Computer Vision

Pattern Recognition Concepts – Part II

Luis F. Teixeira
MAP-i 2012/13

Last lecture

- The Bayes classifier yields the **optimal** decision rule if the prior and class-conditional distributions are **known**.
- This is unlikely for most applications, so we can:
  - attempt to estimate $p(x|\omega_i)$ from data, by means of density estimation techniques
    - Naïve Bayes and nearest-neighbors classifiers
  - assume $p(x|\omega_i)$ follows a particular distribution (i.e. Normal) and estimate its parameters
    - quadratic classifiers
  - ignore the underlying distribution, and attempt to separate the data geometrically
    - discriminative classifiers

k-Nearest neighbour classifier

- Given the training data $D = \{x_1,\ldots,x_n\}$ as a set of n labeled examples, the **nearest neighbour classifier** assigns a test point x the label associated with its closest neighbour (or k neighbours) in D.

- Closeness is defined using a **distance function**.

Distance functions

- A general class of metrics for *d*-dimensional patterns is the **Minkowski metric**, also known as the L_p norm

$$L_p(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{d} \left| x_i - y_i \right|^p \right)^{1/p}$$

- The **Euclidean distance** is the L_2 norm

$$L_2(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{d} \left| x_i - y_i \right|^2 \right)^{1/2}$$

- The **Manhattan** or **city block distance** is the L1 norm

$$L_1(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{d} \left| x_i - y_i \right|$$

Distance functions

- The **Mahalanobis distance** is based on the covariance of each feature with the class examples.

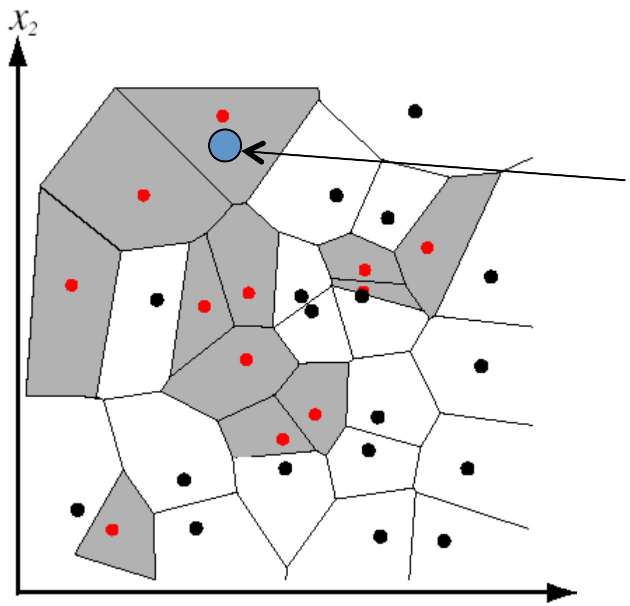
$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)}$$

  - Based on the assumption that distances in the direction of high variance are less important
  - Highly dependent on a good estimate of covariance

1-Nearest neighbour classifier

Assign label of nearest training data point to each test data point

Black = negative
Red = positive



from Duda *et al.*

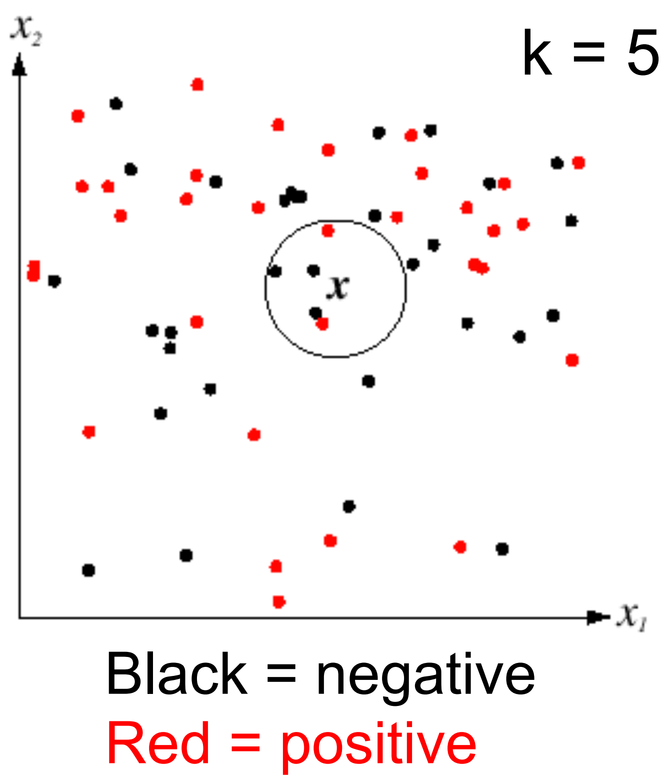
Novel test example

Closest to a positive example from the training set, so classify it as positive.

Voronoi partitioning of feature space
for 2-category 2D data

k-Nearest neighbour classifier

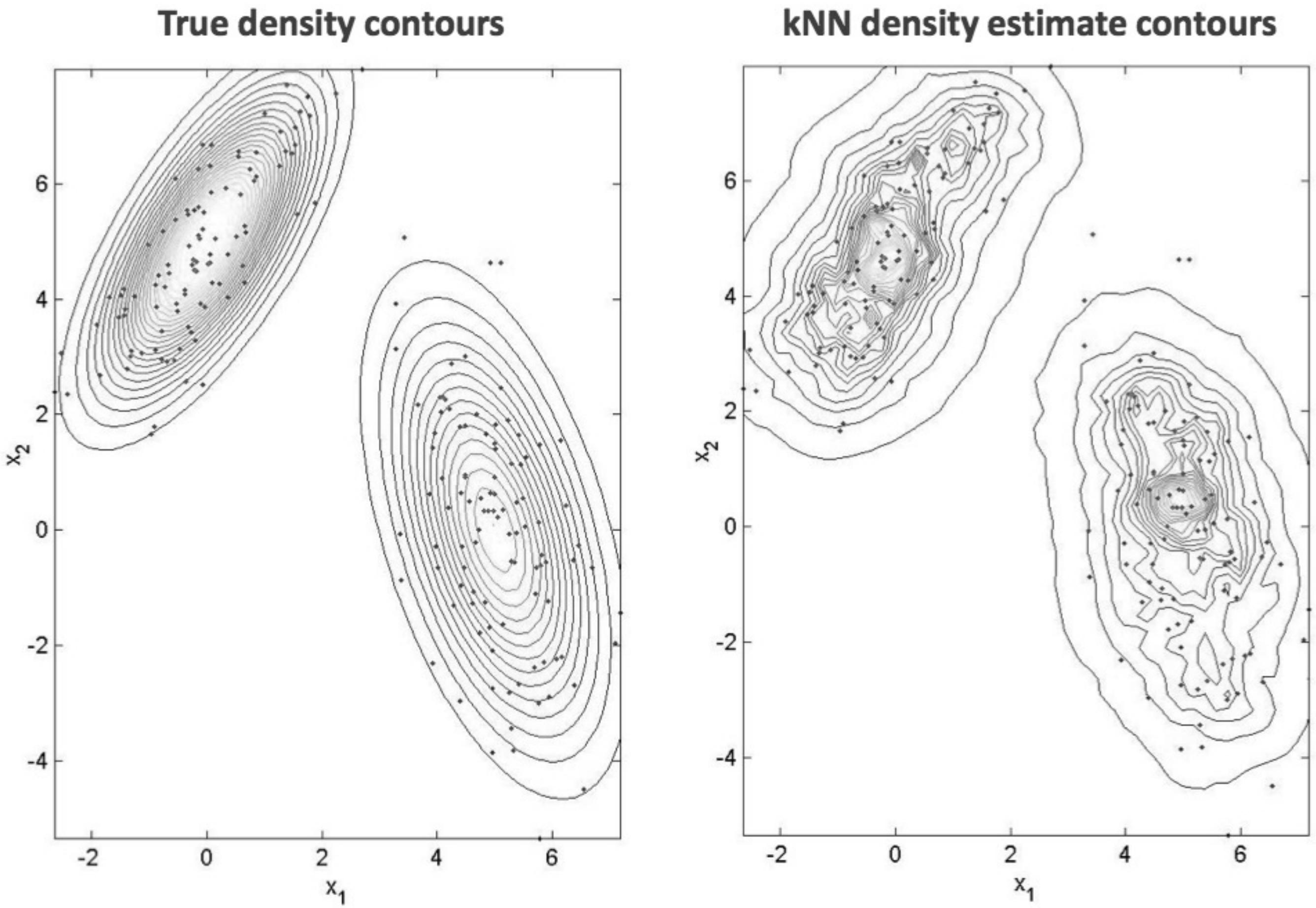
- For a new point, find the k closest points from training data
- Labels of the k points "vote" to classify



If the query lands here, the 5 NN consist of 3 negatives and 2 positives, so we classify it as negative.

k-Nearest neighbour classifier

- The main advantage of kNN is that it leads to a very simple approximation of the (optimal) Bayes classifier



kNN as a classifier

- **Advantages**:
  - Simple to implement
  - Flexible to feature / distance choices
  - Naturally handles multi-class cases
  - Can do well in practice with enough representative data
- **Disadvantages:**
  - Large search problem to find nearest neighbors → Highly susceptible to the **curse of dimensionality**
  - Storage of data
  - Must have a meaningful distance function

