

Introduction to Classification

Daniel Lawson University of Bristol

Lecture 05.1 (v3.0.0)



Chihuahua or muffin?

Signposting

- ▶ The remainder of the course changes the focus towards machine-learning - especially the background of the key tools that are used in practice.
- ▶ It is important to emphasise that classification is statistics, though we use the parlance of machine learning.
 - ▶ Most of machine learning is also modern statistics.
 - ▶ The main distinction is about purpose: whether developed for prediction, or for understanding.
- ▶ Today is about traditional “baseline models”. We discuss bleeding edge ensemble and tree methods next.

Questions

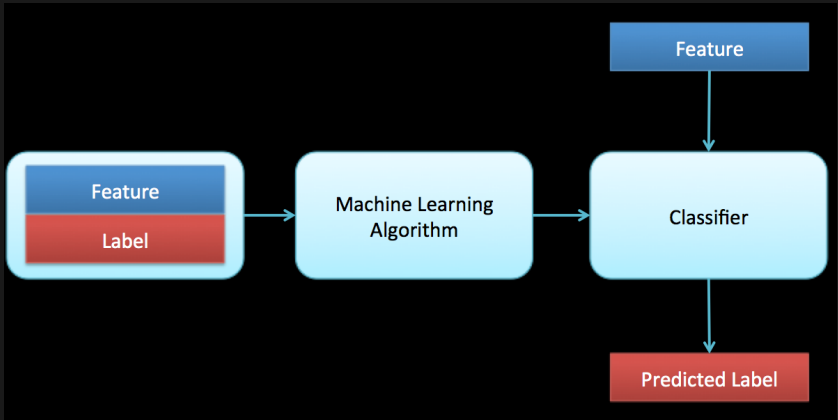
- ▶ Do we need a “baseline model”?
- ▶ How can we compare model performance?
- ▶ What is a “good classifier”?

Types of machine learning

- ▶ Unsupervised: **no labels**. For example,
 - ▶ Clustering
 - ▶ Dimensionality reduction
 - ▶ Smoothing
- ▶ Supervised: exploits **labels**. For example,
 - ▶ Classification
 - ▶ Regression
- ▶ Other types:
 - ▶ Semi-supervised: **some labels** are available
 - ▶ Active: can **choose which labels** to obtain
 - ▶ Reinforcement: **reward based**. explore vs exploit?
 - ▶ etc.

Classification

- ▶ Machine Learning classification is about how to make good predictions of **classes** based on previous experience of how **features** relate to **classes**.



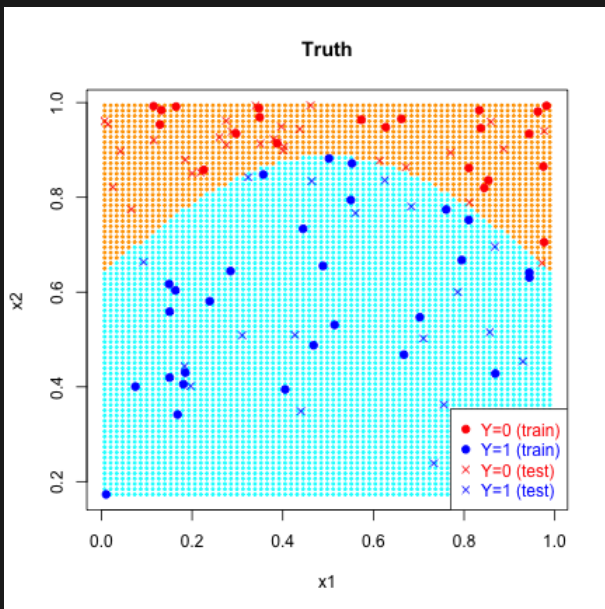
Examples of classification

- ▶ **Spam** filtering (spam/not spam)
- ▶ **Face detection** (image classification)
- ▶ **Speech recognition**
- ▶ **Handwriting recognition**
- ▶ **Turing test** . . . though that is human, not machine!
- ▶ Classification is broadly the “detection, recognition, recall of **prior experience**”.

