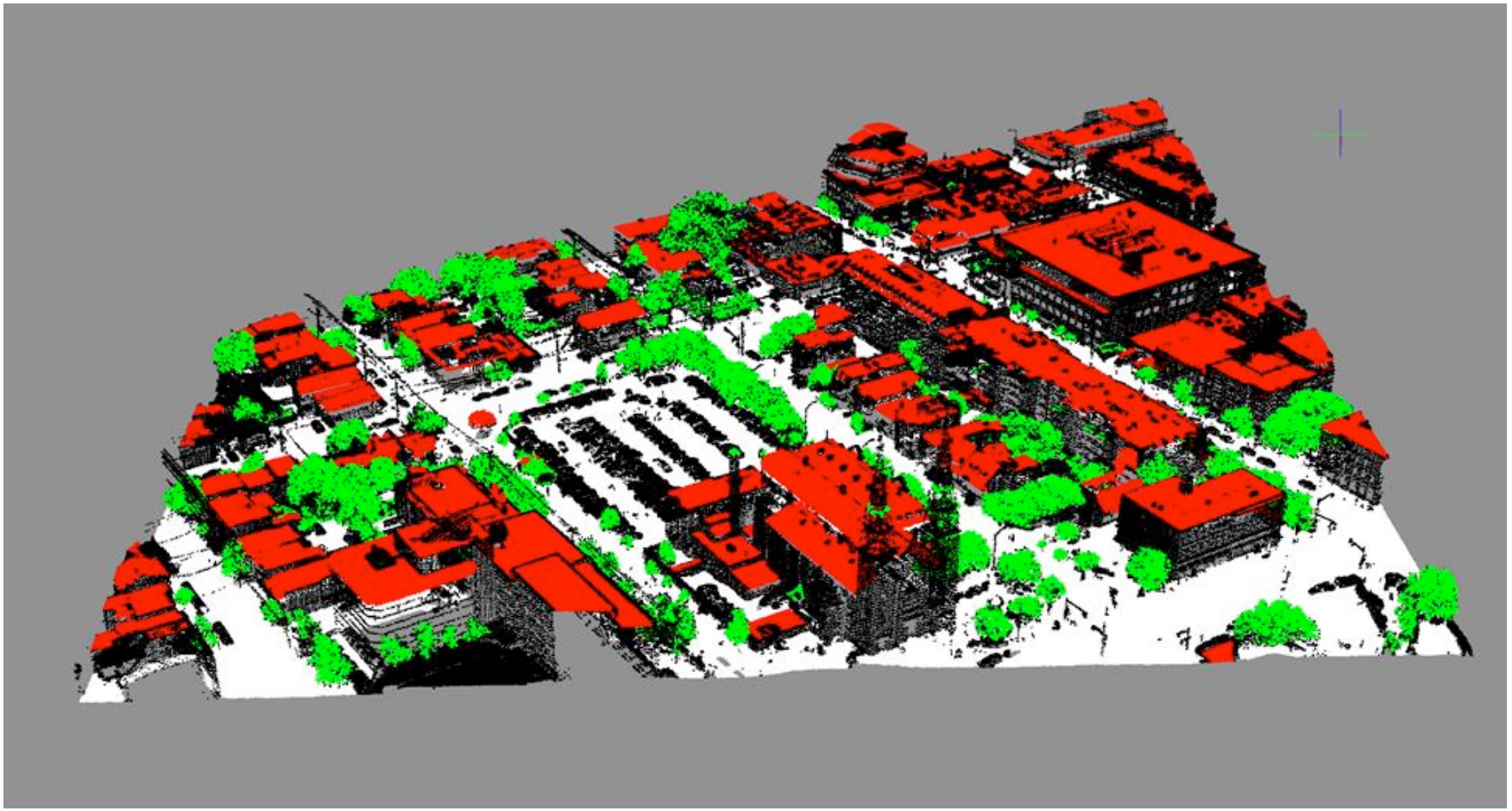
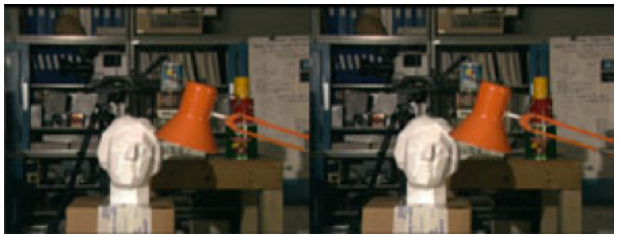
Classification

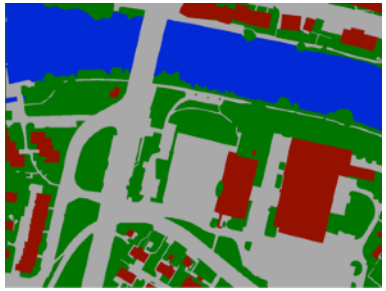


Classification

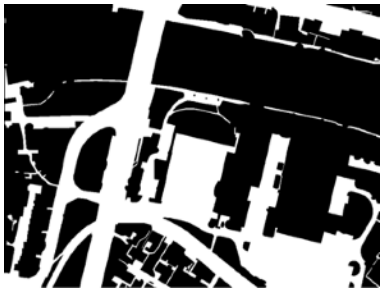
- Given input data, predict a discrete quantity
  - (as opposed to regression, which predicts continuous quantities)
- Discrete output set can be
  - ordered - like regression, but quantised output space
  - unordered - nominal labels, e.g. semantic object types



stereo parallax
1px, 2px, 3px, ...