Dimensionality reduction

- **The curse of dimensionality**
  - The number of examples needed to accurately train a classifier **grows exponentially** with the dimensionality of the model
  - In theory, information provided by additional features should help improve the model's accuracy
  - In reality, however, additional features increase the **risk of overfitting**, i.e., memorizing noise in the data rather than its underlying structure
  - For a given sample size, there is a maximum number of features above which the classifier's performance **degrades** rather than improves

Dimensionality reduction

- The curse of dimensionality can be limited by:
  - incorporating prior knowledge (e.g., parametric models)
  - enforcing smoothness in the target function (e.g., regularization)
  - reducing the dimensionality
    - creating a subset of new features by combinations of the existing features – **feature extraction**
    - choosing a subset of all the features – **feature selection**

Dimensionality reduction

- In feature extraction methods, two types of criteria are commonly used:
  - Signal representation: The goal of feature selection is to accurately represent the samples in a lower-dimensional space (e.g. **Principal Components Analysis**, or PCA)
  - Classification: The goal of feature selection is to enhance the class-discriminatory information in the lower-dimensional space (e.g. Fisher's Li**near Discriminants Analysis**, or LDA)

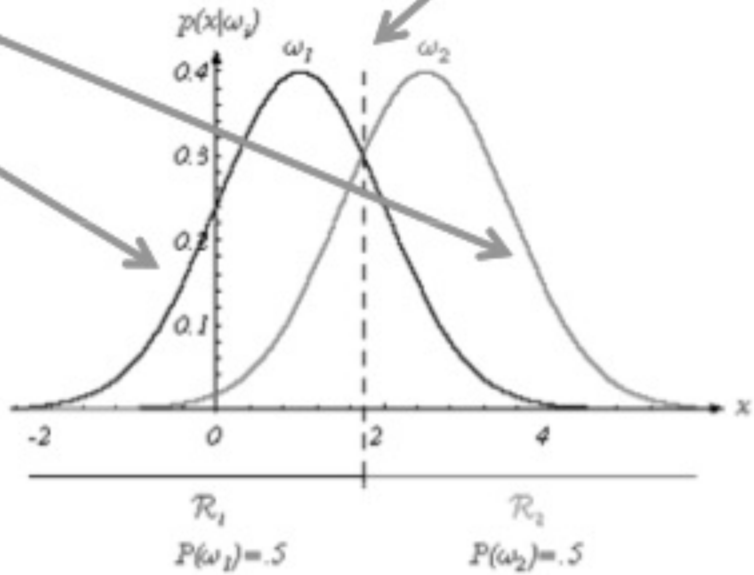
Discriminative classifiers

- Decision boundary-based classifiers:
  - Decision trees
  - Neural networks
  - Support vector machines

Discriminative vs Generative

Discriminative models only define a decision boundary

Generative models estimate the distributions



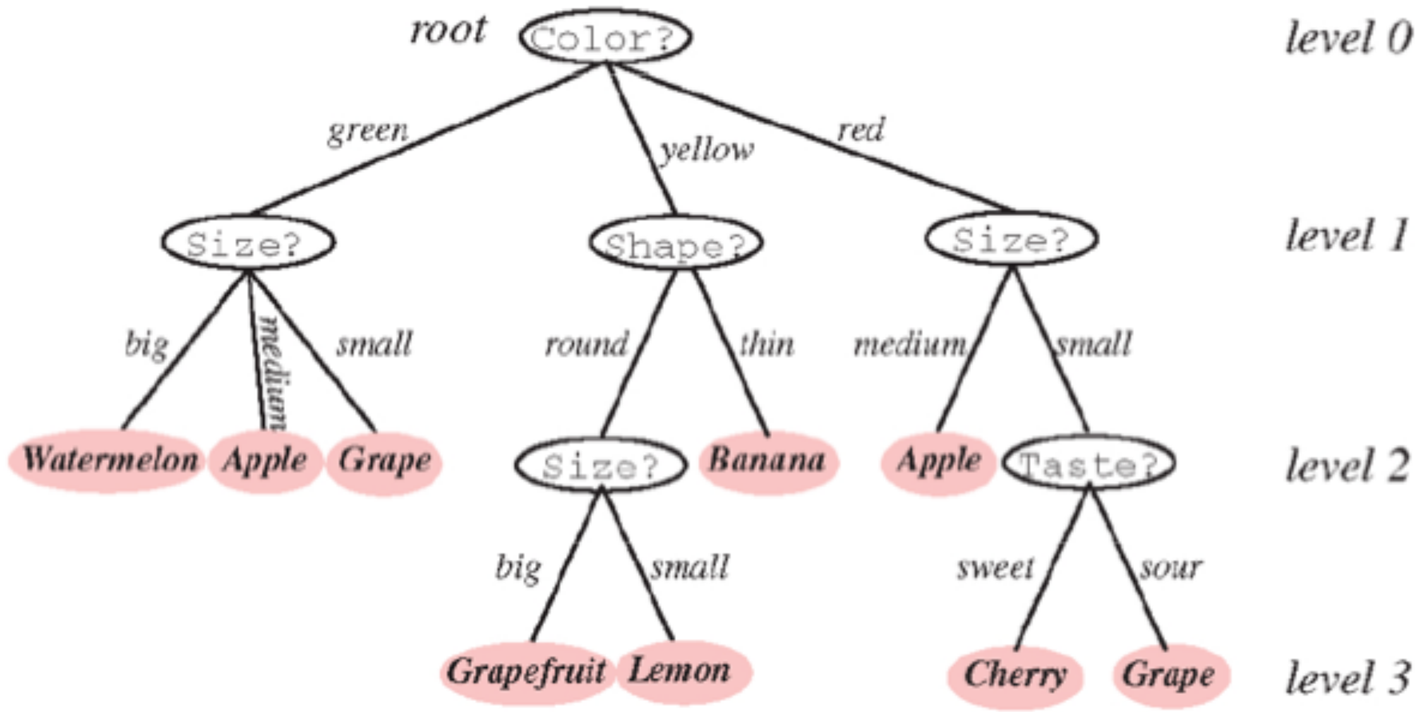
Discriminative vs Generative

- Discriminative models differ from generative models in that they do not allow one to **generate samples** from the joint distribution of x and y.
- However, for tasks such as **classification** and **regression** that do not require the joint distribution, discriminative models generally yield superior performance.
- On the other hand, generative models are typically **more flexible** than discriminative models in expressing dependencies in complex learning tasks.

Decision trees

- Decision trees are **hierarchical** decision systems in which conditions are sequentially tested until a class is accepted
- The feature space is **split** into unique regions corresponding to the classes, in a **sequential** manner
- The searching of the region to which the feature vector will be assigned to is achieved via a **sequence of decisions** along a path of nodes

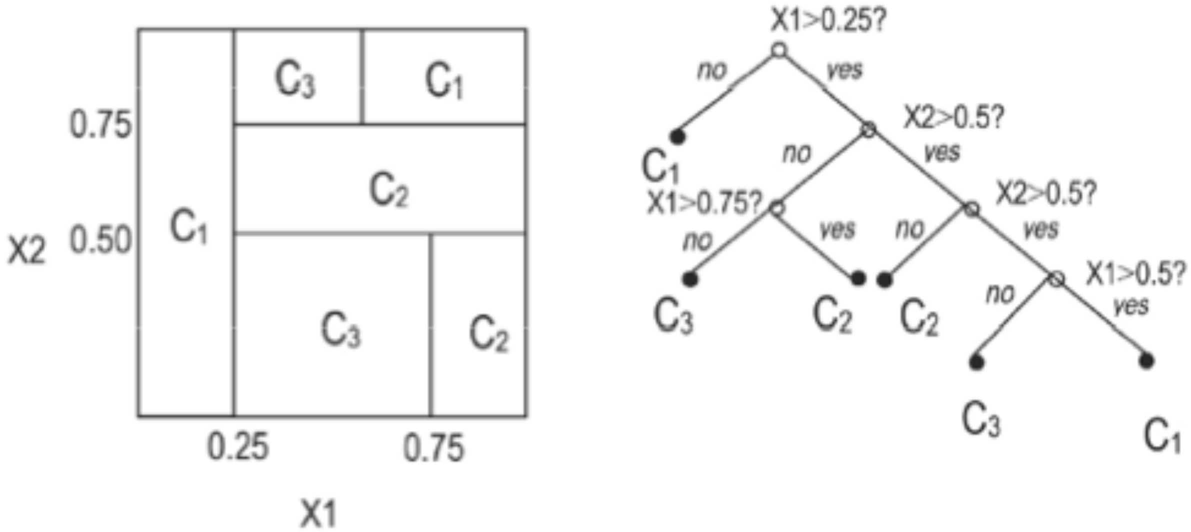
Decision trees



Decision trees classify a pattern through a **sequence** of questions, in which the next question depends on the answer to the current question

Decision trees

- The most popular schemes among decision trees are those that split the space into hyper-rectangles with sides parallel to the axes
- The sequence of decisions is applied to individual features, in the form of "is the feature $x_k < \alpha$?"