Some Important Classifiers

- ▶ **Logistic Regression** (Block 2 and 5)
- ▶ **K-Nearest Neighbours** (Block 4 and 5)
- ▶ **Linear Discriminant Analysis** (Block 5)
- ▶ **Support Vector Machines** (Block 5)
- ▶ **Decision Trees** (Block 6)
- ▶ **CART**: Classification and Regression Trees (Block 6)
- ▶ **Random Forests** (Block 6)
- ▶ **Neural Networks** (Block 7)
- ▶ **Naive Bayes** (Block 8)

Classification



From Regression to Classification

- ▶ In **Block 02** we discussed **linear regression**, i.e. obtaining solutions to:

$$y_i = \vec{x}_i \cdot \beta + e_i$$

- ▶ Definitions: scalar form with p' covariates;
 $\vec{x}_i = (1, x_{1,i}, \dots, x_{p',i})$, so \vec{x}_i and β are vectors of length $p = p' + 1$; e_i are residuals whose squared-sum is minimised.
- ▶ Logistic regression estimates the probability that a binary outcome y is 1:

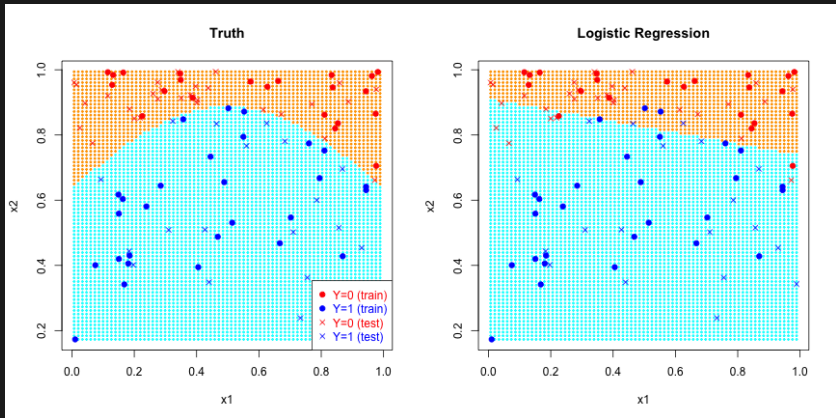
$$\text{logit}(p(y_i)) = \ln \left(\frac{p(y_i)}{1 - p(y_i)} \right) = \vec{x}_i \cdot \beta + e_i$$

- ▶ with $y_i \sim \text{Bern}(p(y_i))$. The prediction is the **log-odds** ratio, with values > 0 predicting a 1 and values < 0 predicting a 0.

Logistic Regression fitting

- ▶ Logistic regression is an example of a **generalised linear model** or **GLM**.
- ▶ In general these cannot be directly solved with Linear Algebra. Options include:
- ▶ **Maximum likelihood** estimation:
 - ▶ (Numerically) maximise the likelihood in terms of the parameters β , and σ the variance of e .
- ▶ Iteratively Reweighted **least squares** (IRLS):
 - ▶ The non-linearity is adopted into weights, and a linear algebra solution reached.
 - ▶ Then the weights are updated, and the procedure iterated.
- ▶ Co-estimation is relatively computationally costly (higher dimensional space) but has better estimation properties.

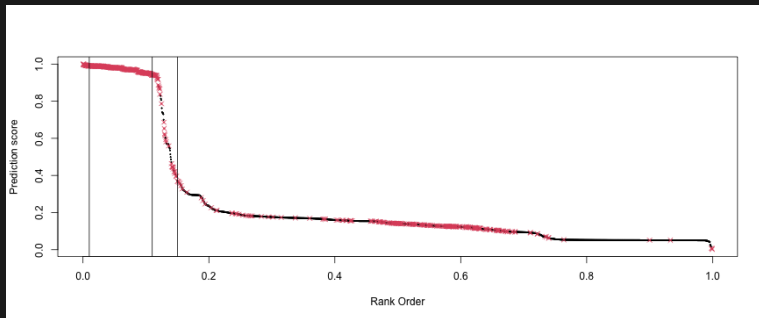
Logistic Regression example



Classification Performance

- ▶ We can always compare performance on **test** data, to prevent over-fitting.
- ▶ Classifiers are understood through their **Confusion Matrix**, that is a comparison between:
 - ▶ Ground truth class, and
 - ▶ Predicted classes.
- ▶ For binary classes, we summarise using (true/false)(positive/negative) outcomes.
- ▶ Binary classification is particularly convenient as most classifiers can provide **scores** rather than **class predictions**.
 - ▶ Scores are **ordered**. So we can choose a threshold to control the total proportion of **positive predictions**.
 - ▶ This provides a **relationship** between **Positive Claims** and **True Positives**.

