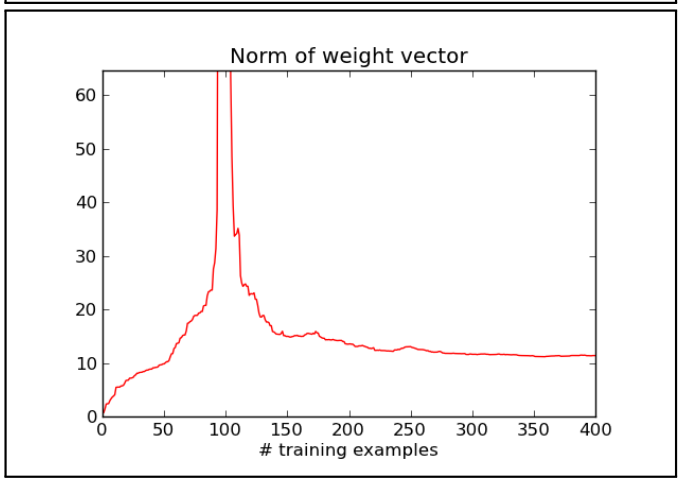
$$\min_{\mathbf{w}} \sum_{i=1}^{N} (\langle \mathbf{w}, \phi(x_i) \rangle - y_i)^2 + \lambda \|\mathbf{w}\|^2$$

- $\lambda = 0$ gives standard least squares linear regression
- High values of λ penalise weights more heavily $\implies$ weights that do not improve regression error enough will be smaller
- Q: But linear regression model is not that flexible, why do we still need to control the complexity?

Overfitting in linear regression

- Consider synthetic dataset of 100 features with gaussian features
- Ground truth generated by fitting a standard linear regression model
- The development of test and training error is shown in picture
- Training error is zero as long as we have less examples than features
- Test error rises together with the norm of the weight vector

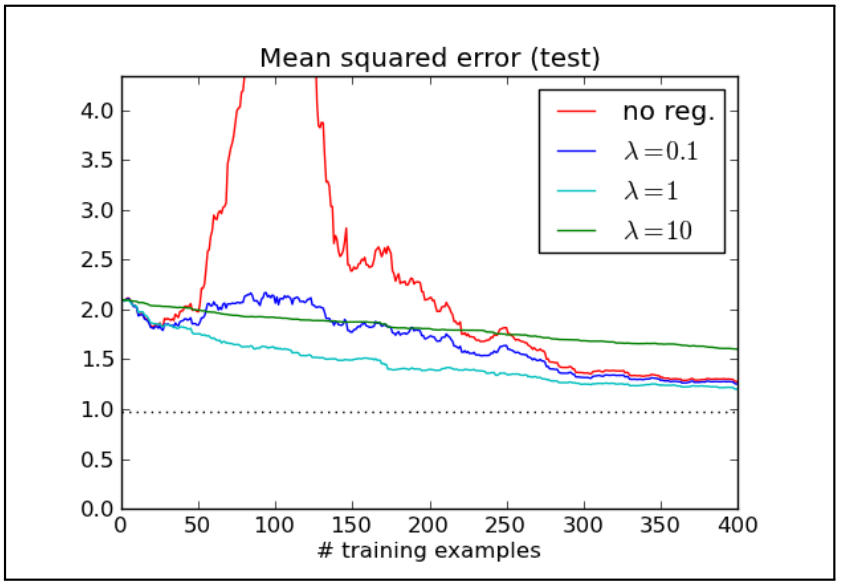


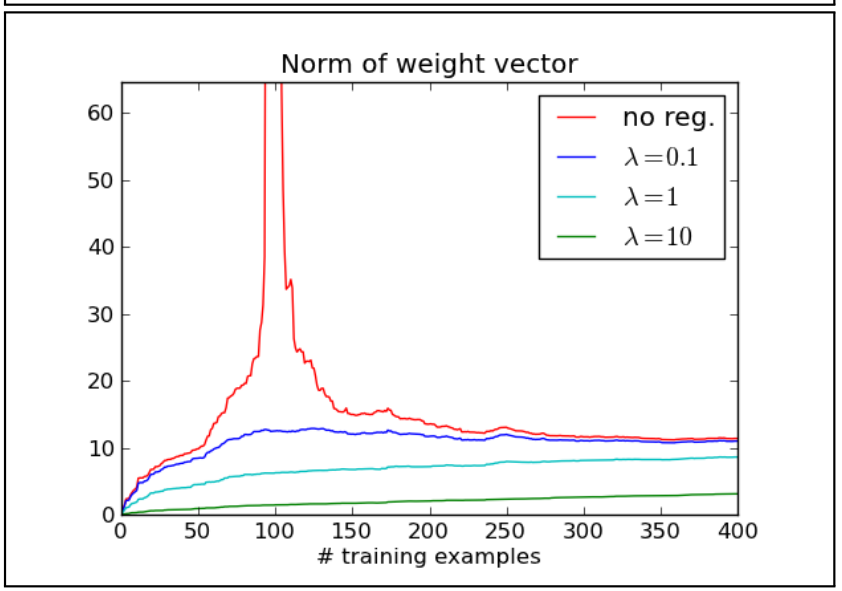


(Example by Roger Grosse, Harvard University)

