Introduction to Classification - The basics (kNN, LDA, SVM)

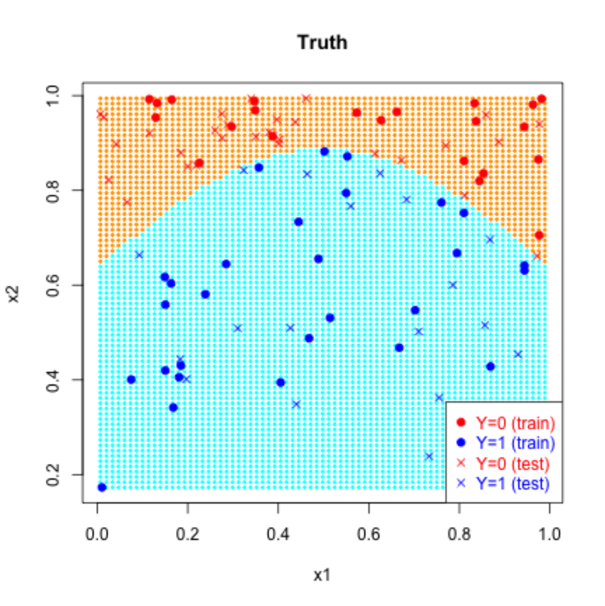
Daniel Lawson University of Bristol

Lecture 05.1.2 (v1.0.2)

Signposting

- You should have come here from 05.1.1 - Introduction to Classification
- This is part 2 of Lecture 5.1, which is split into:
  - 5.1.1 covers a Classification Introduction and Interpretation
  - 5.1.2 covers kNN, LDA, SVM
- In 5.2 we cover boosting and ensemble methods
- In 6 we cover Tree and Forest methods

Classification

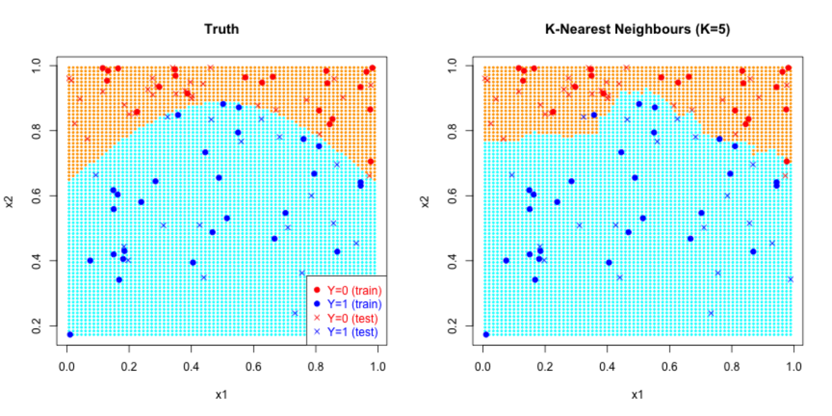


K-Nearest Neighbour classification

- In Block 4, we introduced K-NN for density estimation.
  - We defined some choices of distance function
  - We obtained the K nearest neighbours of points in R
- Armed with those neighbours, a classifier can be implemented by using **majority vote** of the labels of all k neighbours.
- A naive implementation scales poorly with N, but an approximate lookup can control complexity.
- See also: Condensed nearest neighbor[1] approaches to reduce the amount of data required at the classification stage.

[1] Hart P, The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory 18 (1968) 515-516. doi: 10.1109/TIT.1968.1054155

K-Nearest Neighbour example



Linear Discriminant Analysis

- Developed in **1936 by R. A. Fisher**[2] and extended to the current multi-class form in 1948[3].
- The goal is to **project** a high dimensional space into K dimensions, **maintaining** (linear) classification ability.
- Prediction benefit comes only from reducing overfitting
- Strong **relationship with PCA**, often used in tandem (PCA then LDA)
- Assumes that each class k has a different mean μ_k and a shared covariance matrix Σ
- Kernel Discriminant Analysis exists[4]

---

[2] Fisher R, "The Use of Multiple Measurements in Taxonomic Problems" (1936) Annals of eugenics (!), now "Annals of Human Genetics"

[3] Rao C, "Multiple Discriminant Analysis" (1948) JRSSB

[4] Mika, S et al "Fisher discriminant analysis with kernels" (1999) NIPS IX: 41-48

LDA algorithm

1. Compute the mean location μ_k for each class k and the overall mean μ, as well as the assignment sets D_k.
2. Compute the **within-class scatter matrix** S_W:
   $S_W = \sum_{k=1}^{K} S_k$ where

$$S_k = \sum_{i \in D_k} \left(\vec{x} - \vec{\mu}_k\right)\left(\vec{x} - \vec{\mu}_k\right)^T$$