Classification Performance Thresholding



	$Y = 1$	$Y = 0$	Condition
$\hat{Y} = 1$	TP	FP	Prediction positive
$\hat{Y} = 0$	FN	TN	Prediction negative
Claim	Truth positive	Truth Negative	

Classification Performance Representations

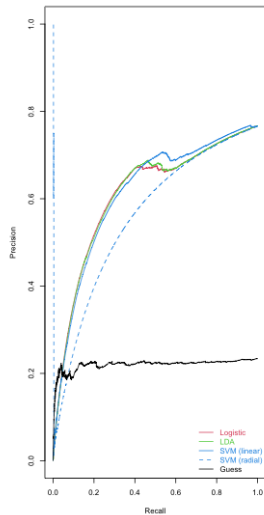
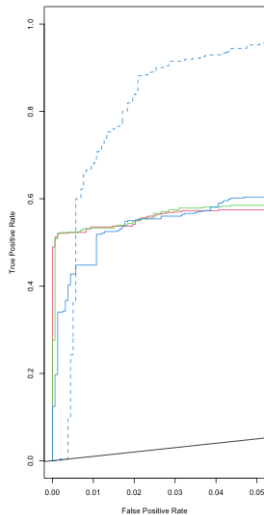
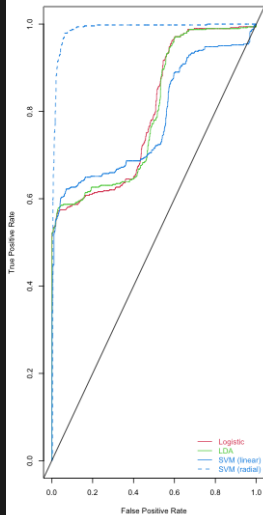
- ▶ There are many ways to represent performance
- ▶ The **Receiver-Operator-Curve (ROC)** is the most popular, as it holds regardless of the true distribution of the data.
 - ▶ X-axis: False Positive Rate (FPR) = $P(\hat{Y} = 1|Y = 0)$
 - ▶ Y-axis: True Positive Rate (TPR) = $P(\hat{Y} = 1|Y = 1)$
 - ▶ The **Area Under the Curve (AUC)** is a measure of Accuracy (0.5=guessing, 1=perfect).
 - ▶ We need to care about the region of the ROC curve that matters.
- ▶ The **Precision-Recall curve** is appropriate when we care specifically about positive cases:
 - ▶ X-axis: Precision = $P(Y = 1|\hat{Y} = 1)$
 - ▶ Y-axis: Recall=TPR = $P(\hat{Y} = 1|Y = 1)$

Some important properties

- ▶ Some nice things¹ can be said about ROC and PR curves:
- ▶ Dominance:
 - ▶ If one curve dominates (is always above) another in ROC, it dominates in PR
 - ▶ and vice-versa
- ▶ ROC curves can be linearly interpolated
 - ▶ This is “flipping a coin” to access classifiers in-between
- ▶ PR curves have a slightly more complex relationship but the same principle can be applied
- ▶ Integrating both scores leads to performance metric that can be optimized

¹Davis and Goadrich, “The Relationship Between Precision-Recall and ROC Curves”, ICML 2006.