Overfitting in linear regression

- Using ridge regression, the effect all but disappears
- The test error approaches the optimal rate (dotted line) from above as more data is used
- The norm of the weight vector stays in control
- λ affects the test error mildly, $\lambda = 1$ giving the test error, $\lambda = 0.1$ being under-regularized, $\lambda = 10$ being over-regularized





Supervised learning: ridge regression[1]

Ridge regression

$$\min_{\mathbf{w}} \sum_{i=1}^{N} (\langle \mathbf{w}, \phi(x_i) \rangle - y_i)^2 + \lambda \|\mathbf{w}\|^2$$

The optimal weight vector is given in closed form:

$$\mathbf{w} = (X^T X + \lambda I_D)^{-1} X^T \mathbf{y}$$

- X is $N \times D$ matrix of training inputs
- $\mathbf{y}$ is a $N \times 1$ vector of training outputs
- I_D is the $D \times D$ identity matrix
- λI_D is the contribution of the regulariser, otherwise the same as standard linear regression

The prediction of the ridge regression model is given in closed form by:

$$g(x) = \langle \mathbf{w}, \phi(x) \rangle = \mathbf{w}^T \phi(x) = \mathbf{y}^T X (X^T X + \lambda I_D)^{-1} \phi(x)$$

[1] Shawe-taylor & Cristianini book, section 2.2.2.

The matrices involved

- Data matrix of feature vectors:

$$X = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_D(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_2 D(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \dots & \phi_D(x_N) \end{bmatrix}$$

- $X^T X = \sum_{i=1}^N \phi(x_i)\phi(x_i)^T$ is a $D \times D$ matrix. If the data is centered so that $\sum_{i=1}^N \phi(x_i) = \mathbf{0}$ it is the empirical co-variance matrix of the data
- $X\mathbf{w} = (\langle w, \phi(x_1)\rangle, \dots, \langle w, \phi(x_N)\rangle)^T = (g(x_1), \dots, g(x_N))^T$ is the vector of predictions for the data set.
- $X^T\mathbf{y} = (c_1, \dots, c_D)^T$ is a vector of inner products $c_j = \langle \mathbf{x}_j, \mathbf{y}\rangle$, where $\mathbf{x}_j = (\phi_j(x_1), \dots, \phi_j(x_N)$ is a column of the data matrix corresponding to the j'th feature.
- $XX^T = K$ is the linear kernel matrix with entries $K_{ih} = \langle \phi(x_i), \phi(x_h)\rangle$

Derivation of the ridge regression solution

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \min_{\mathbf{w}} \sum_{i=1}^{N} (g(x_i) - y_i)^2 + \lambda \|\mathbf{w}\|^2$$

- Write the squared error in vector form, and expand:

$$\langle \mathbf{y} - X\mathbf{w}, \mathbf{y} - X\mathbf{w} \rangle = \mathbf{y}^T\mathbf{y} - 2\mathbf{w}^T X^T y + \mathbf{w}^T X^T X\mathbf{w}$$

- Plug back to the objective

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \min_{\mathbf{w}} \mathbf{y}^T\mathbf{y} - 2\mathbf{w}^T X^T\mathbf{y} + \mathbf{w}^T X^T X\mathbf{w} + \lambda\mathbf{w}^T\mathbf{w}$$

- Setting the derivative of the objective w.r.t $\mathbf{w}$ to zero gives the optimum point

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = 0$$

Multivariate derivatives cheat sheet

- Given $f(\mathbf{x}) : \mathbb{R}^D \mapsto \mathbb{R}$, the derivative is a vector $Df(\mathbf{x})$ with the components

$$Df(\mathbf{x})_j = \frac{\partial f(\mathbf{x})}{\partial x_j}$$

  where the partial derivates adhere to usual single variable calculus rules.
- For Quadratic $f(x)$, we will need the following rules , where $q \in \mathbb{R}^D$ and $P \in \mathbb{R}^{D \times D}$
  - $D\mathbf{x}^T q = q$
  - $D\mathbf{x}^T P\mathbf{x} = 2P\mathbf{x}$

Derivation of the ridge regression solution

- Setting the derivative of the objective

$$\mathcal{L}(\mathbf{w}) = \mathbf{y}^T\mathbf{y} - 2\mathbf{w}^T X^T\mathbf{y} + \mathbf{w}^T X^T X\mathbf{w} + \lambda\mathbf{w}^T\mathbf{w}$$

  w.r.t $\mathbf{w}$ to zero gives the optimum point

$$\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}) = -2X^T\mathbf{y} + 2X^T X\mathbf{w} + 2\lambda\mathbf{w} = 0$$

- Divigin by 2 and rearranging we get

$$X^T\mathbf{y} = X^T X\mathbf{w} + \lambda\mathbf{w} = (X^T X + \lambda I_D)\mathbf{w}$$