Artificial neural networks

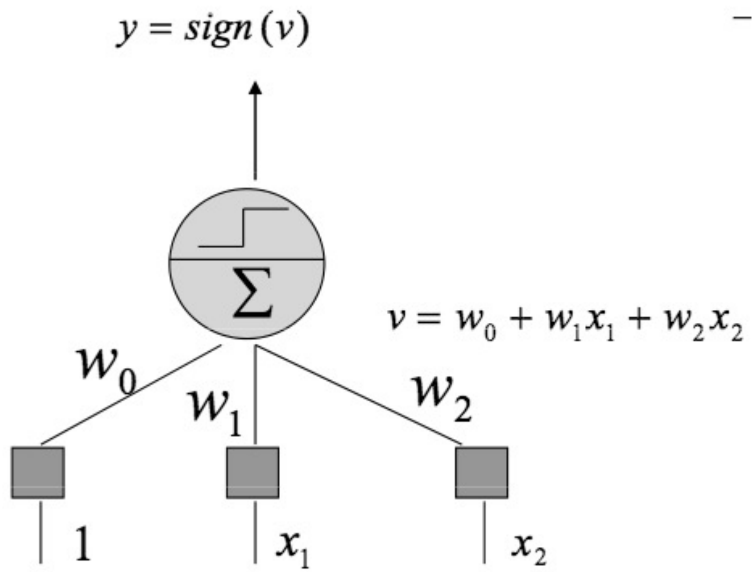
- A neural network is a set of connected input/output units where each connection has a **weight** associated with it
- During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class output of the input signals

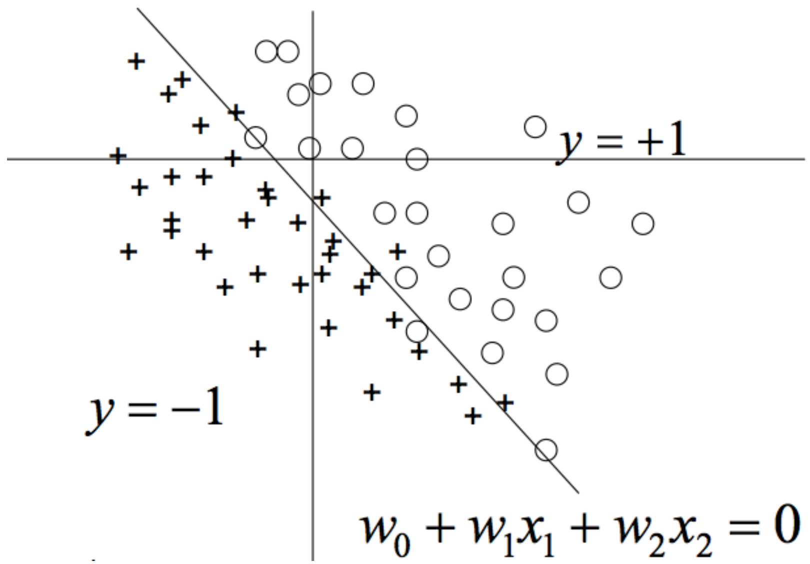
Artificial neural networks

- Examples of ANN:
  - Perceptron
  - Multilayer Perceptron (MLP)
  - Radial Basis Function (RBF)
  - Self-Organizing Map (SOM, or Kohonen map)
- Topologies:
  - Feed forward
  - Recurrent

Perceptron

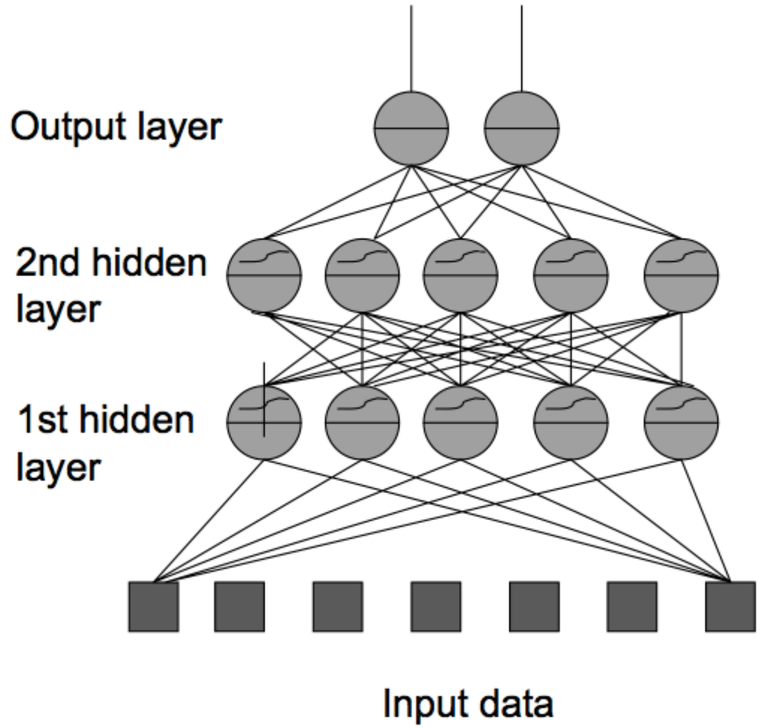
- Defines a (hyper)plane that linearly separates the feature space
- The inputs are real values and the output +1,-1
- Activation functions: step, linear, logistic sigmoid, Gaussian



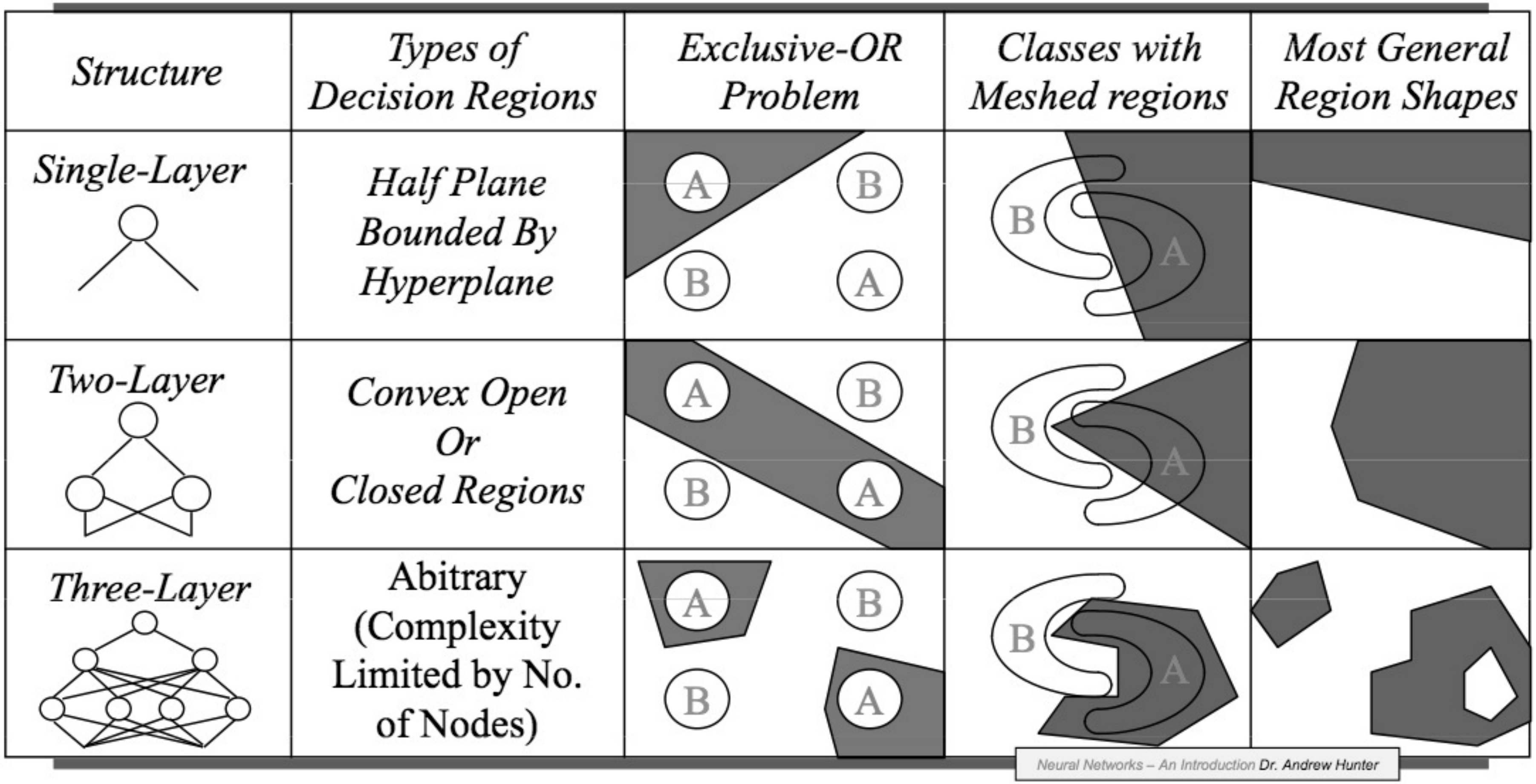


Multilayer perceptron

- To handle more **complex problems** (than linearly separable ones) we need multiple layers.
- Each layer receives its inputs from the previous layer and **forwards** its outputs to the next layer
- The result is the **combination of linear boundaries** which allow the separation of complex data
- Weights are obtained through the **back propagation algorithm**