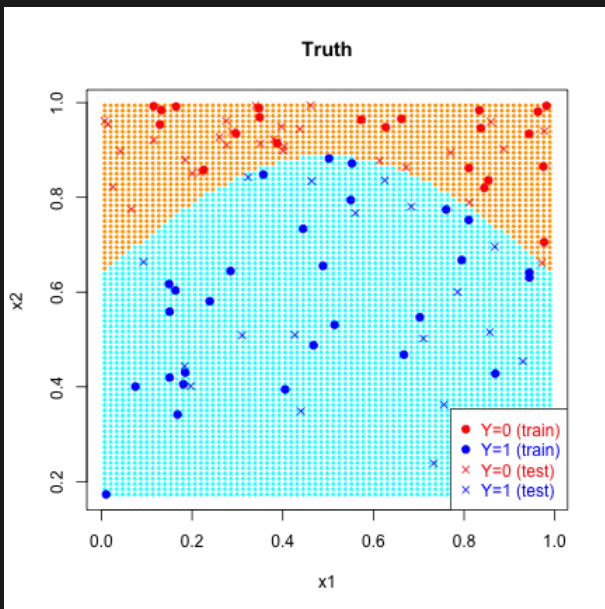
ROC/PR Curve Example



Metrics for Classification

- ▶ Accuracy (Proportion of samples classified correctly) is a **terrible metric** if classes are unequal
 - ▶ TPR at a given FPR is more flexible
- ▶ AUC characterises the whole ROC curve
- ▶ Area Under Precision-Recall Curve (AUPRC?) is also a thing people advocate for
- ▶ None are “right”, we have to define the inference task
- ▶ Any of these and more are often optimized
 - ▶ If we optimise a parameter or perform model comparison based on test data, we need additional test data to test the meta-algorithm!

Classification

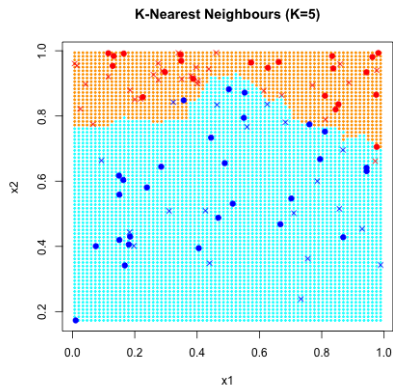
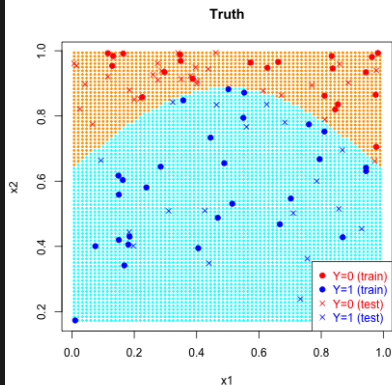


K-Nearest Neighbour classification

- ▶ 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 neighbor² approaches to reduce the amount of data required at the classification stage.

²Hart 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



LDA and SVM

- ▶ Linear Discriminant Analysis (LDA) was developed in **1936 by R. A. Fisher**³ and extended to the current multi-class form in 1948⁴.
 - ▶ **Project** a high dimensional space into K dimensions, **maintaining** (linear) classification ability.
- ▶ Support Vector Machines (SVM) choose a decision boundary to maximally separate the classes
 - ▶ Uses Quadratic Programming, a widely useful tool
- ▶ Performance lags modern approaches so we'll just use them as alternative linear classifiers
 - ▶ Kernel versions also exist, occasionally competitive

³Fisher R, "The Use of Multiple Measurements in Taxonomic Problems" (1936) Annals of eugenics (!), now "Annals of Human Genetics"

⁴Rao C, "Multiple Discriminant Analysis" (1948) JRSSB

Wrapup

- ▶ These are **baseline models**
- ▶ **Logistic regression** is the go-to straw man classifier in machine learning:
 - ▶ It is a natural predictive model
 - ▶ It does surprisingly well in many settings
- ▶ **Linear Discriminant Analysis** and **SVMs** are still used
- ▶ **k-NN** is the interpolation method to beat
 - ▶ **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

References:

- ▶ **References** for classification basics:
 - ▶ Stack Exchange Discussion of ROC vs PR curves.
 - ▶ Davis and Goadrich, "The Relationship Between Precision-Recall and ROC Curves", ICML 2006.
 - ▶ Rob Schapire's ML Classification features a Batman Example. . .
 - ▶ Chapter 4 of *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Friedman, Hastie and Tibshirani).
- ▶ 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*

Bonus material:

- ▶ Linear Discriminant Analysis and
- ▶ Support Vector Machines

Linear Discriminant Analysis

- ▶ Developed in **1936 by R. A. Fisher**⁵ and extended to the current multi-class form in 1948⁶.
- ▶ 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 exists⁷