- Solving the linear equation system for $\mathbf{w}$ gives our optimal weight vector:

$$\mathbf{w} = (X^T X + \lambda I_D)^{-1} X^T y$$

Ridge regression with kernels

- The ridge regression model's prediction

$$g(x) = \langle \mathbf{w}, \phi(x) \rangle = \langle (X^T X + \lambda I_D)^{-1} X^T \mathbf{y}, \phi(x) \rangle$$

can be written in dual form only involving inner products between training points

$$g(x) = \sum_{i=1}^{N} \alpha_i \langle \phi(x_i, \phi(x) \rangle = \sum_{i=1}^{N} \alpha_i k(x_i, x),$$

where k is the linear kernel function

- Dual variables α_i give a weight for each training point, defined by

$$\alpha = \left(XX^T + \lambda I_N \right)^{-1} \mathbf{y} = (K + \lambda I_N)^{-1} \mathbf{y}$$

where K is the linear kernel matrix of the training data.

Derivation of the ridge regression dual solution

- We need to express the solution in terms of a linear combination of the training points, i.e. $\mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi(x_i) = X^T \alpha$
- Setting again the derivative of the objective w.r.t $\mathbf{w}$ to zero

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = -2X^T \mathbf{y} + 2X^T X\mathbf{w} + 2\lambda \mathbf{w} = 0$$

- Dividing by λ and rearranging, the equation

$$X^T y = X^T X\mathbf{w} + \lambda \mathbf{w}$$

we get

$$\mathbf{w} = \frac{1}{\lambda}(X^T y - X^T X\mathbf{w}) = X^T \frac{1}{\lambda}(y - X\mathbf{w}) = X^T \alpha$$

which gives

$$\alpha = \frac{1}{\lambda}(\mathbf{y} - X\mathbf{w})$$

Derivation of the ridge regression dual solution

- Further substituting $\mathbf{w} = X^T\alpha$ to

$$\alpha = \frac{1}{\lambda}(\mathbf{y} - X\mathbf{w})$$

gives

$$\lambda\alpha = \mathbf{y} - XX^T\alpha$$

- The dual variables can be solved from above

$$\alpha = (XX^T + \lambda I_N)^{-1}\mathbf{y} = (K + \lambda I_N)^{-1}\mathbf{y},$$

where $K = XX^T$ is the kernel matrix of inner products of the training points

Ridge regression with kernels

The ridge regression model's prediction is given in kernel form by

$$g(x) = \sum_{i=1}^{N} \alpha_i k(x_i, x) = \mathbf{y}^T (K + \lambda I_N)^{-1} \mathbf{k}$$

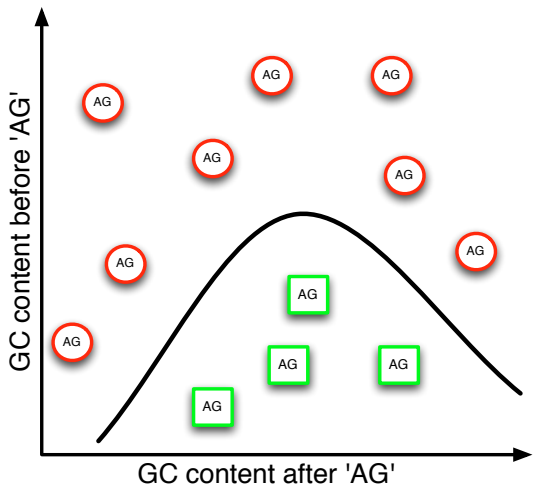
where

- K is a $N \times N$ matrix of inner products of training points $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$, i.e. it is the kernel matrix
- $\mathbf{k}$ is a vector with kernel values $k(x_i, x)$ between each training point and the point x to be predicted
- I_N is an $N \times N$ identity matrix

Supervised learning problems: Classification

- Outputs are categorical objects , with two (e.g. $\{0, 1\}, \{-1, +1\}$) , or multiple classes $\{1, 2, \ldots, K\}$
- We will first deal with binary classification with $\mathcal{Y} = \{-1, +1\}$
- Ideally, one would like to minimise zero-one loss on future instance:

$$\ell_{zo}(t, z) = \begin{cases} 0 & \text{if } t = z, \\ 1 & \text{if } t \neq z, \end{cases}$$



Regularized learning: Binary classification

- Our optimisation problem for **binary classification** takes the form

$$\min_{\mathbf{w}\in\mathcal{H}} \sum_{i=1}^{N} \ell(\langle \mathbf{w}, \phi(x_i)\rangle, y_i) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

- Classification:
  - Minimizing zero-one loss ℓ_{zo} would be desirable statistically
  - But finding the zero-one loss minimising $\mathbf{w}$ is known to be NP-hard
  - Instead different convex upper bounds for ℓ_{zo} are used for faster optimisation