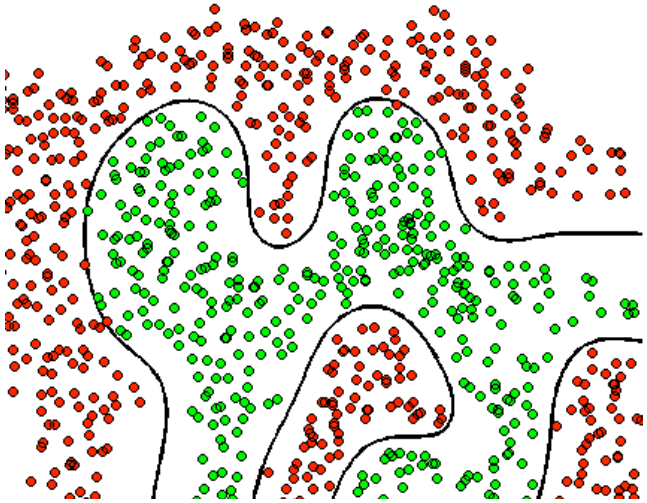
building, road
vegetation, water

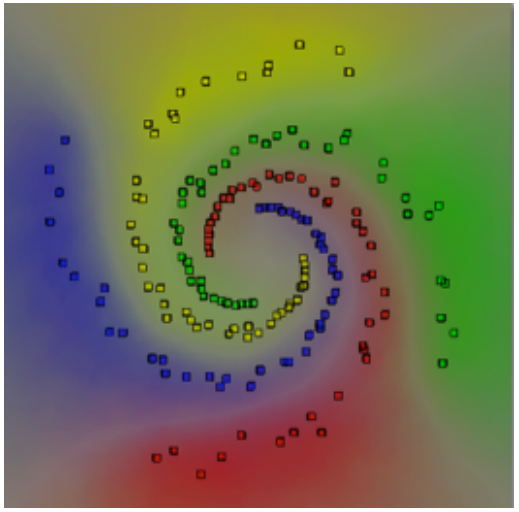


road / non-road

Classification

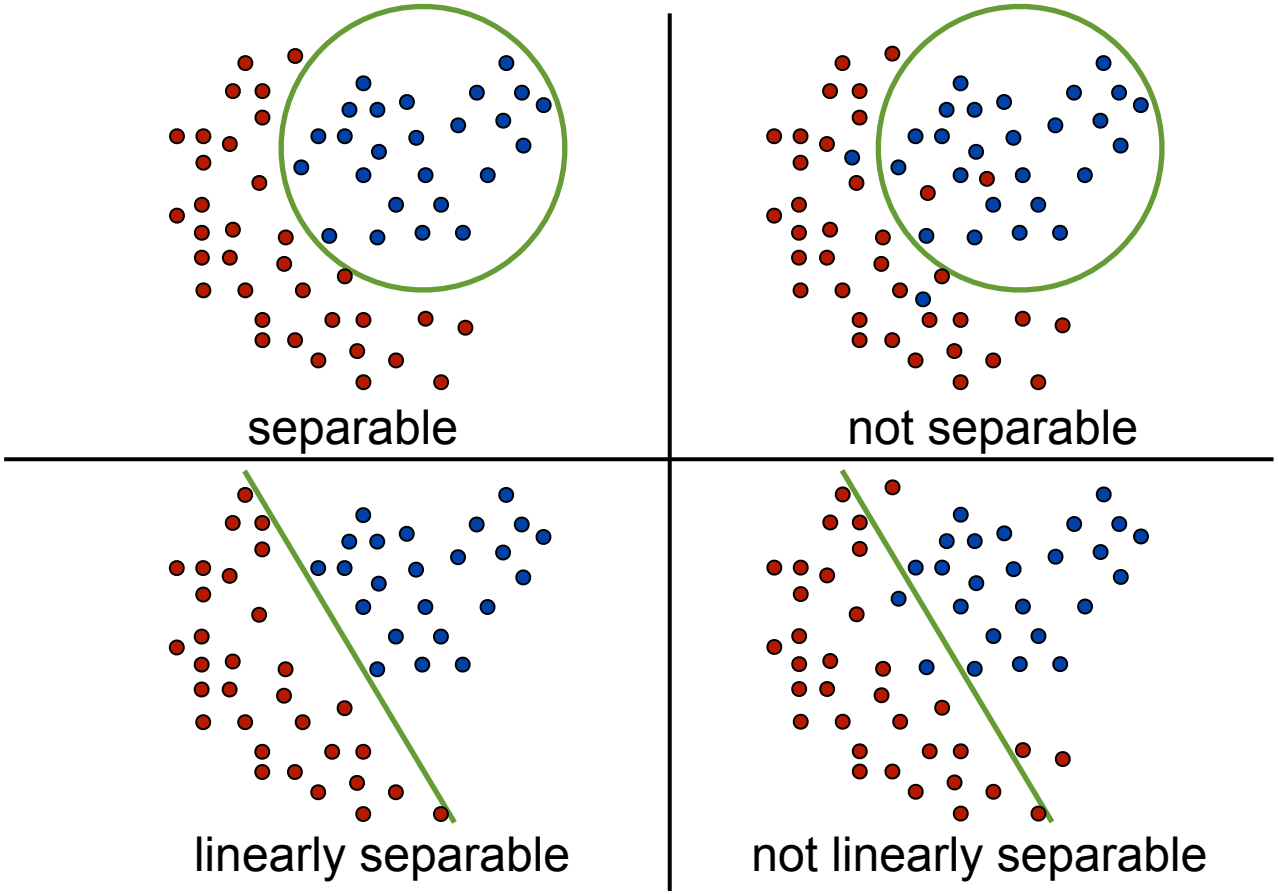
- Formally: predict discrete label *y* from data (features) **x**
  - Assumption: classes are non-overlapping, only one label per input data point
- We are looking for decision boundaries, which split the feature space into class regions
  - Ideally, the classifier should return not only a hard label, but also a confidence value, respectively a probability for each possible label





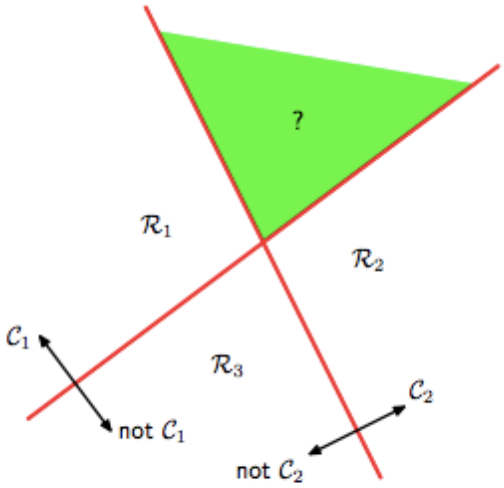
Separability

- Basic cases
  - inputs are separable with the chosen decision function
  - inputs are not separable, some points are bound to be misclassified
  - for linear boundaries: inputs are (or are not) linearly separable

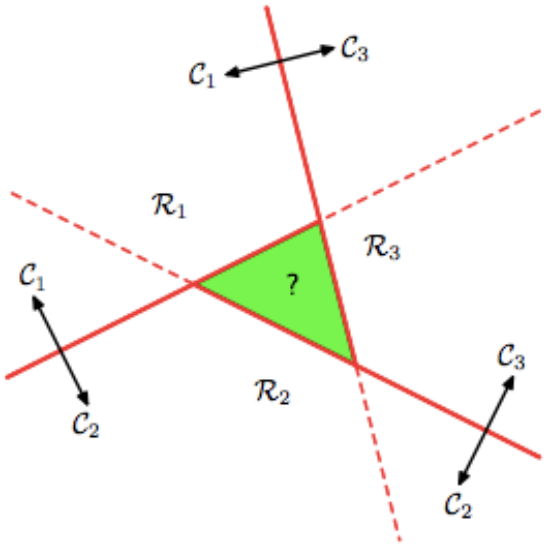


Multi-class separation

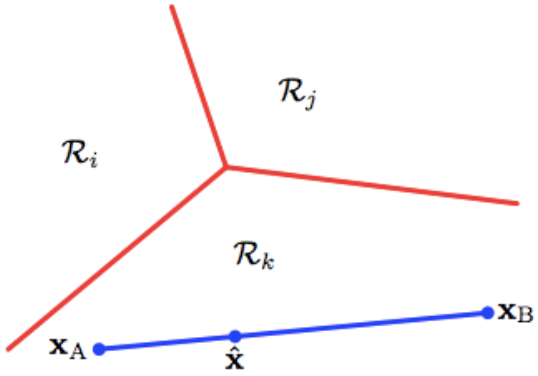
- Can we construct a multi-class classifier from binary decision boundaries?
- Need confidence values (ideally class probabilities), then we can choose the class with the highest value



one-versus-all
(one-versus-the-rest)



one-versus-one

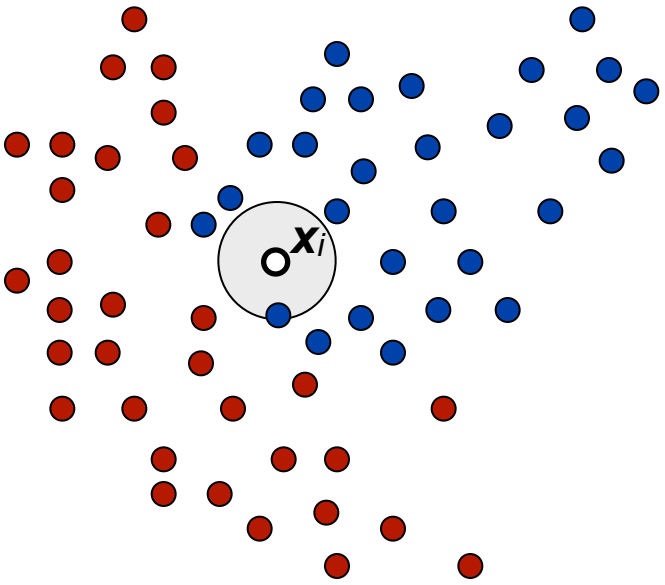


If the discriminant functions are linear, the decision regions are connected and convex

Basic methods

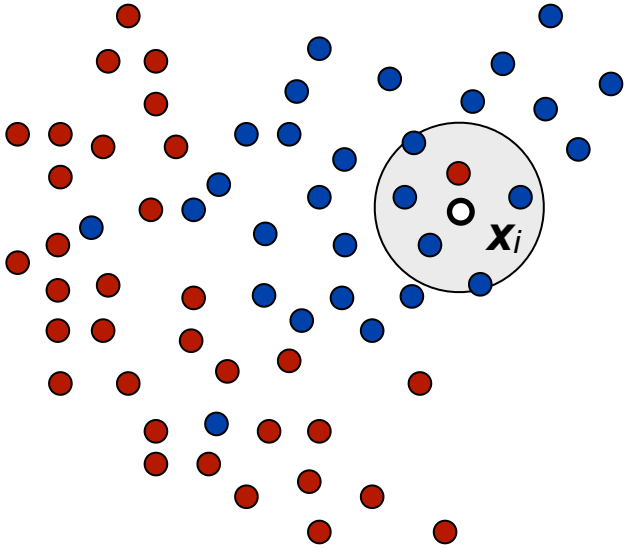
Nearest neighbours

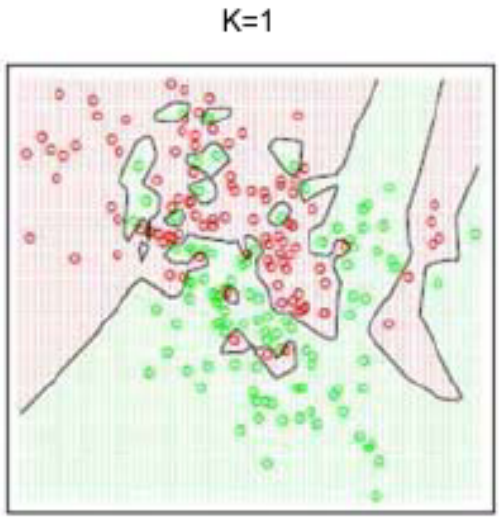
- Obvious idea: nearest-neighbour classifier
- To find the label of a new data point $\mathbf{x}_i$
  - find the training point $\{y_t, \mathbf{x}_t\}$ that is closest (most similar) to $\mathbf{x}_i$
  - assign $\mathbf{x}_i$ the same label: $y_i = y_t$