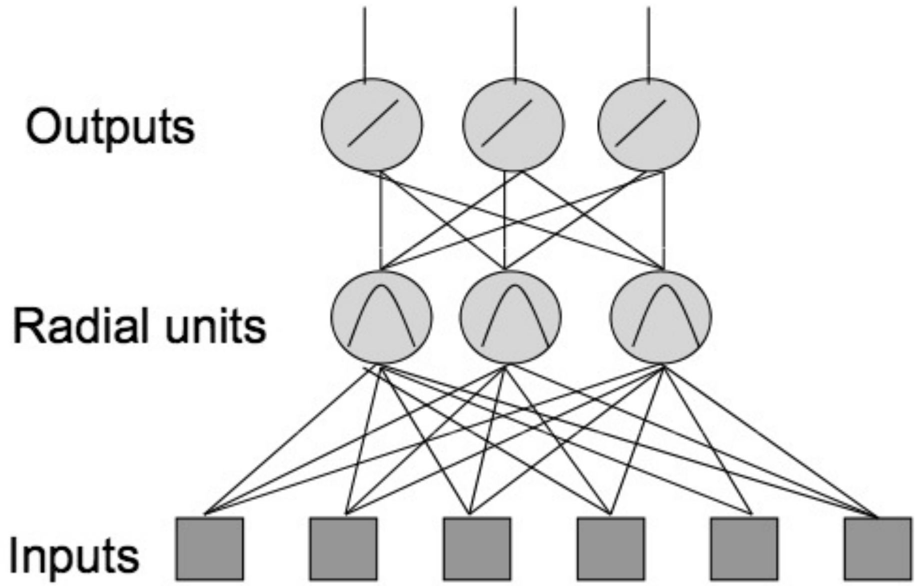


Non-linearly separable problems



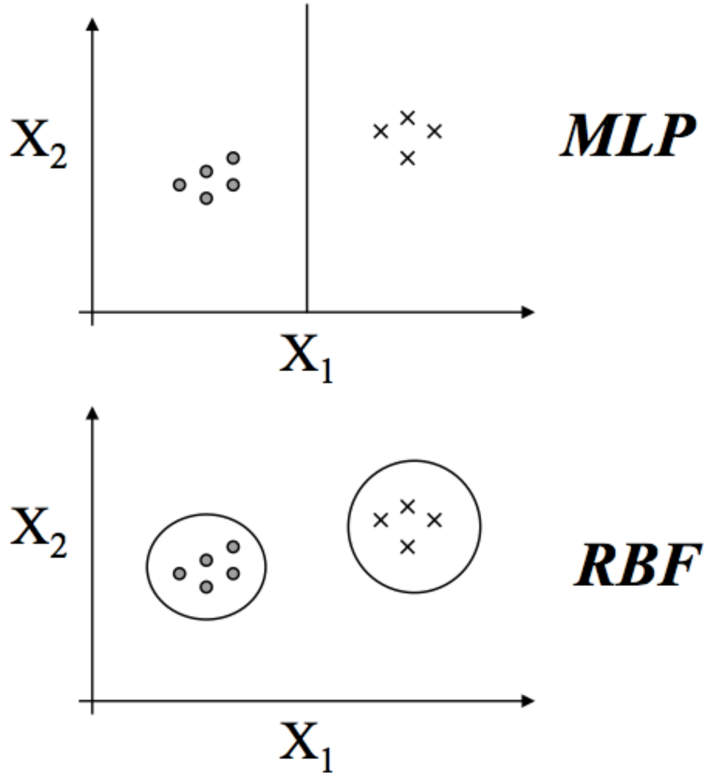
RBF networks

- RBF networks approximate functions using (radial) basis functions as the building blocks. Generally, the hidden unit function is Gaussian and the output Layer is linear



MLP vs RBF

- **Classification**
  - MLPs separate classes via hyperplanes
  - RBFs separate classes via hyperspheres
- **Learning**
  - MLPs use distributed learning
  - RBFs use localized learning
  - RBFs train faster
- **Structure**
  - MLPs have one or more hidden layers
  - RBFs have only one layer
  - RBFs require more hidden neurons => curse of dimensionality



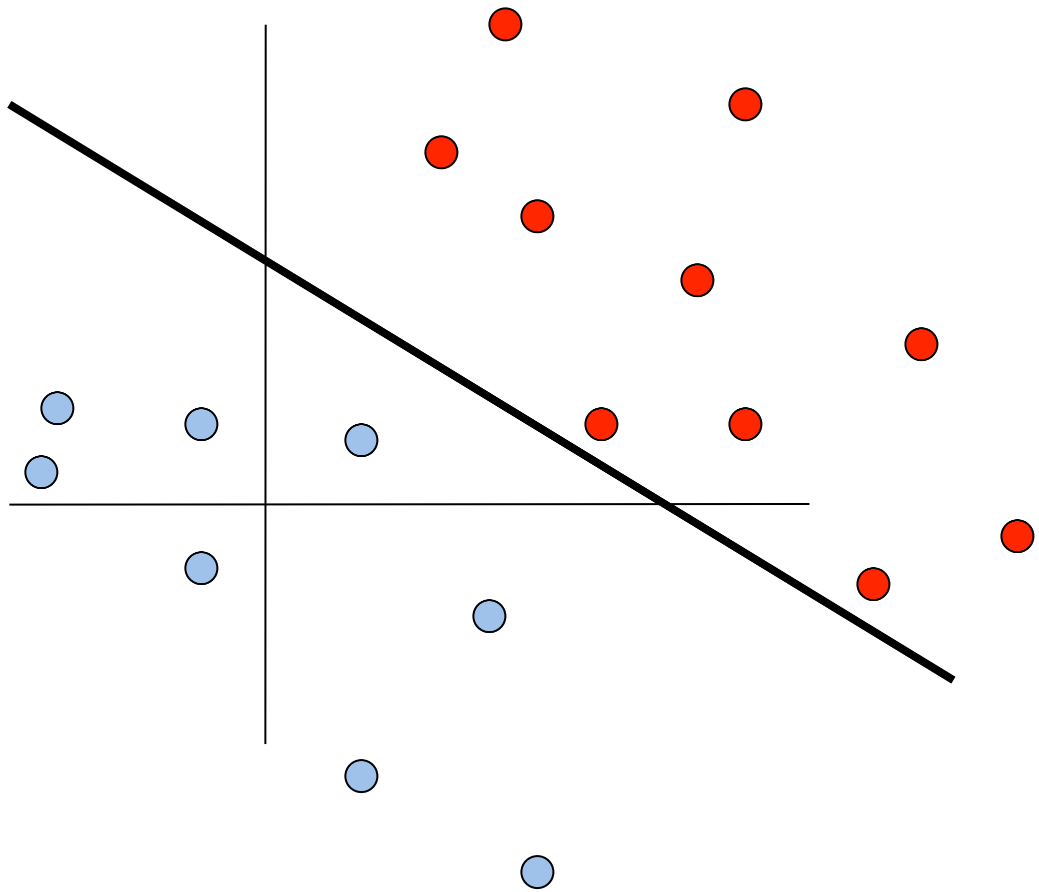
ANN as a classifier

- Advantages
  - High tolerance to noisy data
  - Ability to classify untrained patterns
  - Well-suited for continuous-valued inputs and outputs
  - Successful on a wide array of real-world data
  - Algorithms are inherently parallel
- Disadvantages
  - Long training time
  - Requires a number of parameters typically best determined empirically, e.g., the network topology or ``structure.”
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of ``hidden units" in the network

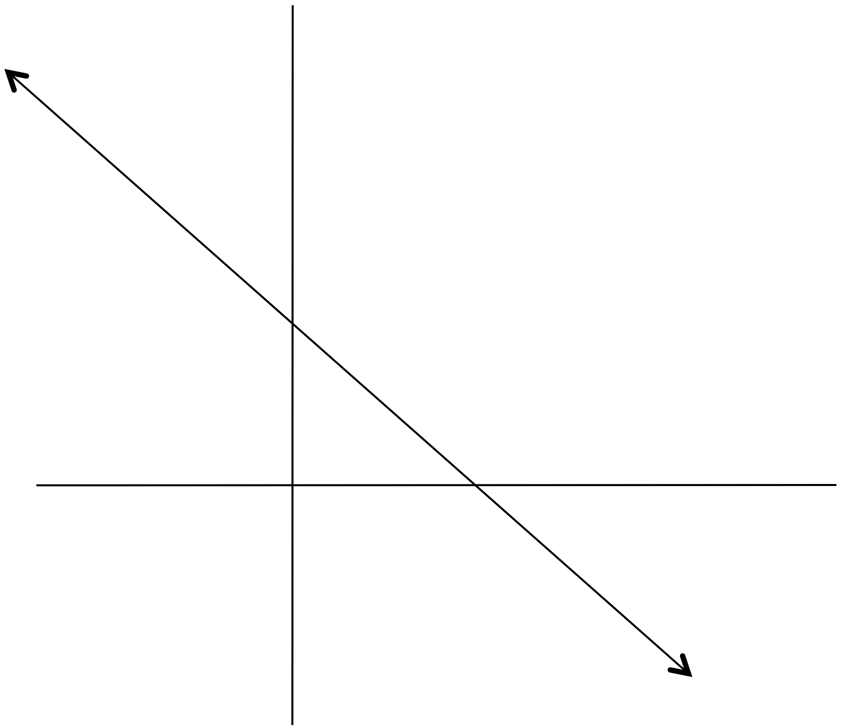
Support Vector Machine

- Discriminant function is a hyperplane (line in 2D) in feature space (similar to the Perceptron)
- In a nutshell:
  - Map the data to a predetermined very high-dimensional space via a kernel function
  - Find the hyperplane that **maximizes the margin** between the two classes
  - If data are not separable find the hyperplane that maximizes the margin and minimizes the (a weighted average of the) misclassifications