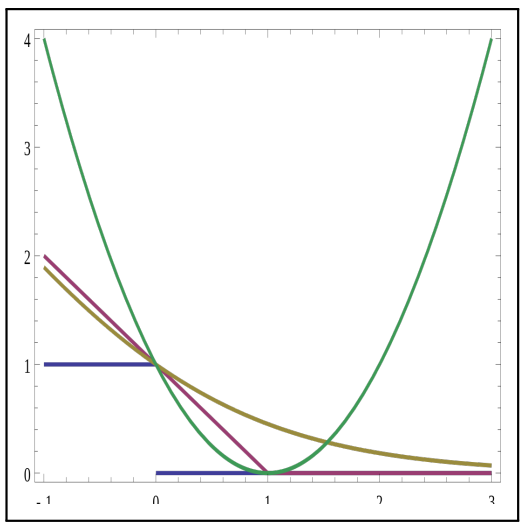
Loss functions for classification

For efficiency of optimisation, two properties are preferable:

- smoothness i.e. differentiability
- convexity

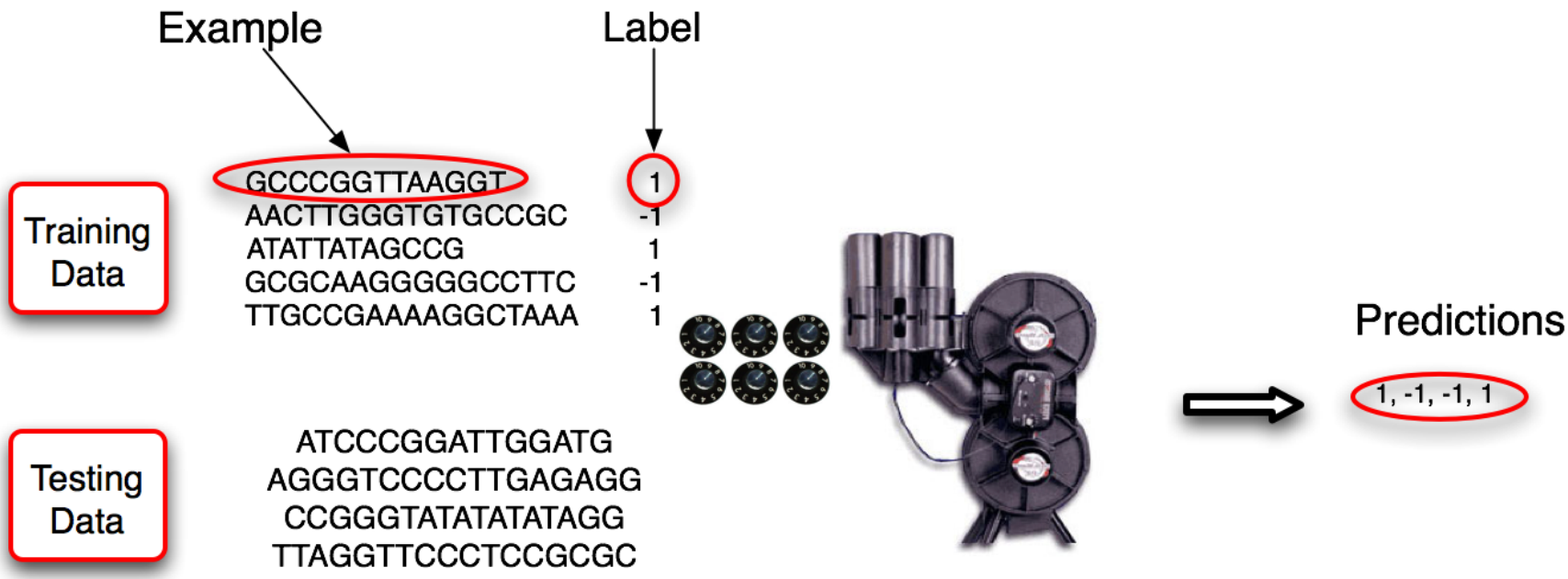
Examples of loss functions:

- Squared loss $\ell(g(x), y) = (g(x) - y)^2$: smooth and convex, however too strict for classification. Why?
- 0/1 loss $\ell(g(x), y)) = 1_{g(x) \neq y}$: non-smooth and non-convex
- Hinge loss: $\ell(g(x), y) = \max(0, 1 - yg(x))$: non-smooth and convex
- Logistic loss $\ell(g(x), y) = \log(1 + \exp(-yg(x)))$: smooth and convex



Classification learning with kernels: Support vector machines[2]

- Training data from two classes (+1,-1)
- Discriminative model is learned
- Testing on an independent test set

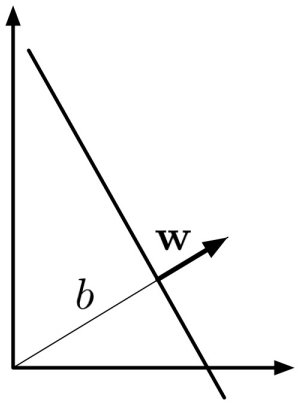


[2] S-T & C book section 7.2

Classification learning with linear models

We will concentrate on models that take a linear form:

$$g(x) = \sum_{j=1}^{D} w_j \phi_j(x) + b = \langle \mathbf{w}, \phi(x) \rangle + b$$