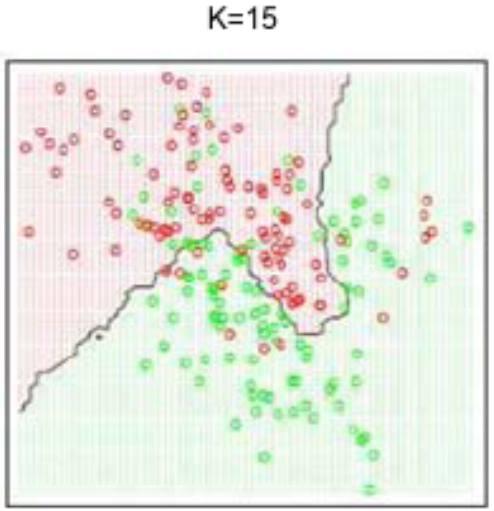


k-nearest neighbours

- Extension: *k*-nearest neighbours
  - letting a single training point decide is not reliable
  - instead, find a set of of *k* nearest neighbours
  - among those neighbours, vote for the label y_i
  - nearby points will have many common neighbours, hence decision boundaries become smoother with higher *k*







k-nearest neighbours

- Probabilistic interpretation
  - we are searching for probabilities *Pr*(*y*|**x**) of different labels *y* at a data point (feature vector) **x**
  - Let us look at a small region in feature space with volume *V*

- probability that a data point falls into the region: P_{inside}

- thus, number of data points inside the region: $K \approx N \cdot P_{inside}$

- density in a small region is roughly constant, so $Pr(\mathbf{x}) = \frac{P_{inside}}{V} = \frac{K}{N \cdot V}$

- among the points in the region, K_k have label *k*: $Pr(\mathbf{x}|y=k) = \frac{K_k}{N_k \cdot V}$

- prior probability for class *k*: $Pr(y) = \frac{N_k}{N}$

by Bayes' law

$$Pr(y=k|\mathbf{x}) = \frac{Pr(\mathbf{x}|y=k)Pr(y=k)}{Pr(\mathbf{x})} = \frac{K_k}{K}$$

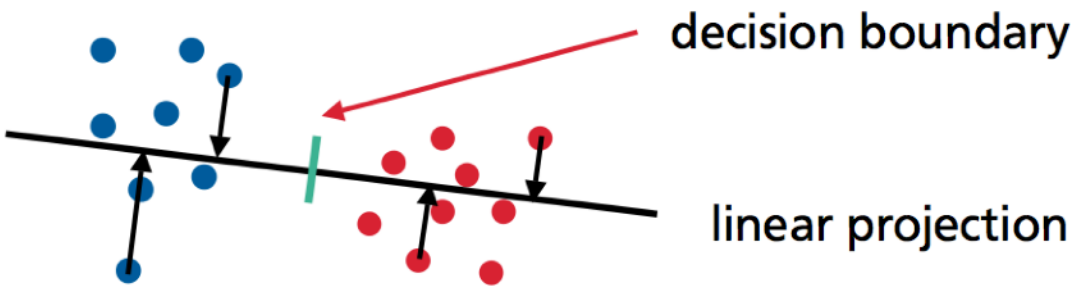
→ the class with the highest *Pr*(*y*) is the one with the highest number K_k of points in the region

k-nearest neighbours

- Pros
  - simple - no model fitting required
  - adapts well to different data - no choice of mapping function (respectively, probability density function) required
  - only one parameter *k* to set
  - works in the same way with different distance functions
  - works unchanged for multiclass problems
- Cons
  - needs sufficient training data - in empty regions of the feature space the nearest neighbours are relatively random
  - does not scale well: need to store and search all training data every time we classify a point (can be solved to some degree with efficient approximate nearest neighbour search structures)
- Note
  - *k*NN is discriminative, does not recover the distribution of the data

Linear discriminant

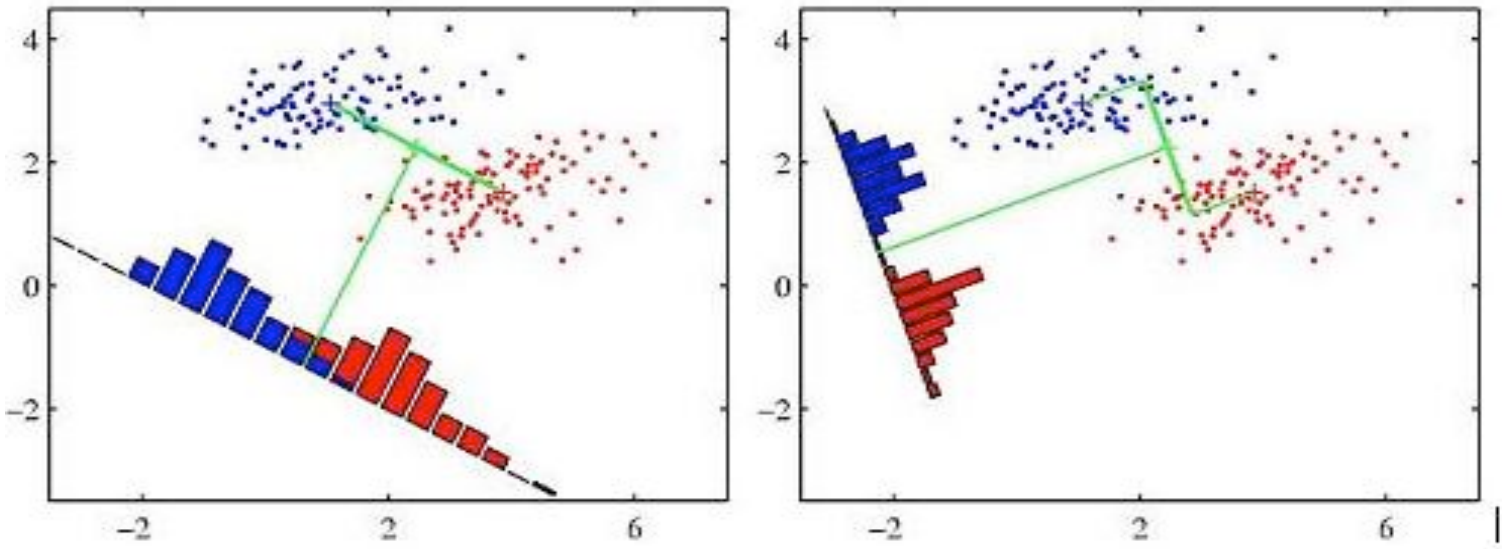
- Our goal is to find the decision boundary
- Let's look at the simplest case
  - two classes
  - linear decision boundary
  - equivalent to linearly projecting the data onto the normal of the decision boundary, and thresholding the result in 1D
  - projection: $y = \boldsymbol{\theta}^\top \mathbf{x}$
  - thresholding: $y > -\theta_0 \quad , \quad \boldsymbol{\theta}^\top \mathbf{x} + \theta_0 > 0$



Linear discriminant

- Idea: represent each class by its centroid
  - find line connecting the centroids
  - set the decision boundary orthogonal to that line

- Not a good idea, even if classes are linearly separable



Linear discriminant

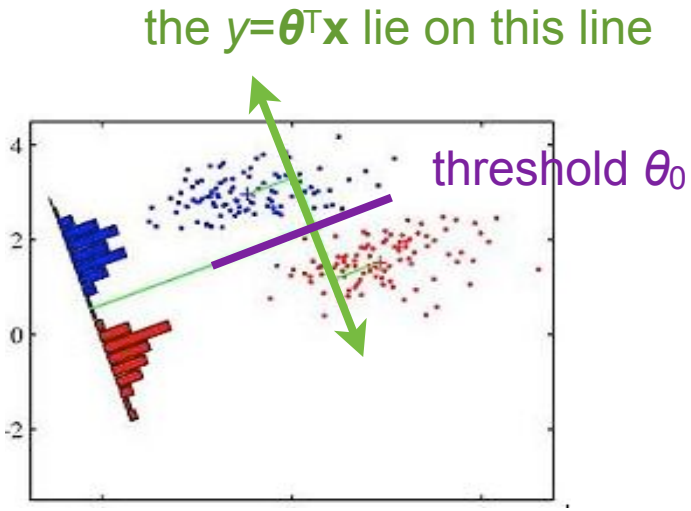
- Better idea: linear discriminant analysis (LDA)
  - project such that the variance between the classes is high and the variance within each class is low
  - Covariance matrix between classes

$$\mathbf{S}_B = \sum (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^\top$$

  - sum of within-class covariance matrices

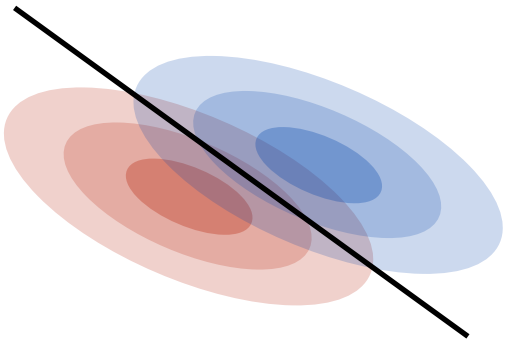
$$\mathbf{S}_W = \sum_{i \in \mathcal{C}_1} (\mathbf{x}_i - \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1)^\top + \sum_{i \in \mathcal{C}_2} (\mathbf{x}_i - \mathbf{m}_2)(\mathbf{x}_i - \mathbf{m}_2)^\top$$