Linear classifiers



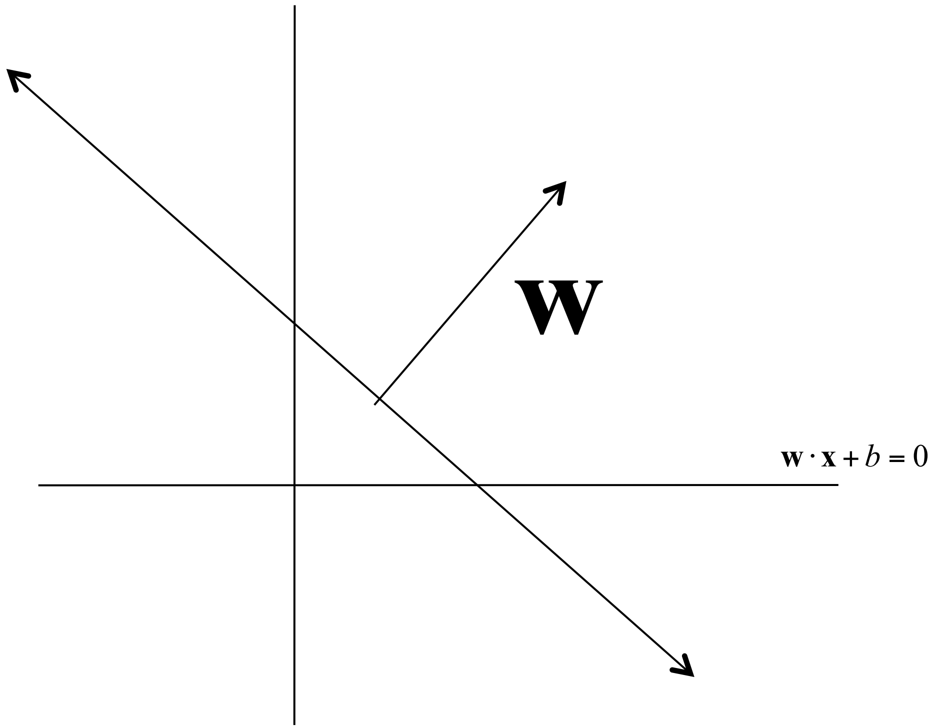
Linear functions in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

Linear functions in R^2



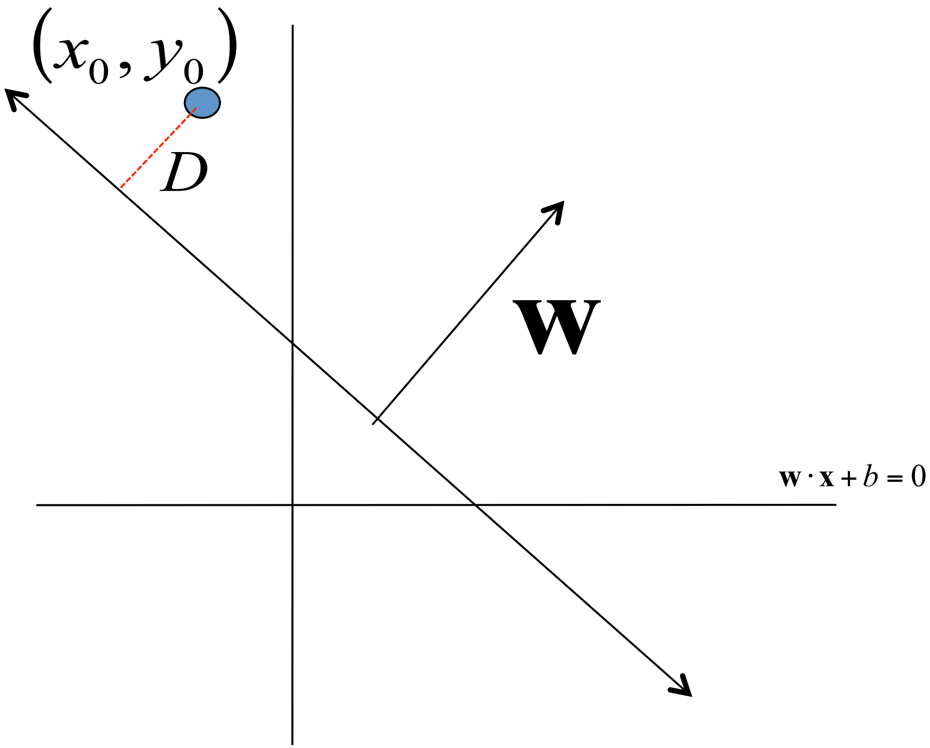
Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\Updownarrow$$

$$\mathbf{w}^T \mathbf{x} + b = 0$$

Linear functions in R^2



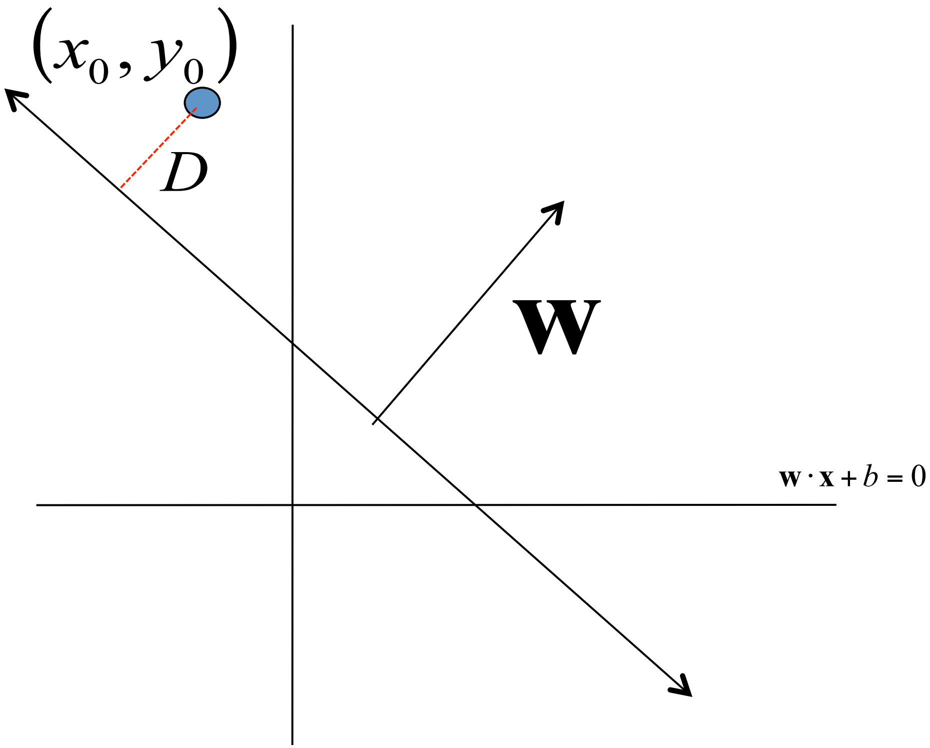
Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\Updownarrow$$

$$\mathbf{w}^T \mathbf{x} + b = 0$$

Linear functions in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

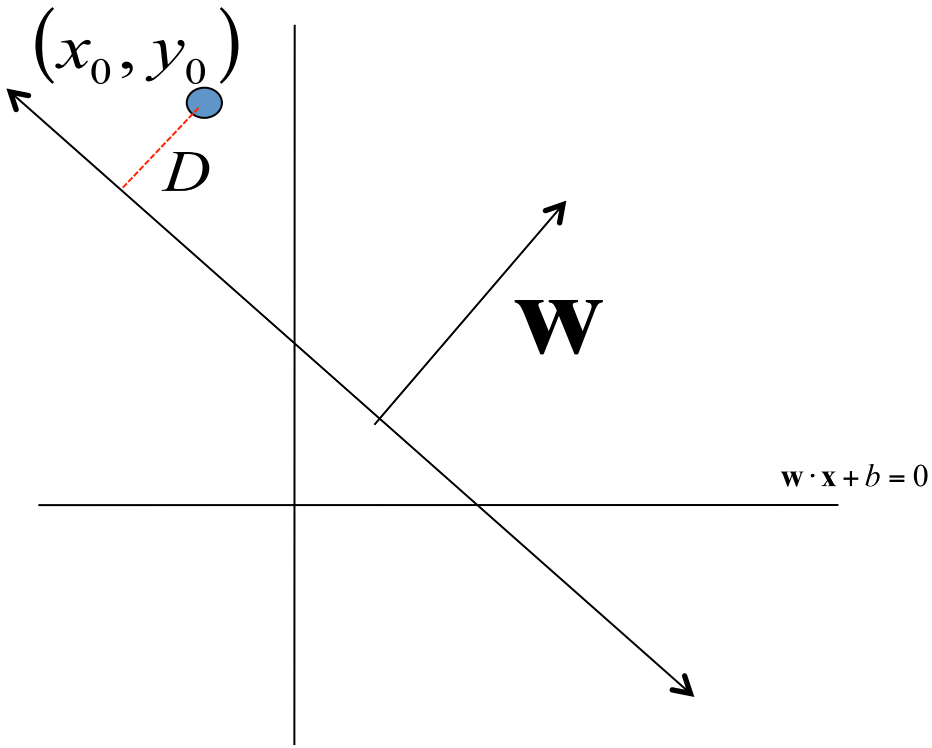
$$\Updownarrow$$

$$\mathbf{w}^T \mathbf{x} + b = 0$$

$$D = \frac{\left| a x_0 + c y_0 + b \right|}{\sqrt{a^2 + c^2}}$$

distance from point to line

Linear functions in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\Updownarrow$$