3. Compute the **between-class scatter matrix** S_B:

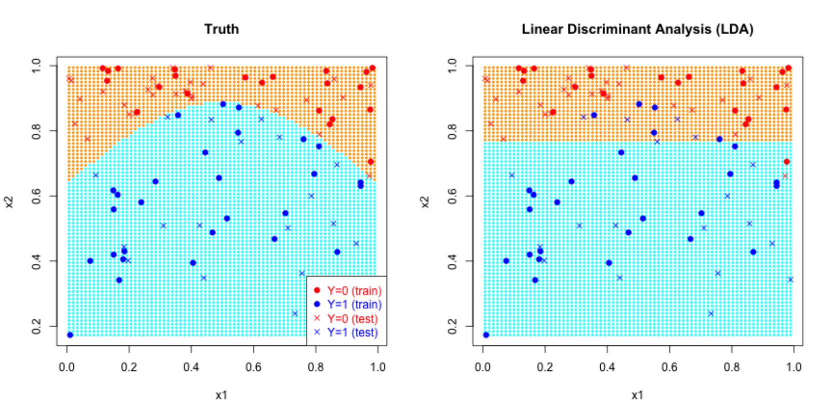
$$S_B = \sum_{k=1}^{K} n_i \left(\vec{\mu}_k - \vec{\mu}\right)\left(\vec{\mu}_k - \vec{\mu}\right)^T$$

4. Solve for the **eigenvalues** λ_k and **eigenvectors** v_k of $S_W^{-1} S_B$
5. **Choose a dimension** threshold K^*, either using the same methods as for PCA, or cross-validation
6. **Predict** using μ_k ...

LDA prediction

- Class prediction can use any information in the LDA data summary. Options include:
  - Nearest cluster
  - **Likelihood**: $\Pr(\vec{x}|y_k = c) = \mathrm{Normal}\left(\mu_k, \Sigma\right)$
  - **Posterior**: $\Pr(y_k = c|\vec{x}) \propto \Pr(\vec{x}|y_k = c)p(y_k = c)$; i.e. reweight classes according to their frequency

LDA example



Truth
Linear Discriminant Analysis (LDA)
Y=0 (train)
Y=1 (train)
Y=0 (test)
Y=1 (test)
x1
x2
0.0
0.2
0.4
0.6
0.8
1.0

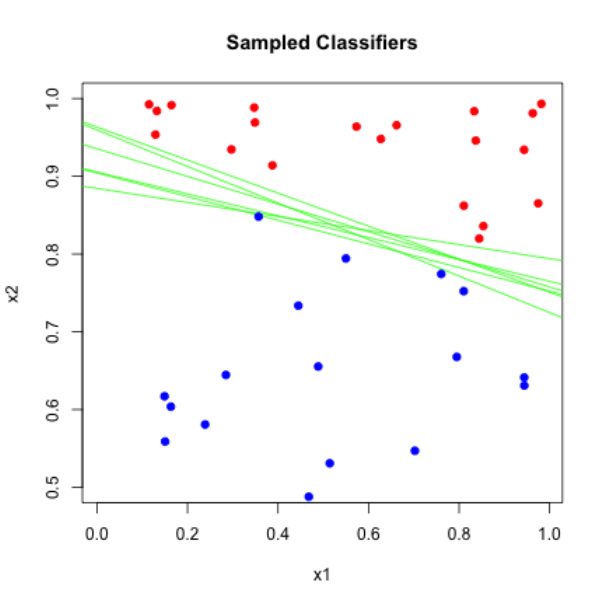
Towards Support Vector Machines

- LDA uses **all the points** for classification, which makes it slow
- It is also **linear**
- (It could be made non-linear by mapping the data to high dimensions, but this is often infeasible)
- Moving towards SVM, we:
  - Can exploit the **kernel-trick** to make a non-linear decision boundary without explicit mapping
  - Switch focus from group **means** to making the largest group **separation**
  - If we only want to **discriminate classes**, we can only use a subset of the data, the **support vectors**, for the decision
- This makes the method:
  - robust to distributional assumptions
  - non-generative

Support Vector Machine overview

- Find the **maximum margin hyperplane** separating the classes **closest** points
- Allow soft margins: misclassified points are down-weighted
- Nonlinearity: express distances as **inner products**, allowing non-linearities via the Kernel trick
- Algorithm: finding the hyperplane is a “quadratic optimisation problem”.