  - maximize $\dfrac{\boldsymbol{\theta}^\top \mathbf{S}_B \boldsymbol{\theta}}{\boldsymbol{\theta}^\top \mathbf{S}_W \boldsymbol{\theta}}$
  - leads to $\boldsymbol{\theta} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$



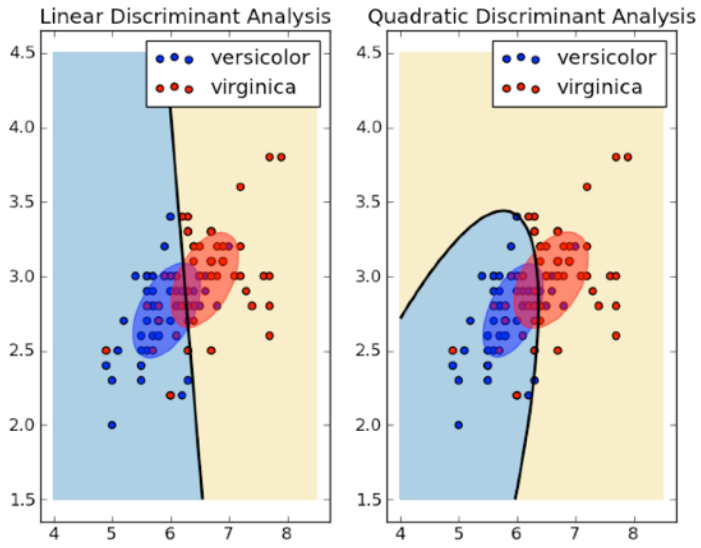
Linear discriminant

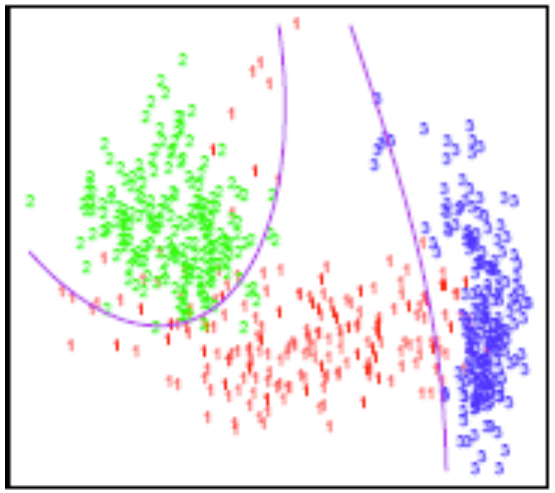
- Pro
  - distribution-free: we need not make assumptions about the data, as long as it is linearly separable
  - few parameters (*m* weights for m feature dimensions), hence few training data points needed
- Con
  - restricted to linear decision boundaries
  - without assuming distributions, no probabilistic interpretation
- Note
  - if we assume the two classes are normally distributed and have the same covariances, LDA gives the maximum-likelihood decision
  - ...but LDA is more efficient, you need not compute the likelihood
  - the Gaussian assumption turns LDA into a generative model, one defines $Pr(\mathbf{x}|y)$
  - thus it is possible to include priors $Pr(y)$



Quadratic discriminant analysis (QDA)

- Take the idea further: each class is normally distributed with an individual covariance matrix
  - It can be shown that the decision boundaries then are quadratic
  - There is no shortcut, you need to fit each of the Gaussians
  - this requires enough data to estimate each of the covariance matrices
  - in remote sensing sometimes called “Maximum Likelihood Classifier”





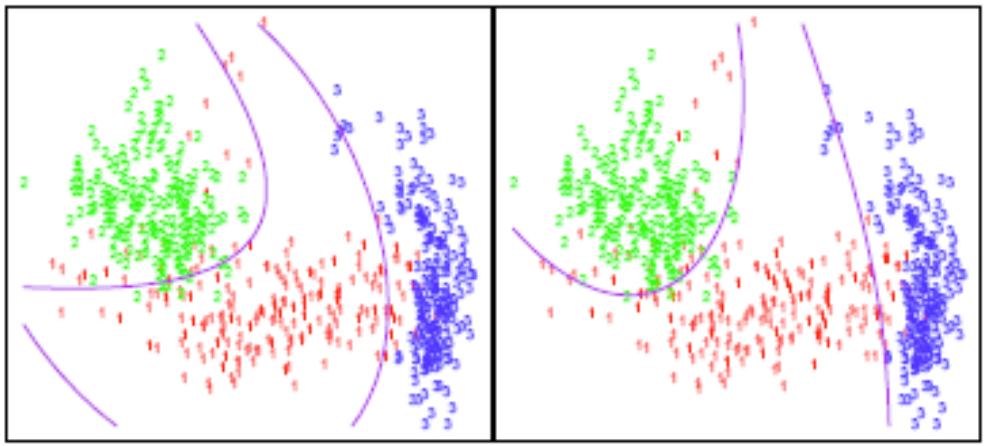
Quadratic discriminant analysis

- Pro
  - individual covariance per class a lot more realistic
  - still relatively easy, just fitting Gaussians
  - works unchanged for multiclass problems
- Con
  - will typically not work well for non-Gaussian distributions
  - curse of dimensionality: with m feature dimensions, need to estimate $m(m+1)/2$ entries for the covariance matrix of each class
- Note
  - although it is called “discriminant analysis” QDA is a generative model: one defines the $Pr(\mathbf{x}|y)$ to be Gaussians
  - thus, it is easy to include prior assumptions $Pr(y)$

Non-linear lifting

- LDA can be extended to more complex decision boundaries using basis functions
  - exactly the same as with linear regression
  - fit linear combination of arbitrary basis functions
  - still linear in the weights
- Example: quadratic decision boundaries in 2D
  - does not give the same result as QDA, usually differences are small

$$\mathbf{z}(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 & x_1 x_2 & x_1^2 & x_2^2 \end{bmatrix}$$



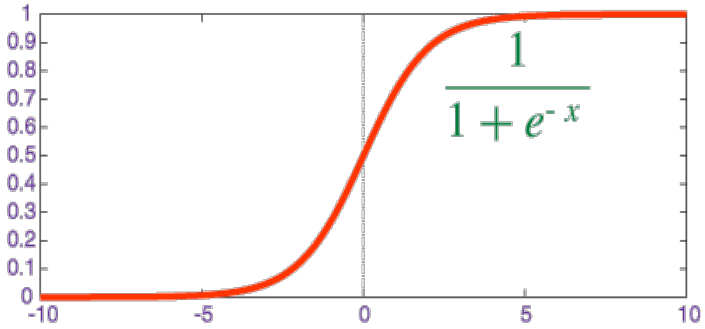
LDA with lifting QDA

Logistic regression

- Let's take a discriminative viewpoint
  - what we are looking for is the posterior $Pr(y|\mathbf{x})$
  - for two classes, the only possible outcomes are $y=f(\mathbf{x})=\{0,1\}$
  - over these we can define a Bernoulli distribution

    $Pr(y=1)=\lambda \qquad Pr(y)=\mathcal{B}_y[\lambda]=\lambda^y(1-\lambda)^{1-y}$
  - so we need to find λ as a function of data $\mathbf{x}$
  - a linear relation will not work, it returns values $\lambda \in [-\infty,+\infty]$
- Use linear model, transform output with a sigmoid

$$Pr(y|\boldsymbol{\theta},\mathbf{x})=\mathcal{B}_y\left[\frac{1}{1+e^{-\boldsymbol{\theta}^\top\mathbf{x}}}\right]$$

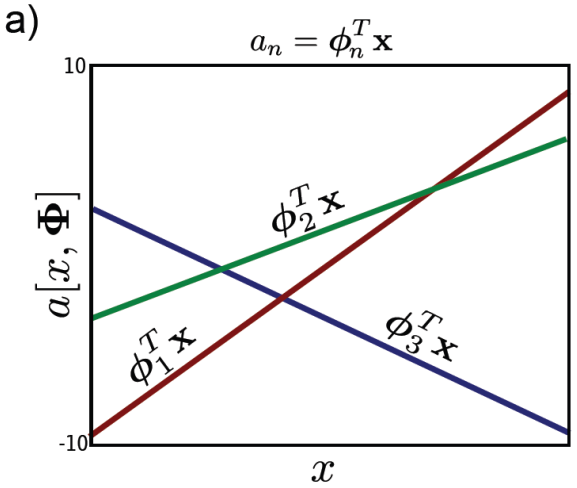


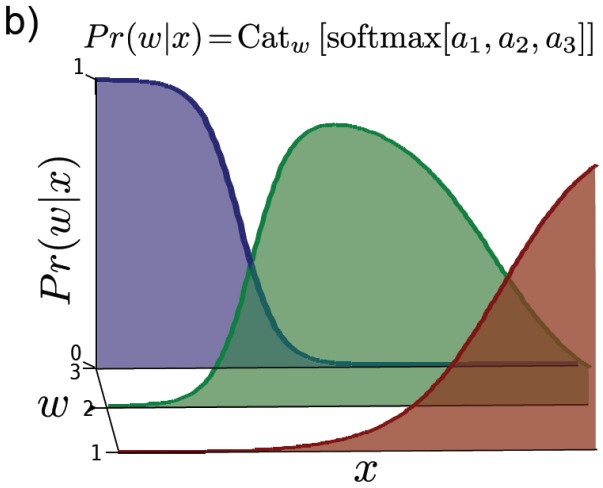
- Note: confusing name - a method for classification, not regression