⁵Fisher R, "The Use of Multiple Measurements in Taxonomic Problems" (1936) Annals of eugenics (!), now "Annals of Human Genetics"

⁶Rao C, "Multiple Discriminant Analysis" (1948) JRSSB

⁷Mika, S et al "Fisher discriminant analysis with kernels" (1999) NIPS IX:

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

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

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

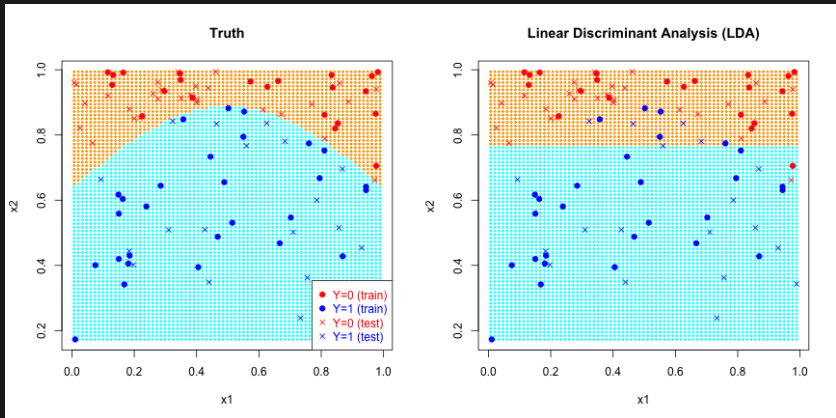
$$S_B = \sum_{k=1}^K n_i (\vec{\mu}_k - \vec{\mu}) (\vec{\mu}_k - \vec{\mu})^T$$

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 $\mu_k \dots$

LDA prediction

- ▶ Class prediction can use any information in the LDA data summary. Options include:
 - ▶ Nearest cluster
 - ▶ **Likelihood**: $\Pr(\vec{x}|y_k = c) = \text{Normal}(\mu_k, \Sigma)$
 - ▶ **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



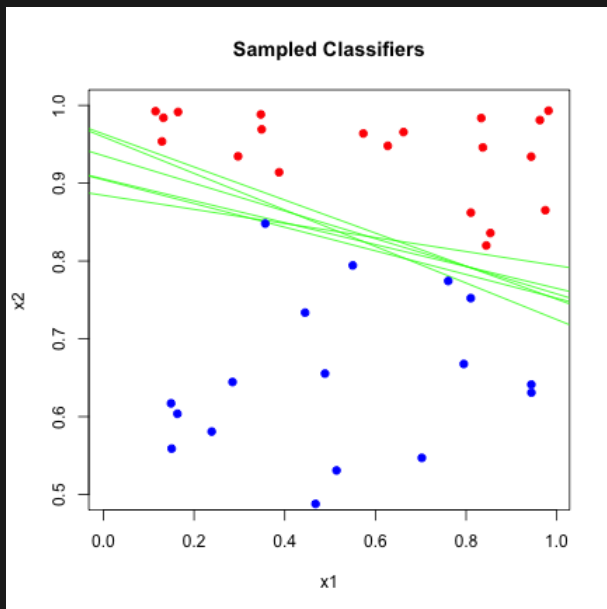
Towards Support Vector Machines

- ▶ LDA uses **all the points** for classification, which makes it slow
- ▶ It is obviously **linear** (without using a kernel)
- ▶ 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”.

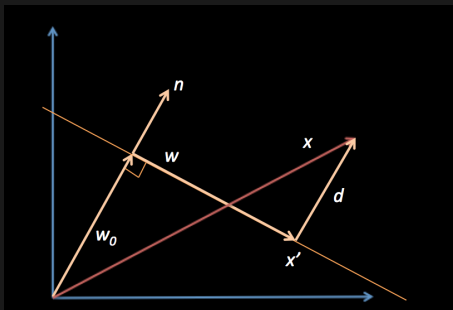
SVM illustration: solution space



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:

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

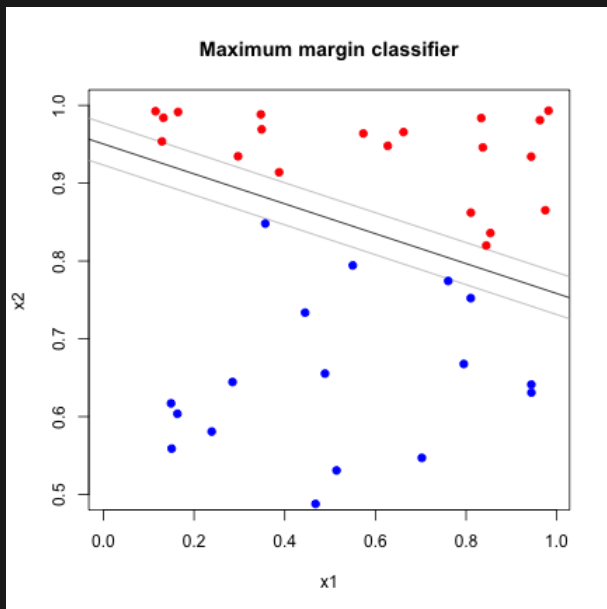
- ▶ For a given hyperplane, the **minimum margin** is

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

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

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

SVM illustration: SVM solution

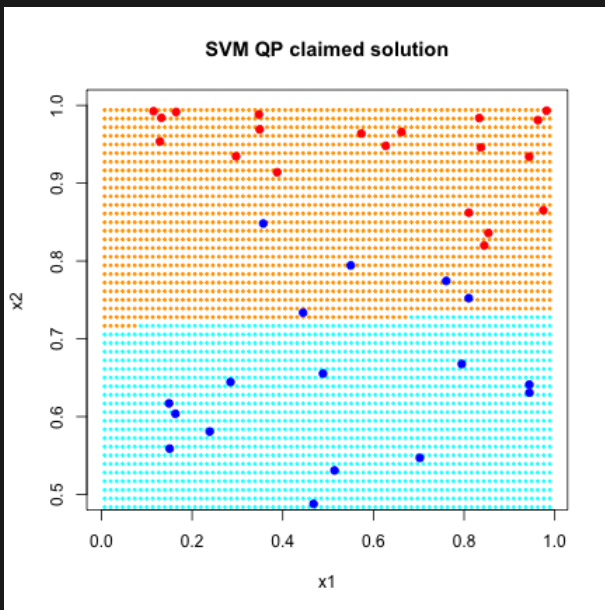


Computing the margins

- ▶ This is a classic **Quadratic Programming** problem⁸
- ▶ Broadly:
 - ▶ quadratic penalty: distance to the plane \propto squared norm of the hyperplane vector $\frac{1}{2} \|\vec{w}\|^2$
 - ▶ 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

⁸For 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:

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

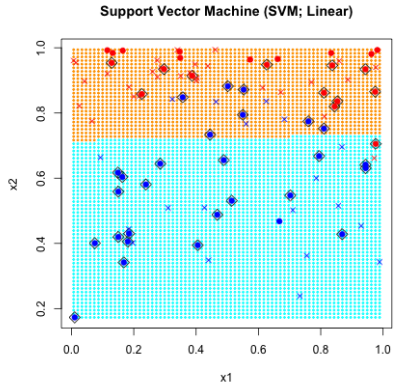
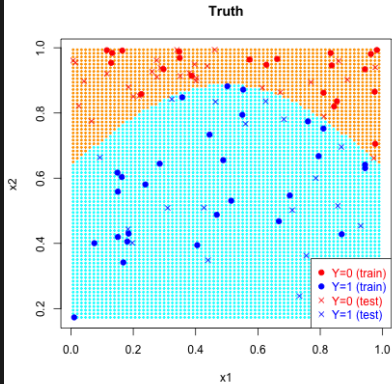
- ▶ where ϵ_i is the “distance” needed to move the point to the correct decision boundary, i.e.

$$\vec{w} \cdot \vec{x}_i + b \geq 1 - \epsilon_i \quad \text{if :} \quad y_i = 1 \quad (1)$$

$$\vec{w} \cdot \vec{x}_i + b \leq -1 + \epsilon_i \quad \text{if :} \quad y_i = -1 \quad (2)$$

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

SVM example



kernel SVM example