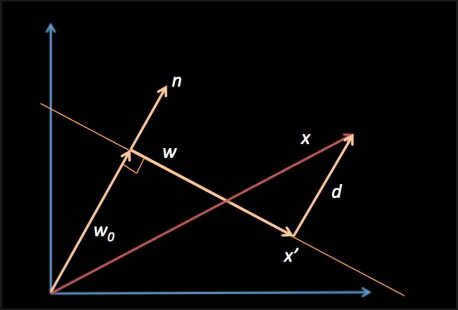
SVM illustration: solution space



Planar geometry

- The data are $\vec{x} \in D$ containing N examples
- The labels are $y_i \in (-1, 1)$
- A **hyperplane** is defined via:
  - $\vec{w}$, the coordinates of the plane
  - $\vec{w}_0$, a point on the plane chosen such that $\vec{w}_0$ is perpendicular to $\vec{w}$:

$$\vec{w} \cdot (\vec{x} - \vec{w}_0) = \vec{w} \cdot \vec{x} + b = 0$$



SVM margins

- The **distance of a point to the line** is the residual after the point is projected onto the line:

$$d_{\vec{w}}(\vec{x}) = \vec{n} \cdot (\vec{x} - \vec{x}') = \frac{|\vec{w} \cdot \vec{x} + b|}{|\vec{w}|}$$

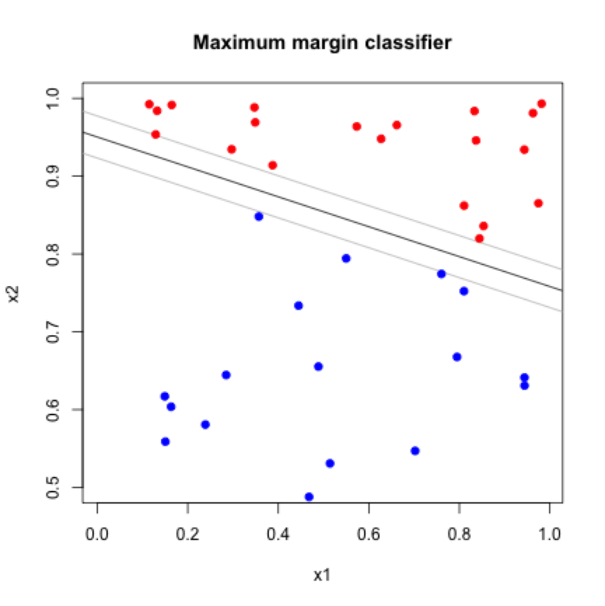
- For a given hyperplane, the **minimum margin** is

$$M_{\vec{w}} = \mathrm{argmin}_{x \in D} d_{\vec{w}}(\vec{x})$$

- The **maximum margin hyperplane** is therefore:

$$\mathrm{argmax}_{\vec{w}} \mathrm{argmin}_{x \in D} d_{\vec{w}}(\vec{x})$$

SVM illustration: SVM solution

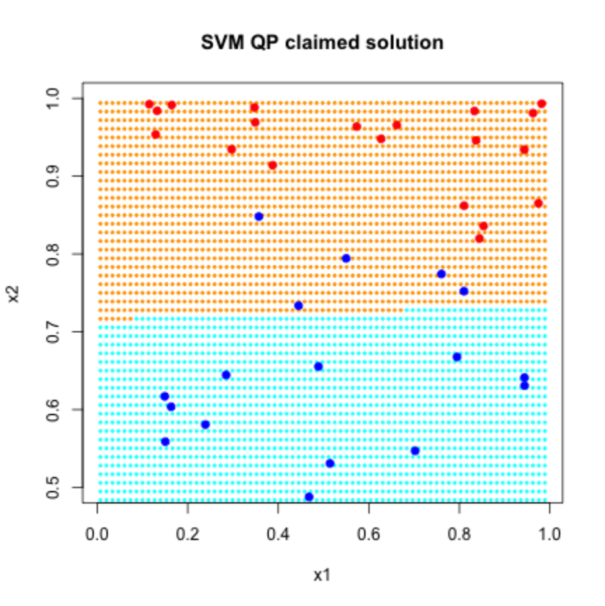


Computing the margins

- This is a classic **Quadratic Programming** problem[5]
- Broadly:
  - quadratic penalty: distance to the plane $\propto$ squared norm of the hyperplane vector $\frac{1}{2}\left|\vec{w}\right|^2$
  - linear inequalities: none of the data are closer than $M_{\vec{w}}$. So $\forall i : y_i(\vec{w} \cdot \vec{x} + b) \geq 1$
- and pass these to a standard QP solver
- A computational trick: only evaluate the points on the margins

[5] For this course, you need to know what QP can do for you. You don't need to know how it works.