Logistic regression

- Maximum likelihood fitting
  - there is no pretty closed-form solution for $\max_{\boldsymbol{\theta}} \left(\log \prod_{i=1}^{I} \mathcal{B}_{y_i} \left(\frac{1}{1+e^{-\boldsymbol{\theta}^\top \mathbf{x}_i}} \right) \right)$
  - still, it is a concave function with a single maximum, which can be found with iterative optimisation
- Extension to multiple classes
  - map output of linear models to class probabilities that sum to 1
  - needs multi-class version of the sigmoid, so-called softmax function

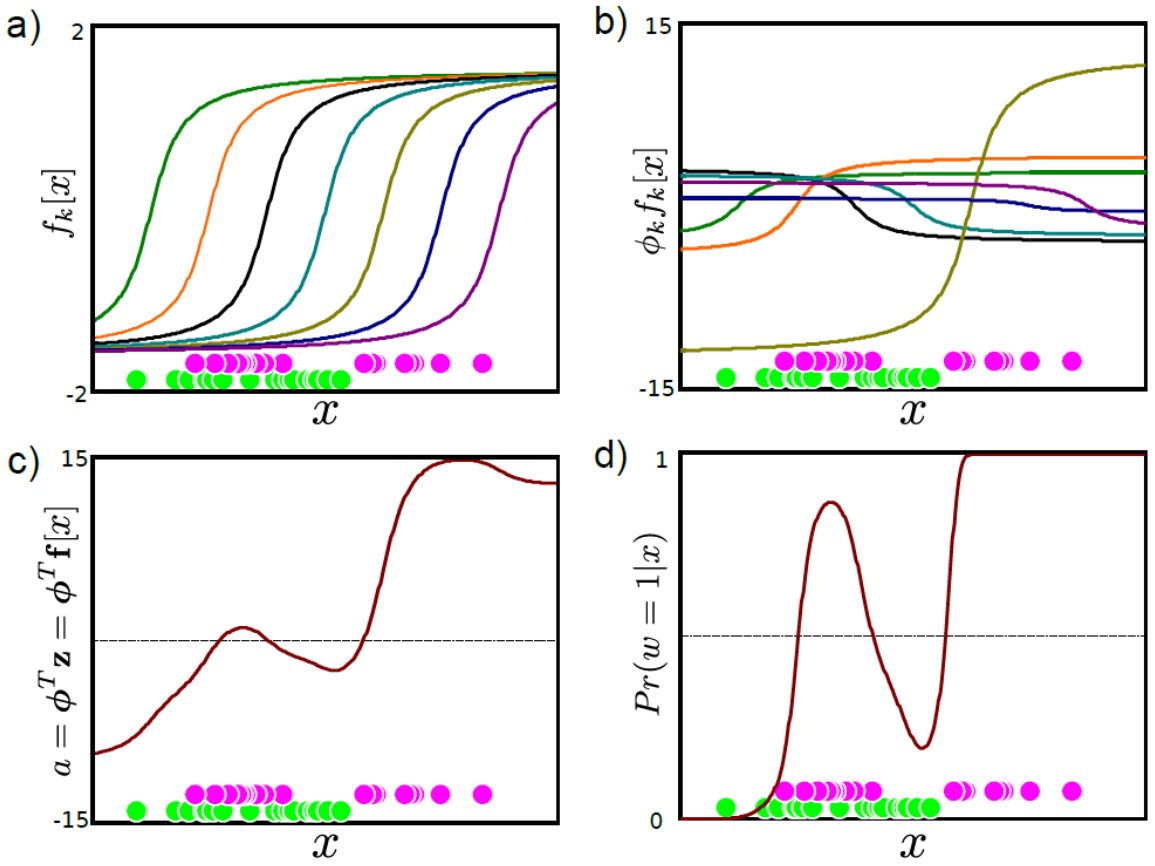
$$Pr(y = k|\mathbf{x}, \boldsymbol{\theta}_1 \ldots \boldsymbol{\theta}_K) = \frac{e^{-\boldsymbol{\theta}_k^\top \mathbf{x}}}{\sum_{i=1}^{K} e^{-\boldsymbol{\theta}_i^\top \mathbf{x}}}$$





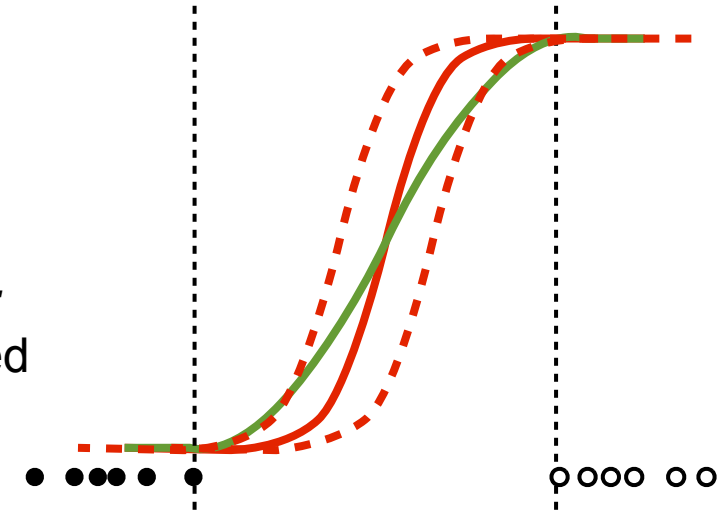
Logistic regression

- Extension to non-linear boundaries
  - as usual, the lifting trick applies: transform input data with non-linear basis functions **z**(**x**), fit a linear combination $\boldsymbol{\theta}^{\mathsf{T}}\mathbf{z}(\mathbf{x})$
- Example: arctangent basis functions



Logistic regression

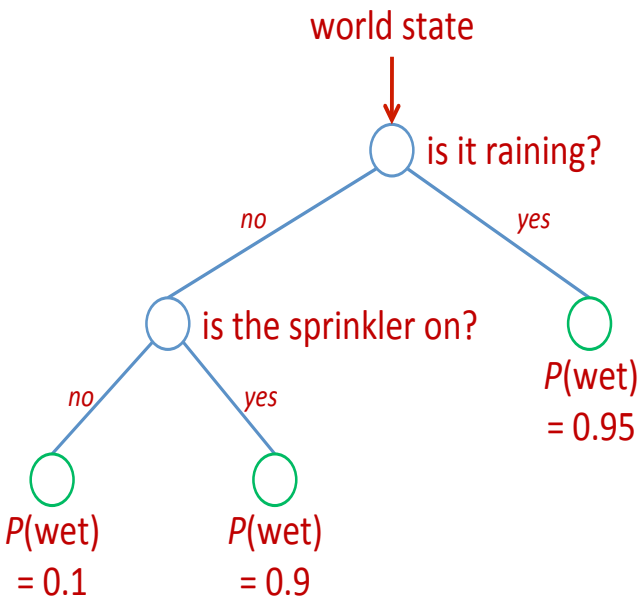
- Pro
  - probabilistically principled, builds on discrete distributions
  - no need to know/assume the data distribution
  - few parameters: binary logistic regression in a m-dimensional feature space has m unknowns; fitting even two simple Gaussians would have $2m+m(m+1)/2$ unknowns
- Con
  - tendency to overfit simple problems: if inputs $\mathbf{z}(\mathbf{x})$ are linearly separable, the sigmoid can become arbitrarily steep
  - overfitting can be prevented with a (e.g. Gaussian) prior $Pr(\boldsymbol{\theta})$ on the parameters***;*** but parameter learning can only be solved approximately, and gets rather messy

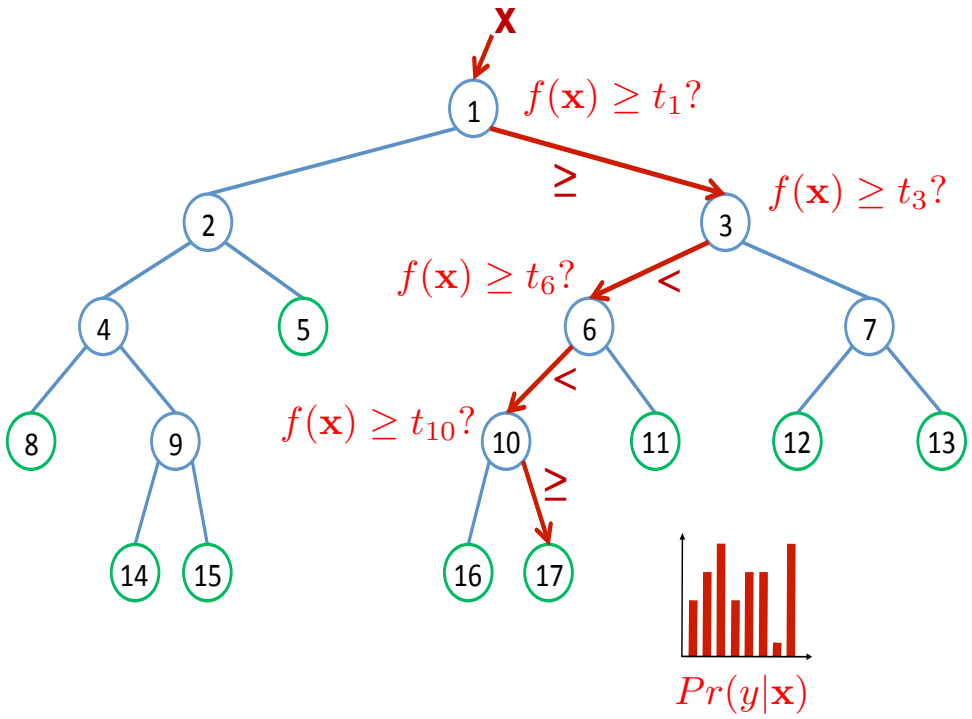


Further methods

Decision trees

- We have already seen regression trees
- In fact, trees are even more natural for classification
  - take a sequence of binary decisions until the class is clear
  - in a probabilistic world: binary splits to make *Pr*(*y*|**x**) more peaked
  - note: trees were first used for classification