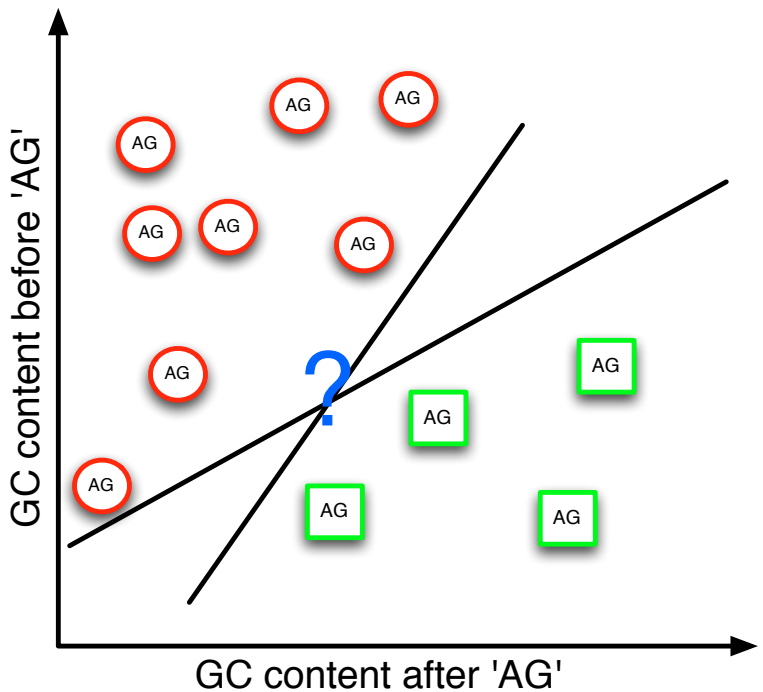
- $\langle \mathbf{w}, \mathbf{x} \rangle$ denotes the inner product (also known as dot product and scalar poduct), between the weight vector and the feature vector
- Geometrically $g(x)$ is a hyperplane ($D-1$-dimensional plane) dividing the feature space into two half-spaces
- $\mathbf{w}$ is the normal vector of the hyperplane, orthogonal to the hyperplane
- Values of $g(x)$ increase in the direction of $\mathbf{w}$

Classification learning with linear models

The model $g(x)$ is turned into a classifier by thresholding at 0:

$$f(x) = \begin{cases} +1 & \text{if } g(x) > 0 \\ -1 & \text{if } g(x) < 0 \end{cases}$$

- The goal of learning the parameters $(\mathbf{w}, b)$ is to put the hyperplane in between the two classes
  - $f(x) = -1 \Leftrightarrow g(x) < 0$ for the negative class
  - $f(x) = +1 \Leftrightarrow g(x) \geq 0$ for the positive class

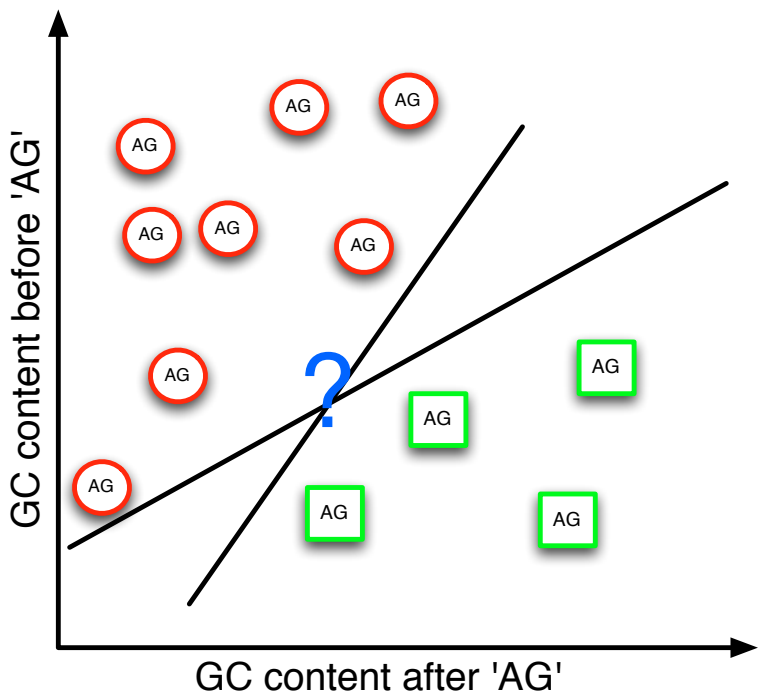


Measuring classification success: loss function

In binary classification ($\mathcal{Y} = \{-1, +1\}$), one typically uses the 0/1-*loss function to count errors*:

$$\ell(f(x_i), y_i) = \begin{cases} 0 & \text{if } f(x_i) = y_i \\ 1 & \text{if } f(x_i) \neq y_i \end{cases}$$