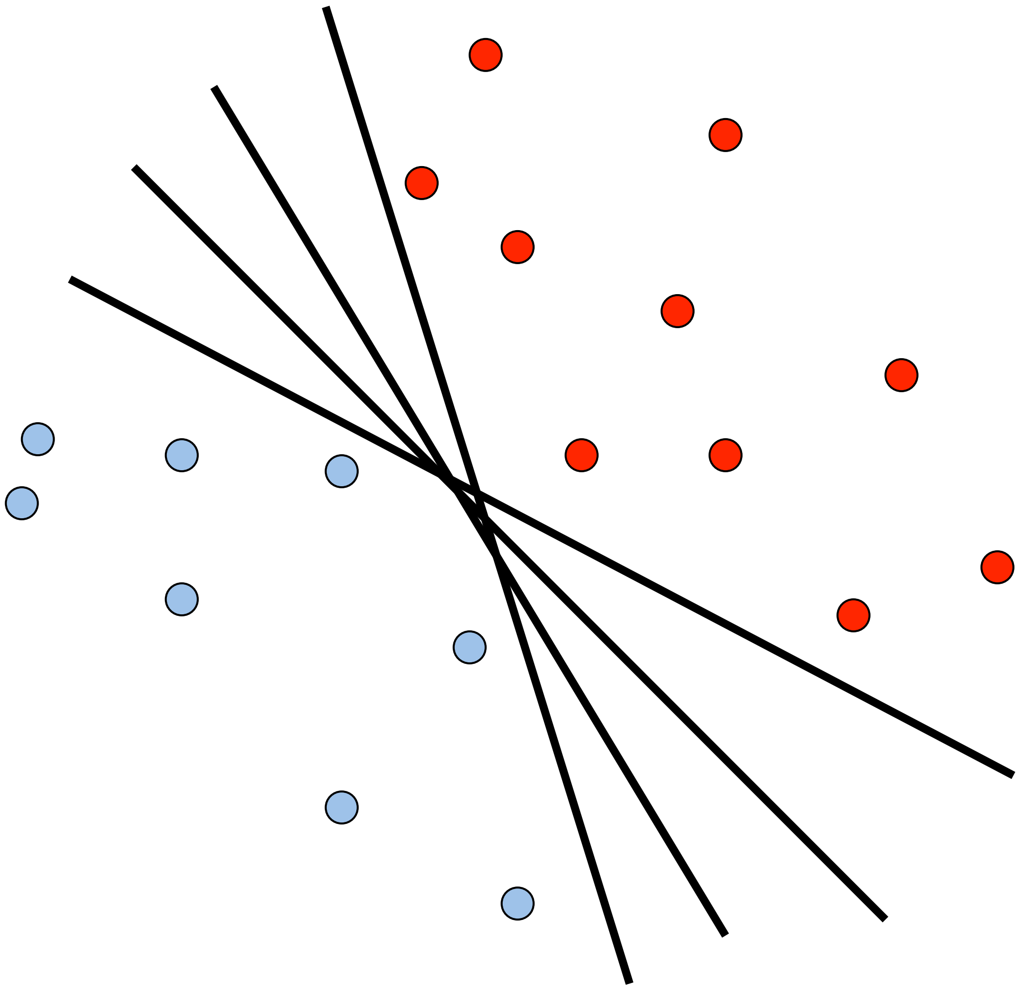
$$\mathbf{w}^T \mathbf{x} + b = 0$$

$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{x} + b}{\|\mathbf{w}\|}$$

distance from point to line

Linear classifiers

Find linear function to separate positive and negative examples

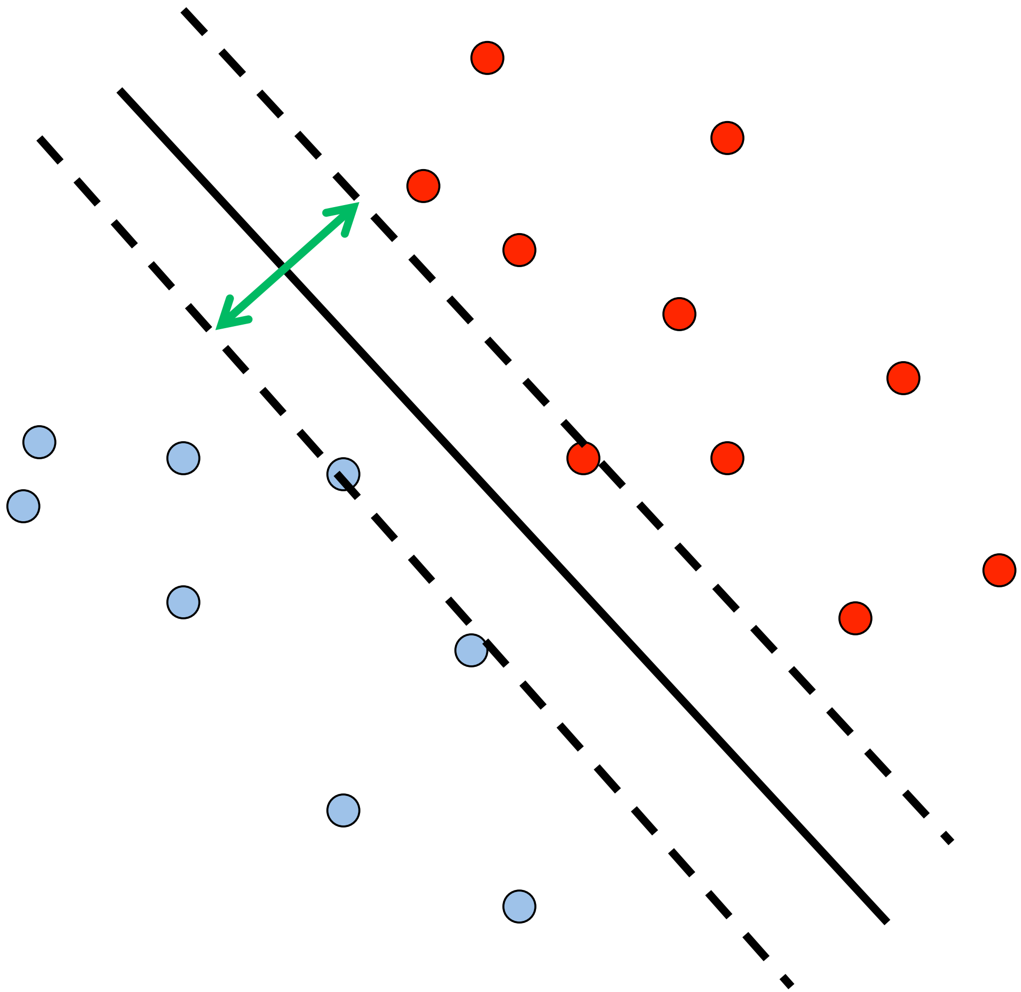


$$\mathbf{x}_i \text{ positive}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

$$\mathbf{x}_i \text{ negative}: \quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$$

Which line
is best?