Decision trees

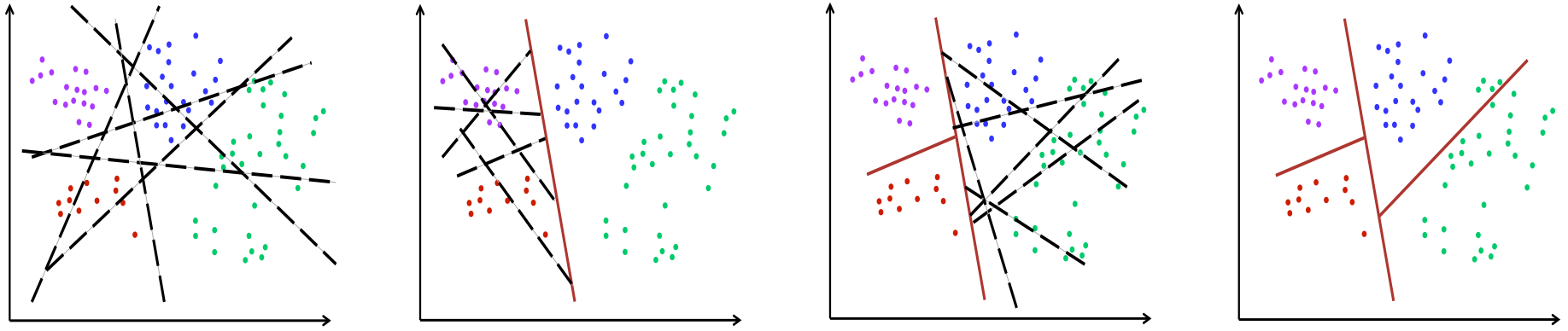
- Learning the split nodes
  - each decision should improve class separation as much as possible
  - measures for degree of separation: Gini impurity, information gain
  - finding the best decision: randomly try many splits
  - both training and prediction are very efficient

$$Gini = \sum_{k=1}^{K} \frac{N_k}{N}\left(1 - \frac{N_k}{N}\right)$$

$$IGain = -\sum_{k=1}^{K} \frac{N_k}{N} \log_2\left(\frac{N_k}{N} + \epsilon\right)$$

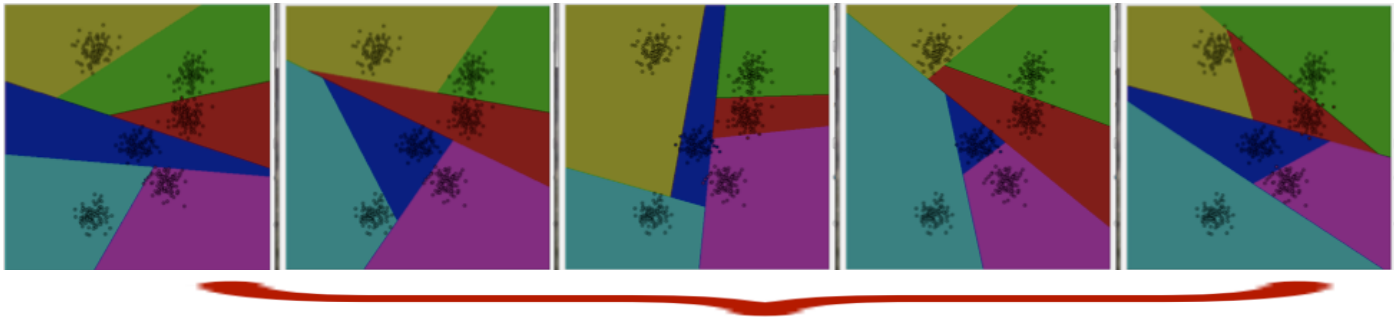
low for N_k/N=0 or N_k/N=1, high for N_k/N=0.5

→ favour splits such that most data of a class are in the same child node

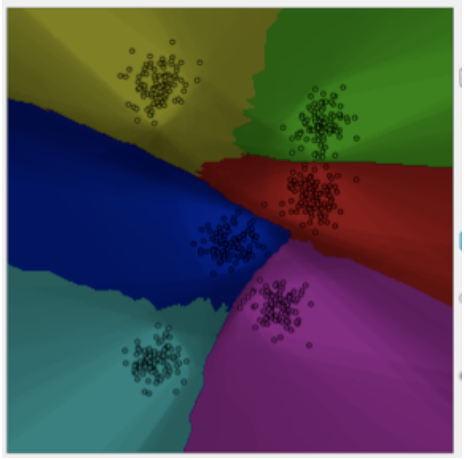


Decision forests

- Again, like regression forests
  - learn many different trees
  - to make them different, randomise candidate splits and/or training data
  - average class probabilities across all trees
  - forest will be (a lot) more robust than individual trees



$$Pr(y|\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} Pr_t(y|\mathbf{x})$$



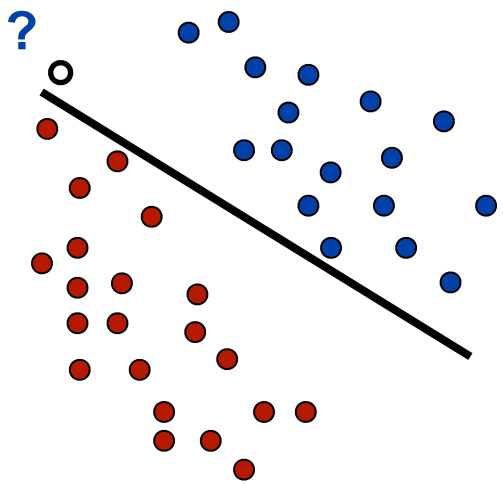
Decision forests

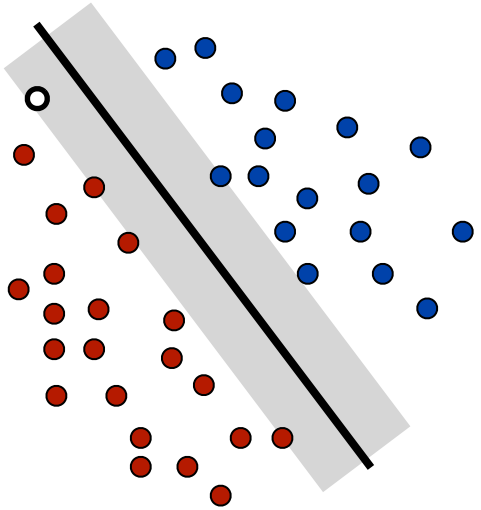
- Pros
  - computationally very efficient
  - flexible, can encode complicated relations without fitting problems
  - excellent prediction results in practice
- Cons
  - no parametric relation between inputs and outputs
  - unsuitable if very little training data is available

- ...not surprisingly, same as for regression forests

Maximum margin classifiers

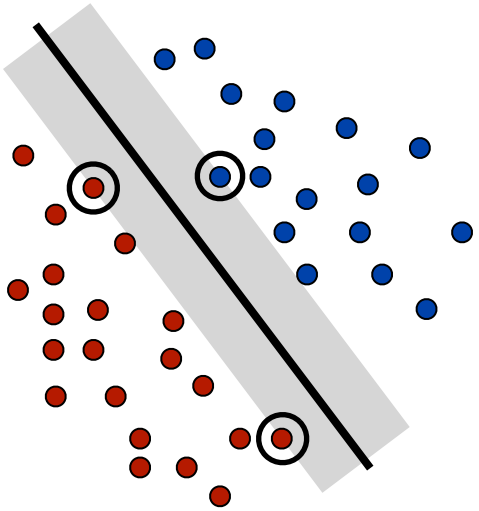
- Support vector machines (SVMs)
  - origin in computational learning theory
  - non-probabilistic (but output can be mapped to a pseudo-probability)
- Basic concept: large margin
  - training points should not only be separated, but should be far away from the decision boundary
  - if there is a large margin between the classes, the decision rule is expected to generalise better to new data