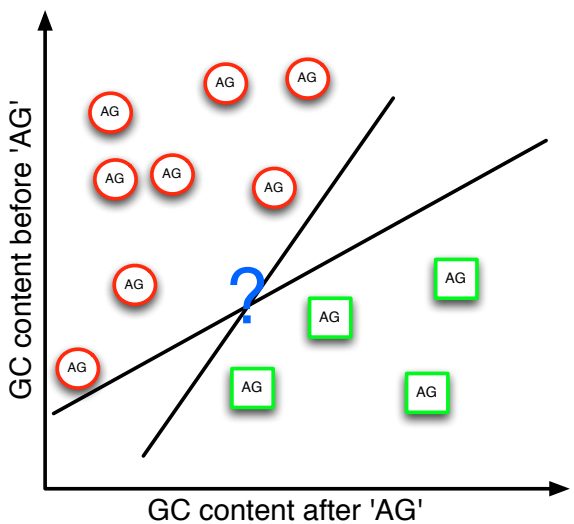
- If all training data are on the correct side of the hyperplane we have $\sum_{i=1}^{n} \ell(f(x_i), y_i) = 0$
- However, there might be several hyperplanes that achieve zero loss
- Does it matter which one we choose?

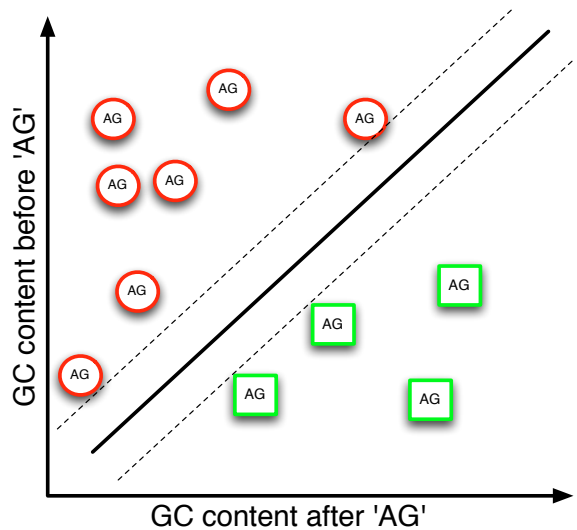


Maximum margin hyperplane

One good solution is to choose the hyperplane that lies furthest away from the training data:

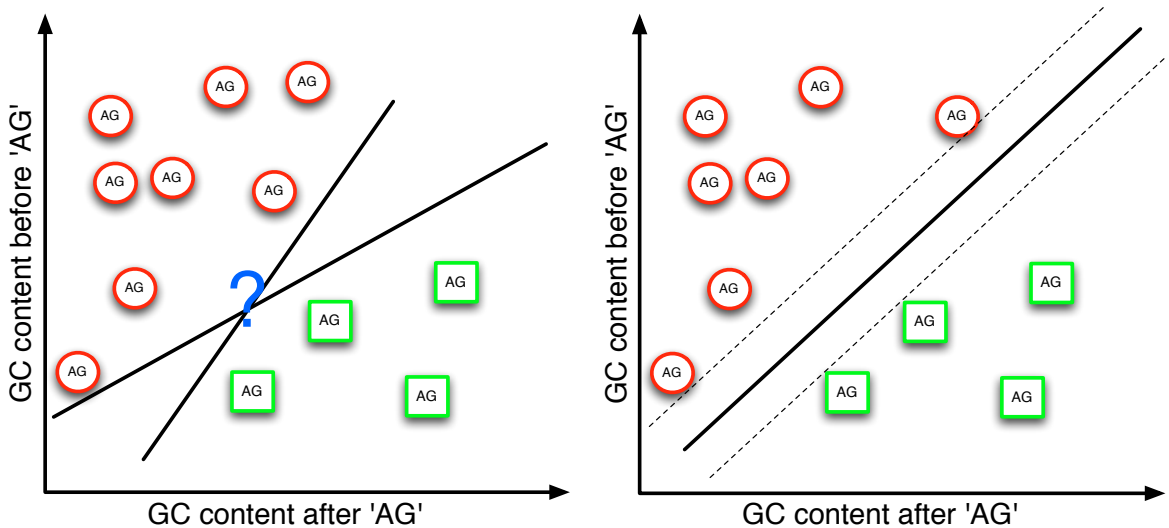
- Robustness: small change in the training data will not change the classifications too much
- Theoretically can be shown to lead to good performance — a large margin, distance between the hyperplane, is tied to low error on unseen data
- Support vector machines (SVM) are based on this principle





How to Maximize the Margin?

- The margin is given by $\gamma(x) = yg(x) = y(\langle \mathbf{w}, \phi(x) \rangle + b)$
- By multiplying the weights with a arbitrary $c > 1$, one can increase the margin without limit
  $y(\langle c\mathbf{w}, \phi(x) \rangle + b) = c \cdot y(\langle \mathbf{w}, \phi(x) \rangle + b) = c \cdot \gamma(x_i)$
- Any separating hyperplane can be made to have as large margin as we wish, cannot choose between them by taking the maximum!