Support Vector Machines

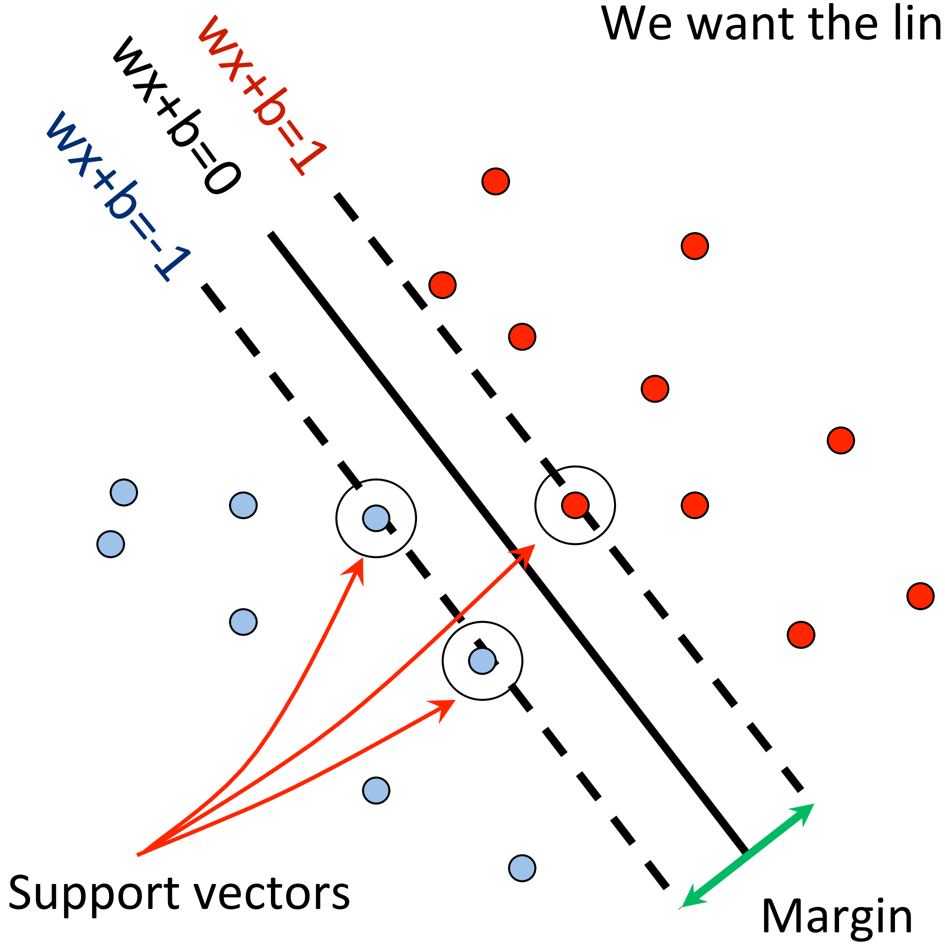


Discriminative classifier based on *optimal separating line (for 2D case)*

Maximize the ***margin*** between the positive and negative training examples

Support Vector Machines

We want the line that maximizes the margin.



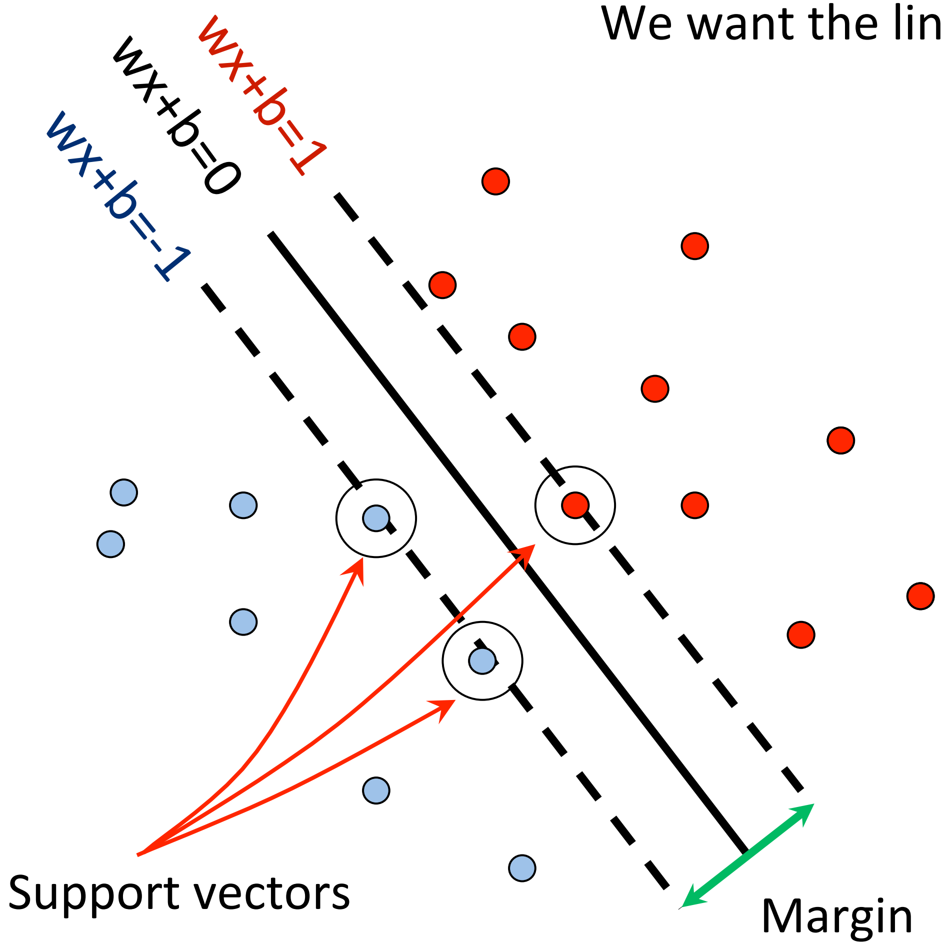
$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

For support, vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Support Vector Machines

We want the line that maximizes the margin.



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support, vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

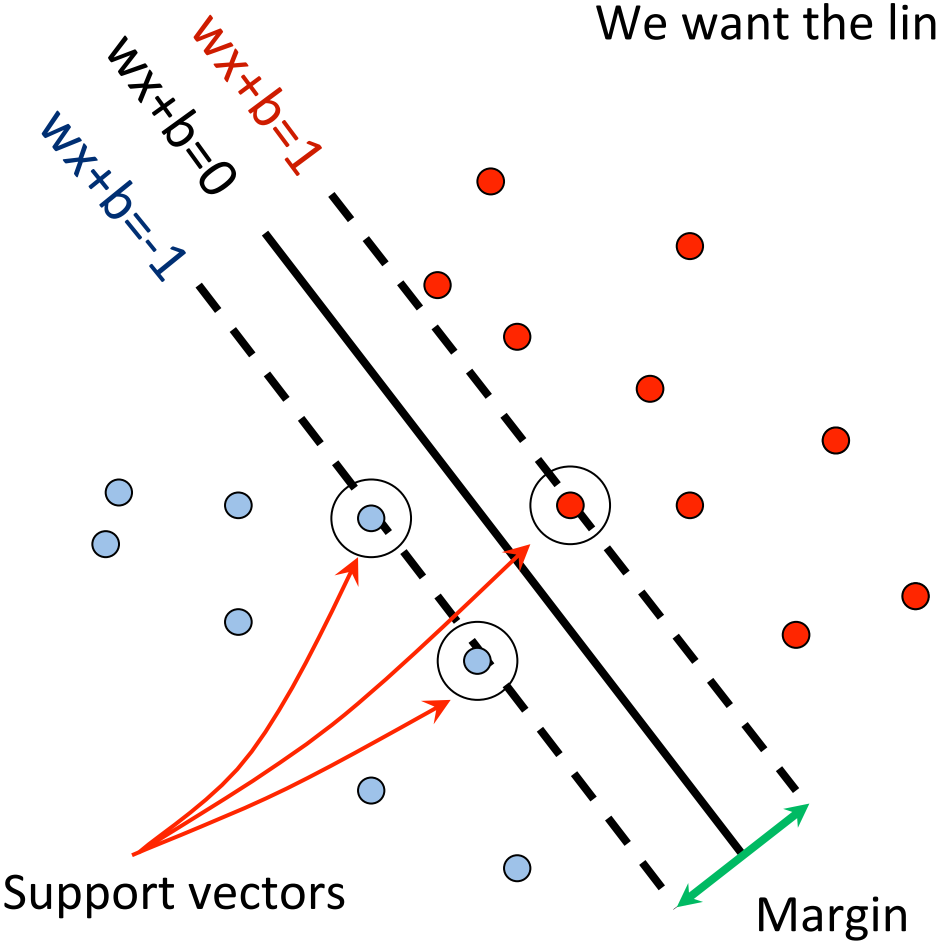
Distance between point and line: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

For support vectors:

$$\frac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|} = \frac{\pm 1}{\|\mathbf{w}\|} \qquad M = \left|\frac{1}{\|\mathbf{w}\|} - \frac{-1}{\|\mathbf{w}\|}\right| = \frac{2}{\|\mathbf{w}\|}$$

Support Vector Machines

We want the line that maximizes the margin.



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support, vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and line: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Therefore, the margin is $2 / ||w||$

Finding the maximum margin line

1. Maximize margin $2/||\mathbf{w}||$
2. Correctly classify all training data points:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

Quadratic optimization problem:

Minimize $\frac{1}{2}\mathbf{w}^T\mathbf{w}$

Subject to $y_i(\mathbf{w}\cdot\boldsymbol{x}_i+b) \geq 1$

Finding the maximum margin line

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

learned weight

support vector

Finding the maximum margin line

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

  $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)

  $$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

- Classification function:

$$f(x) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + \text{b})$$
$$= \text{sign}\left(\sum_{\text{i}} \alpha_{\text{i}} \mathbf{x}_i \cdot \mathbf{x} + b\right)$$

If f(x) < 0, classify as negative,
if f(x) > 0, classify as positive

Questions

- **What if the features are not 2D?**
- What if the data is not linearly separable?
- What if we have more than just two categories?

Questions

- What if the features are not 2D?
  - Generalizes to d-dimensions – replace line with "hyperplane"
- What if the data is not linearly separable?
- What if we have more than just two categories?