Support vector machines

- Maximising the margin
  - linear classifier $y_i = \boldsymbol{\theta}^\top \mathbf{x}_i + \theta_0$
  - let the two class labels be {-1,+1}
  - if all points are classified correctly $y_i(\boldsymbol{\theta}^\top \mathbf{x}_i + \theta_0) - 1 \geq 0$
  - points on the margin lie on hyperplanes $\boldsymbol{\theta}^\top \mathbf{x}_i + \theta_0 = \pm 1$
  - thus the margin is $2/\|\boldsymbol{\theta}\|$
  - to find the hyperplane with the largest margin, minimise $\|\boldsymbol{\theta}\|^2$ under the constraints $y_i(\boldsymbol{\theta}^\top \mathbf{x}_i + \theta_0) - 1 \geq 0$
  - a convex problem (quadratic program) with a unique solution



- Only the points on the margin (support vectors) define the decision boundary
  - prediction $y_{new} = \boldsymbol{\theta}^\top \mathbf{x}_{new} + \theta_0 = \sum_{i=1}^{T} a_i y_i \mathbf{x}_i^\top \mathbf{x}_{new} + \theta_0$

(for those who want to know)

- Derivation of SVM margin

equation of (unknown) separating hyperplane

$$\boldsymbol{\theta}^\top \mathbf{x} + \theta_0 = 0$$

bounding planes of the margin (through the nearest points on either side of the hyperplane)

$$\boldsymbol{\theta}^\top \mathbf{x} + \theta_0 = 1$$

$$\boldsymbol{\theta}^\top \mathbf{x} + \theta_0 = -1$$

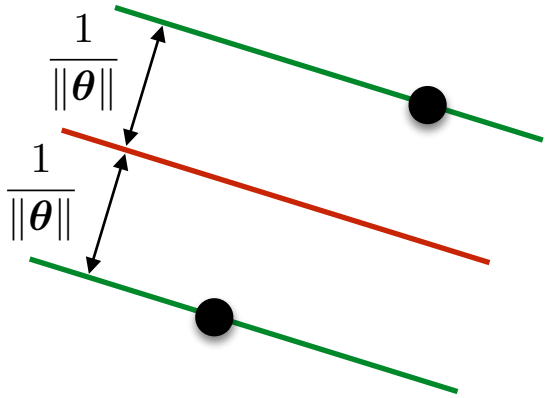
(instead of 1 we could use any non-zero constant, because plane parameters can be rescaled)

$$\boldsymbol{\theta}^\top \mathbf{x} + \theta_0 = 0 \Leftrightarrow q \cdot \boldsymbol{\theta}^\top \mathbf{x} + q \cdot \theta_0 = 0$$

distance from margin to separating plane (scale normal vector to unit length to get Hessian normal form)

$$\frac{1}{\|\boldsymbol{\theta}\|}\boldsymbol{\theta}^\top \mathbf{x} + \frac{1}{\|\boldsymbol{\theta}\|}\theta_0 = \frac{1}{\|\boldsymbol{\theta}\|}$$

total width of margin is twice that distance



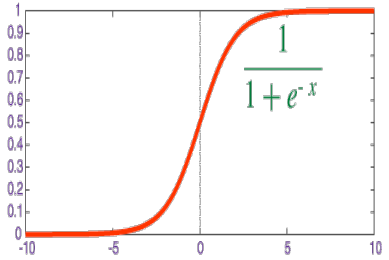
Support vector machines

- Extension to non-separable data
  - allow mistakes, but penalise them, $y_i(\boldsymbol{\theta}^\top \mathbf{x}_i + \theta_0) - 1 \geq -\xi_i \ , \ \xi_i \geq 0$
  - points with $0 < \xi_i < 1$ lie inside the margin
  - points with $\xi_i > 1$ are miss-classified, so at most $\sum \xi_i$ mistakes
  - hence, instead of $\|\boldsymbol{\theta}\|^2$ minimise $C \sum \xi_i + \|\boldsymbol{\theta}\|^2$
  - still a convex problem with a unique solution
- Non-linear decision boundaries
  - the usual lifting trick, replace $\mathbf{x}_i$ by $\mathbf{z}(\mathbf{x}_i)$
  - using the "kernel function" $k(\mathbf{x}_i,\mathbf{x}_j)=(\mathbf{z}(\mathbf{x}_i)^\top \mathbf{z}(\mathbf{x}_j))$ avoids explicit lifting to high-dimensional spaces, both for learning and prediction
  - placing basis functions only at the support vectors, thus suitable for high-dimensional spaces with rather little training data

$$y_{new} = \sum_{i=1}^{T} a_i y_i k(\mathbf{x}_i, \mathbf{x}_{new}) + \theta_0$$

Support vector machines

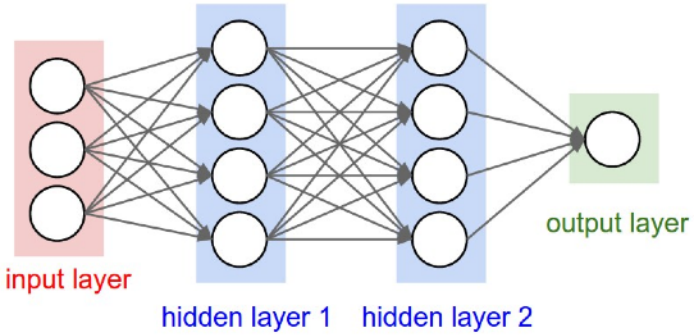
- Pros
  - convex problem, mathematically well-behaved
  - excellent performance in practice
- Cons
  - does not deliver class probabilities (one can get "pseudo-probabilities" by squashing the distance to the decision boundary through a sigmoid)
  - multi-class extension not straight-forward



- Note
  - there is a related probabilistic method, relevance vector machines, which returns proper probabilities
  - learning RVMs is not convex, only a local optimum for the parameters can be found
  - RVMs usually lead to even fewer relevance vectors
  - still, they are a lot less popular in practice

Other methods

- Many more classification methods exist, e.g.
  - variants of boosting
  - neural networks → later in this course

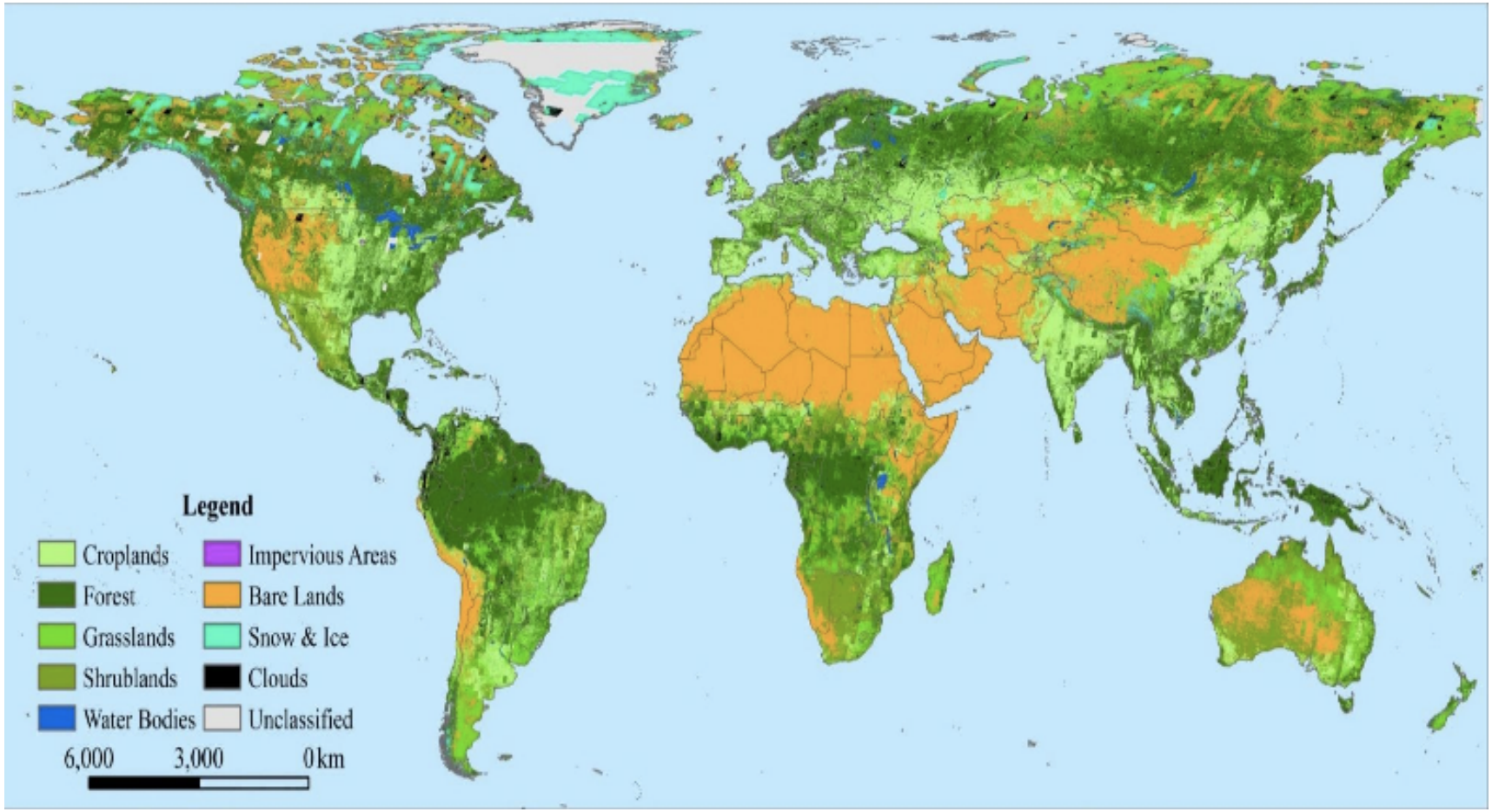


- If you come across a method you do not know, look it up in a textbook
  - All qualified methods are ok if used properly
  - There is no inherently superior method
  - but: you should know what your method is doing
  - and: for a specific problem some methods might be a lot better than others