SVM problem



Imperfect classification with SVM

- To account for data the **wrong side of the margins**, the penalty is changed to:

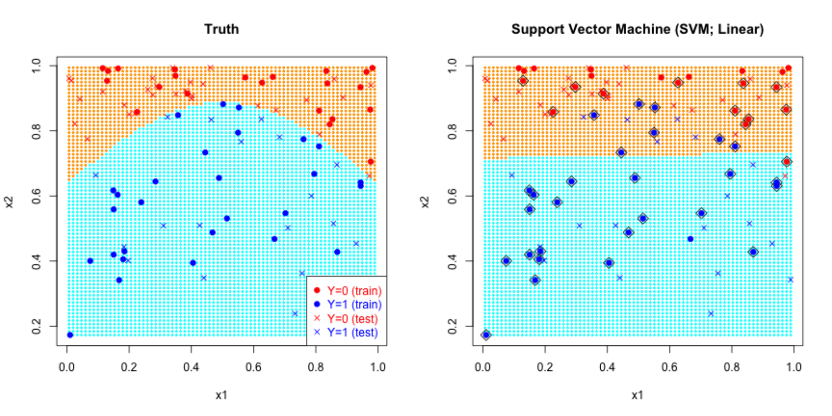
$$\frac{1}{2}|\vec{w}|^2 + C\sum_{i=1}^{N}\epsilon_i$$

- where ϵ_i is the "distance" needed to move the point to the correct decision boundary, i.e.

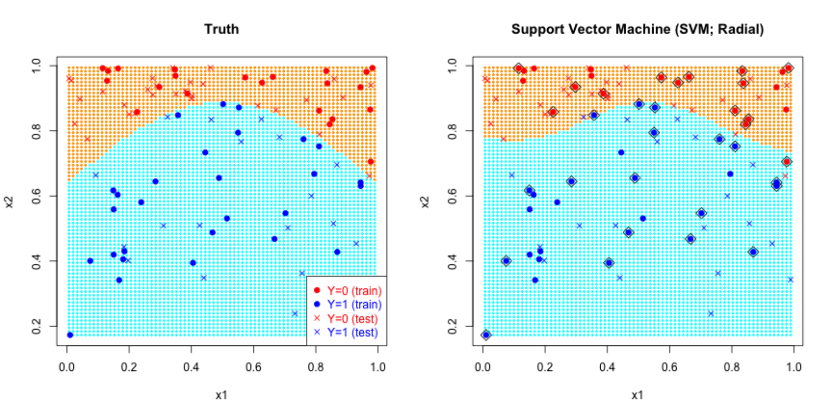
$$\begin{aligned} \vec{w}\cdot\vec{x}_i + b &\geq 1 - \epsilon_i & \text{if}: && y_i &= 1 & (1) \\ \vec{w}\cdot\vec{x}_i + b &\leq -1 + \epsilon_i & \text{if}: && y_i &= -1 & (2) \end{aligned}$$

- and $\epsilon_i = 0$ if already inside it, so also requiring the constraint $\epsilon_i \geq 0$

SVM example


Truth
Support Vector Machine (SVM; Linear)
x1
x2
0.0
0.2
0.4
0.6
0.8
1.0
Y=0 (train)
Y=1 (train)
Y=0 (test)
Y=1 (test)

kernel SVM example



Wrapup

- **Logistic regression** is the go-to straw man classifier in machine learning:
  - It is easy to implement
  - It is a natural predictive model
  - It does reasonably well in many settings
- **k-NN** is the interpolation method to beat
- **Linear Discriminant Analysis** is also widely used:
  - It is easy to bolt onto PCA
  - Clusters are more **interpretable** than logistic regression
- **SVMs** remain an important competitor at the bleeding edge:
  - A hyperplane is a natural **discriminatory model**
  - Feature engineering can allow complex **non-linear** models
  - Low-complexity classifier once training is performed
- **Neighbourhoods** are always competitive, but are costly at test time

Reflection

- Why is LDA used with PCA, and not instead-of?
- How would you imagine an approximate lookup for k-NN would work?
- How sparse should the SVM solution be? In what sense is SVM efficient? When would it be cutting edge?
- By the end of the course, you should:
  - Be able to navigate the many approaches to classification
  - Understand and be able to explain the high level function of:
  - Logistic Regression, Nearest Neighbour classification, LDA, SVMs

Signposting:

- In this Block's workshop we'll experiment with these and other classifiers on cyber data, as well as introducing **boosting**.
- In the following Block we'll introduce Random Forests, as well as boosted decision and regression trees. Naive Bayes comes in Block 7 with other Bayesian Methods.
- **References**:
- k-Nearest Neighbours:
  - Chapter 13.3 of The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Friedman, Hastie and Tibshirani).
- Linear Discriminant Analysis:
  - Sebastian Raschka's PCA vs LDA article with Python Examples
  - Chapter 4.3 of The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Friedman, Hastie and Tibshirani).
- SVMs:
  - Jason Weston's SVMs tutorial
  - e1071 Package for SVMs in R
  - Chapter 12 of The Elements of Statistical Learning: Data