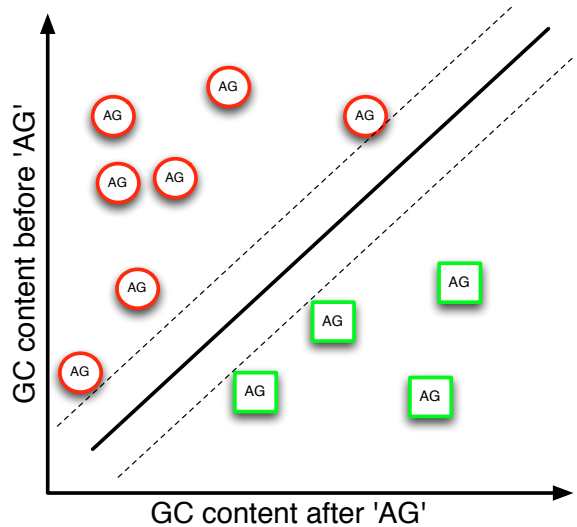


How to Maximize the Margin?

Margin maximization becomes sensible if we add a constraint that the length of the weight vector should not change: $||\mathbf{w}|| = \sqrt{\sum_{i=1}^{m} w_j^2} = 1$.

Our optimization problem becomes:

$$\begin{aligned} \textit{Maximize} \quad & \gamma \\ \textit{Subject to} \quad & y_i(\langle \mathbf{w}, \phi(x_i) \rangle + b) \geqslant \gamma \\ & \text{for all } i = 1, \ldots, N, \\ & ||\mathbf{w}|| = 1 \end{aligned}$$

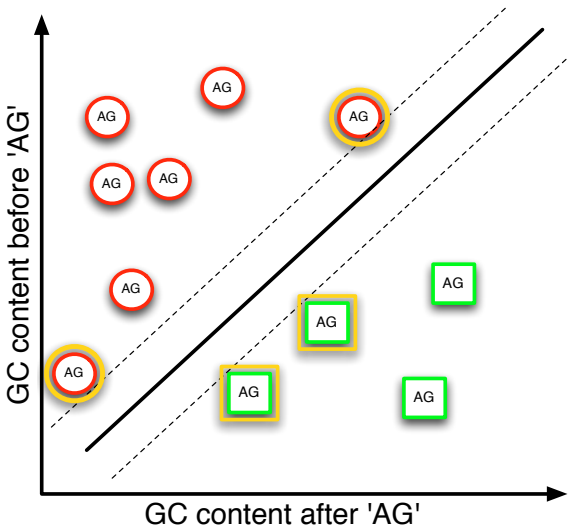


- First constraint says that all examples have at least margin of γ
- Second constraint fixes the norm of the weight vector — intuitively, gives fixed measurement scale
- Now there will be a maximum margin hyperplane

How to Maximize the Margin? Canonical hyperplane

We can equivalently fix the margin $\gamma = 1$ and seek for shortest weight vector that achieves the margin

$$\begin{aligned} \textit{Minimize} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \textit{Subject to} \quad & y_i(\langle \mathbf{w}, \phi(x_i) \rangle + b) \geqslant 1 \\ & \text{for all } i = 1, \ldots, N. \end{aligned}$$



- The hyperplane thus obtained is called the canonical hyperplane
- All points have at least margin of $= 1$
- The points that have margin $\gamma(x) = 1$ are called *support vectors*
- The set of support vectors uniquely identifies the hyperplane

SVM as a regularised learning problem

- Consider the SVM problem

$$\begin{aligned} \textit{Minimize} \quad & \tfrac{1}{2}||\mathbf{w}||^2 \\ \textit{Subject to} \quad & 1 - y_i(\langle \mathbf{w}, \phi(x_i) \rangle + b) \leq 0 \\ & \text{for all } i = 1, \ldots, N. \end{aligned}$$

- The constraint can be seen as a requirement of having zero Hinge loss

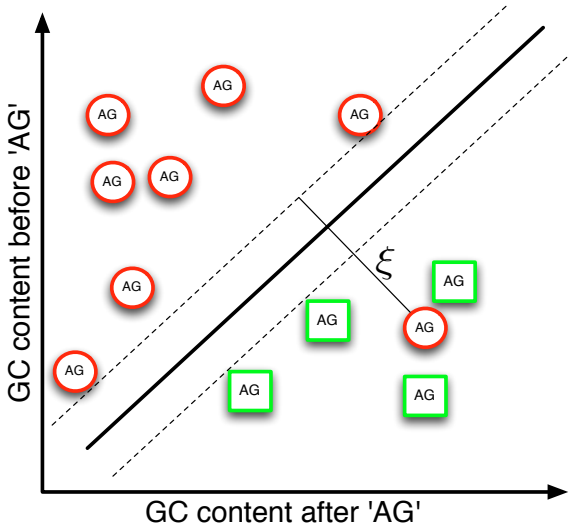
$$\ell_{Hinge}(g(x), y) = \max(1 - yg(x), 0)$$