Application example

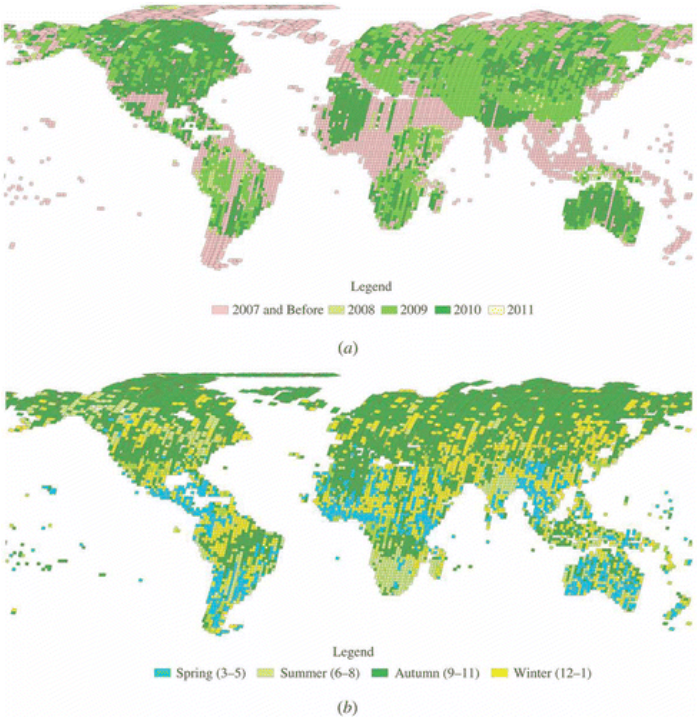
Classification example

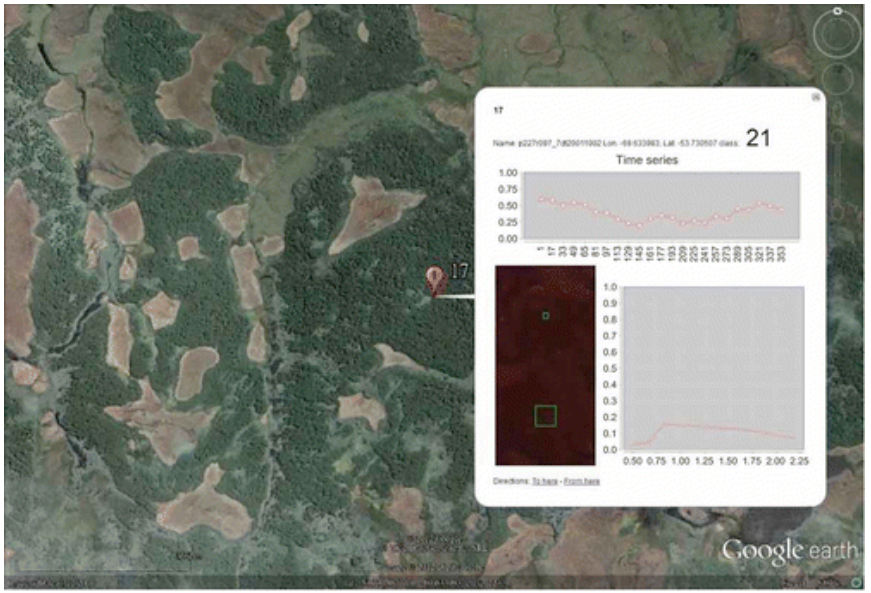
- FROM global land cover
  - [Gong et al., Int. J. Remote Sensing 34, 2013]



Classification example

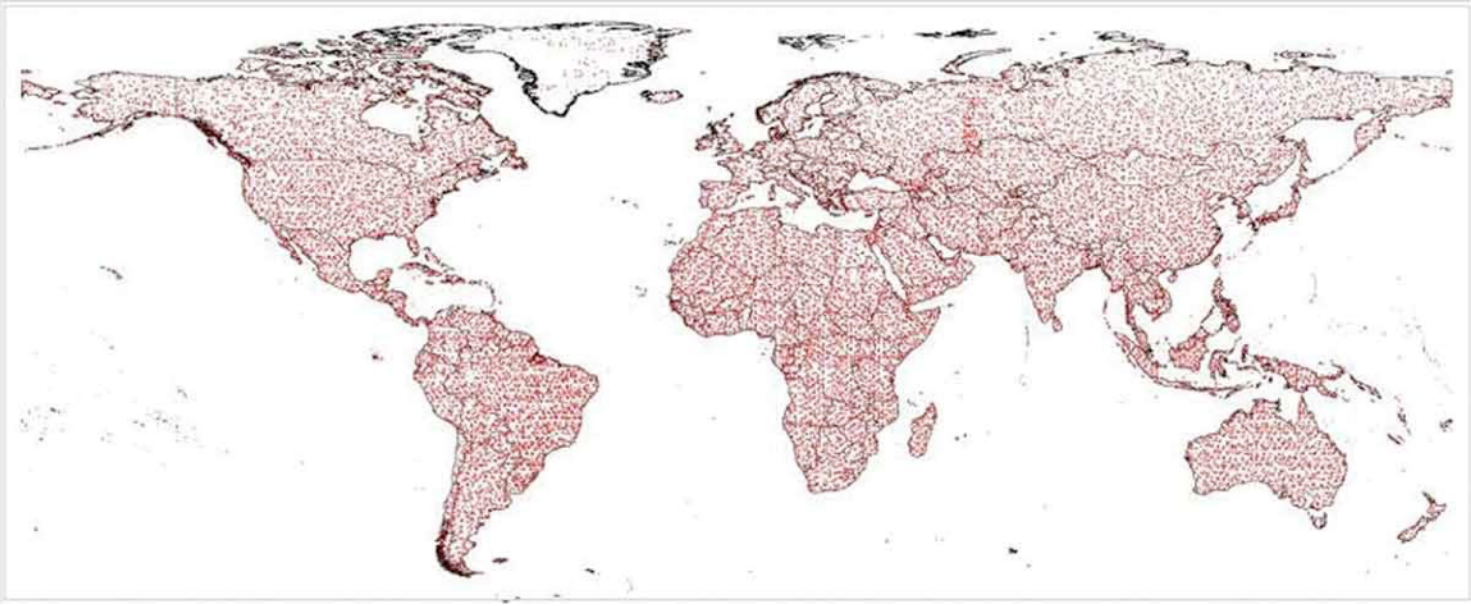
- Input
  - LandSat TM / ETM+ imagery with 30 m resolution, 6 bands 450-2350 nm, corrected for sun & viewing angle, atmospheric effects, clouds
  - 10-20 training samples ≥ 8 x 8 pixels per scene (~91000 total), found with interactive photo-interpretation





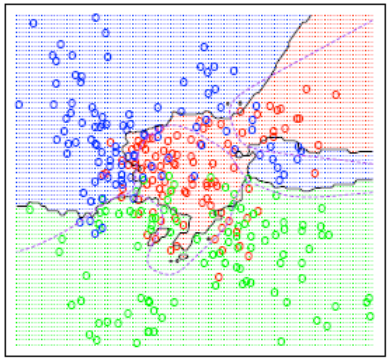
Classification example

- Method
  - Classify each scene separately, using training data also from 30 (spatially and temporally) neighbouring scenes in same eco-region
  - Support Vector Machine with Gaussian radial basis function kernel
- Output
  - world-wide land cover map at 30 m resolution
  - prediction accuracy: 71.5% correct predictions (measured at ~8600 homogeneous 500 x 500 m validation sites)

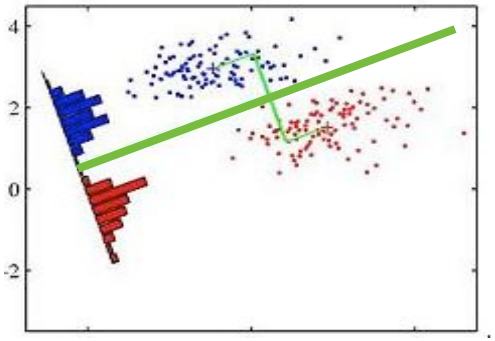


Summary

- Classification
  - predict discrete quantities from observed (image) data
  - labels can be categorical or ordered



- Basic methods
  - *k*-nearest neighbours
  - linear / quadratic discriminant
  - logistic regression



- Further methods
  - classification trees, random forests
  - support vector machines
  - ...