- The regulariser is $\Omega(g) = \frac{1}{2}||\mathbf{w}||^2$ and the parameter $\lambda = 1$
- Thus it is an extreme case of regularised learning problem that does not allow any error

How to Maximize the Margin? Non-separable data

In practise data rarely separates cleanly into two halfspaces by a hyperplane, for multitude of reasons:

- Measurement errors (noisy features)
- Insufficient features
- Annotation errors (noisy labels)

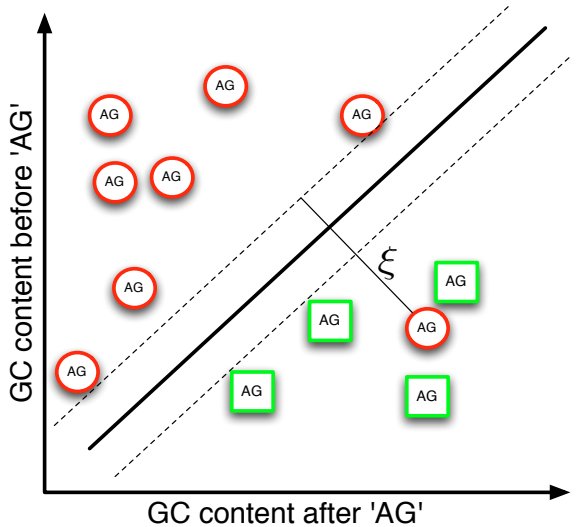


- For any hyperplane, there will be an example with a negative margin
- Our optimization problem has no feasible solution

Soft-Margin SVM (Cortes & Vapnik, 1995)

The soft-margin SVM allows some of the training points to have smaller margin than $\gamma(x) = 1$, subject to a penalty:

$$\begin{aligned} \textit{Minimize} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i \\ \textit{Subject to} \quad & 1 - y_i(\langle \mathbf{w}, \phi(x_i)\rangle + b) \leq \xi_i \\ & \text{for all } i = 1, \ldots, N. \\ & \xi_i \geq 0 \end{aligned}$$



- ξ_i is called the slack variable, when positive, the margin $\gamma(x_i) < 1$
- The sum of slacks is to be minimized so the objective still favours hyperplanes that separates the classes well
- The coefficient $C > 0$ controls the balance between maximizing the margin and the amount of slack needed

Soft-margin SVM as a regularised learning problem

- Consider the soft-margin SVM problem

$$\begin{aligned} \textit{Minimize} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i \\ \textit{Subject to} \quad & 1 - y_i(\langle \mathbf{w}, \phi(x_i)\rangle + b) \leq \xi_i \\ & \text{for all } i = 1, \ldots, N. \\ & \xi_i \geq 0 \end{aligned}$$

- The slack variables control the Hinge loss

$$\xi_i \geq \ell_{Hinge}(g(x_i), y_i) = \max(1 - y_i g(x_i), 0)$$

- The balance between error minimisation is controlled by C, we can change the variable to $\lambda = \frac{1}{C}$
- Thus we have an equivalent problem

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \ell_{Hinge}(\langle \mathbf{w}, \phi(x_i)\rangle + b, y_i) + \frac{\lambda}{2}||\mathbf{w}||^2$$

From features to kernels

- Theorem: The optimal $\mathbf{w}$ can be written as a linear combination of the examples (for appropriate α's):

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i) \qquad \Rightarrow \text{Plug in!}$$

- Corollary: Optimization problem only depends on the inner products of the examples, no reference to original features required:
  - $\langle \mathbf{w}, \phi(x) \rangle \qquad \Rightarrow \sum_{i=1}^{N} \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle$
  - $\langle \mathbf{w}, \mathbf{w} \rangle \qquad \Rightarrow \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle$
- Denote the inner product with a kernel function: $k(x, x') = \langle \phi(x_i), \phi(x') \rangle$

Dual Soft-Margin SVM

A dual optimization problem (gives the same solution) to the soft-margin SVM is

$$\begin{aligned} \textit{Maximize} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ \textit{Subject to} \quad & 0 \leq \alpha_i \leq C \\ & \text{for all } i = 1, \ldots, N. \\ & \sum_{i=1}^{N} \alpha_i y_i = 0 \end{aligned}$$

- Note that the slack variables (ξ) and the weight vector ($\mathbf{w}$) have disappeared, the only variables to be optimized are the α_i's
- The data only appears through the kernel functions $k(x_i, x_j)$
- The full derivation requires convex optimisation theory

Recap

- Regularized learning framework allows us to fight overfitting in high-dimensional feature spaces
- We can express regularised learning problems in dual formulation, where data only appears through kernels
- Ridge regression as an example of a regression method
- Support vector machines as an example of a classification method
- Margin maximisation and regularisation as two connected concepts