Questions

- What if the features are not 2d?
- **What if the data is not linearly separable?**
- What if we have more than just two categories?

Soft-margin SVMs

- Introduce **slack variable** and allow some instances to fall within the margin, but penalize them
- Constraint becomes: $y_i(w \cdot x_i + b) \geq 1 - \xi_i, \ \forall x_i$

$$\xi_i \geq 0$$

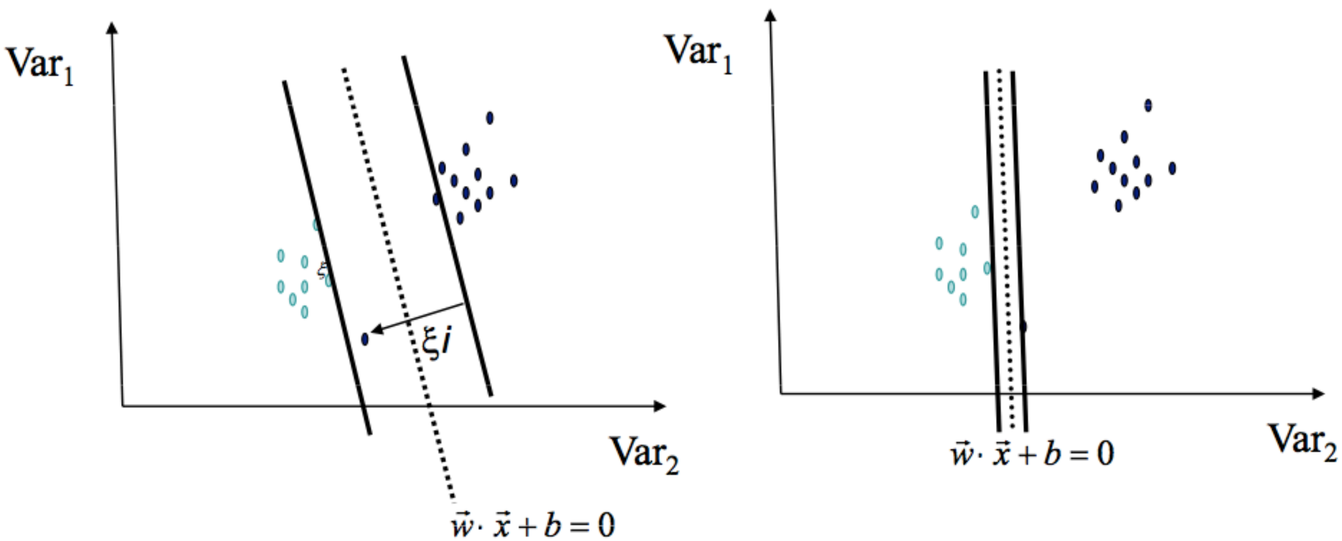
- Objective function penalizes for misclassified instances within the margin

$$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

- C trades-off margin width and classifications
- As $C \to \infty$, we get closer to the hard-margin solution

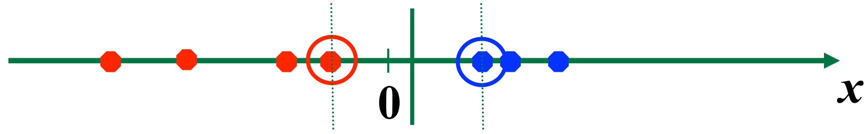
Soft-margin vs Hard-margin SVMs

- Soft-Margin always has a solution
- Soft-Margin is more robust to outliers
  - Smoother surfaces (in the non-linear case)
- Hard-Margin does not require to guess the cost parameter (requires no parameters at all)

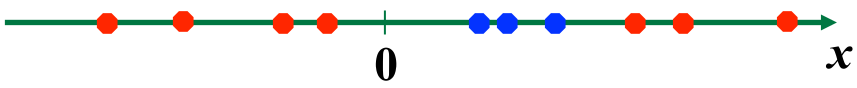


Non-linear SVMs

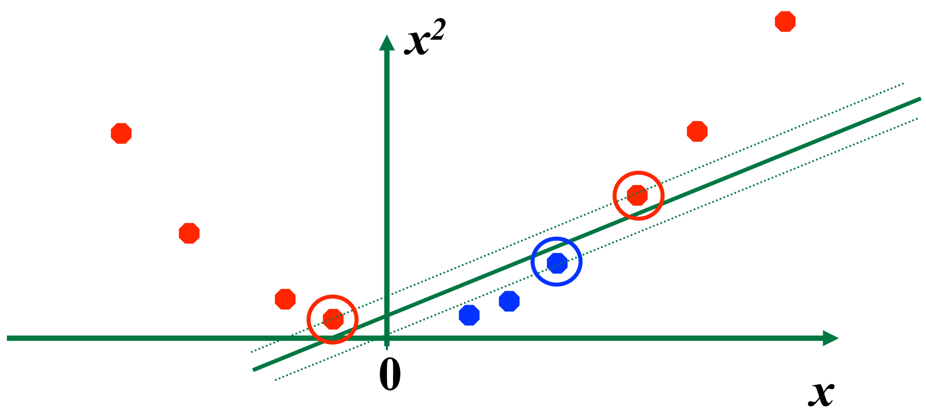
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

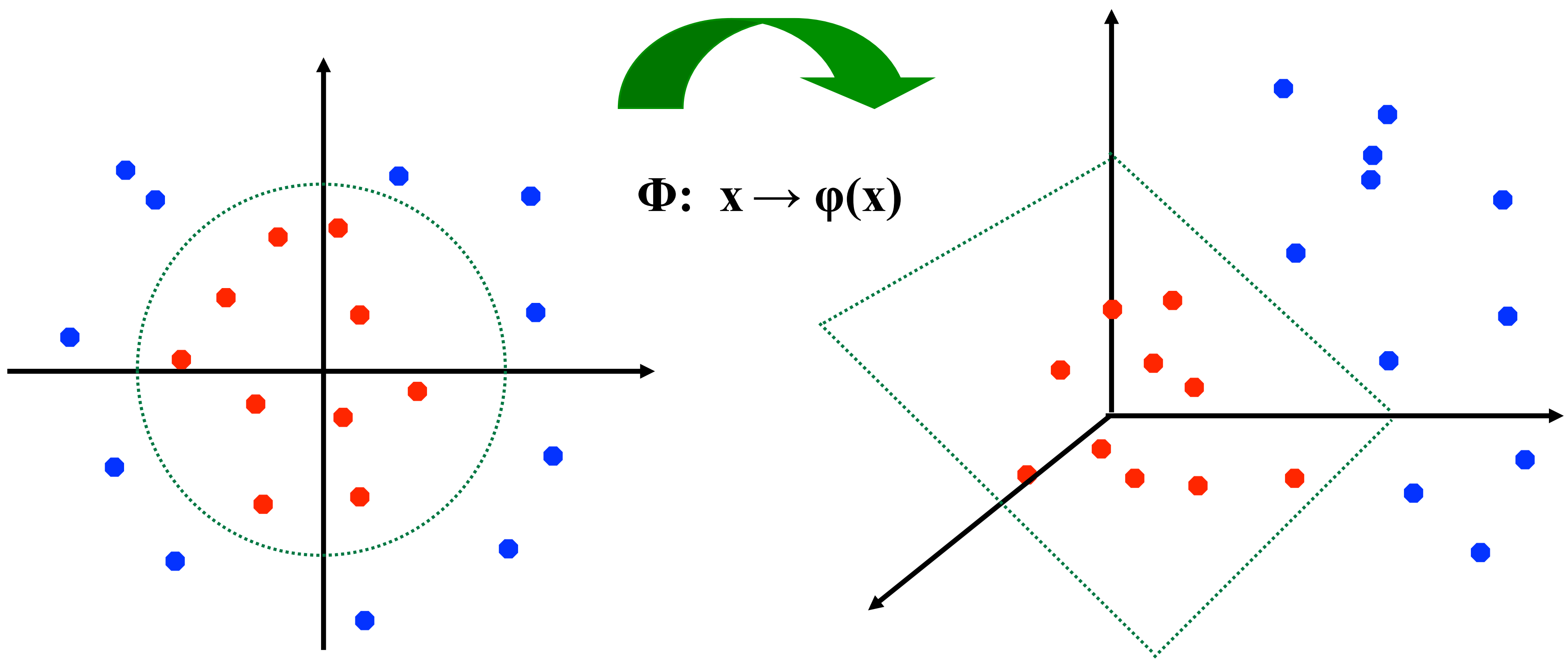


- How about… mapping data to a higher-dimensional space:



Non-linear SVMs

- General idea: the original input space can be **mapped** to some higher-dimensional feature space where the training set is separable:



The “Kernel Trick”

- The linear classifier relies on dot product between vectors $K(x_i,x_j)=x_i^T x_j$

- If every data point is mapped into high-dimensional space via some transformation Φ: $x \rightarrow \varphi(x)$, the dot product becomes:

$$K(x_i,x_j)= \varphi(x_i)^T \varphi(x_j)$$

- A *kernel function* is a similarity function that corresponds to an inner product in some expanded feature space.

Non-linear SVMs

- *The kernel trick*: instead of explicitly computing the lifting transformation $\boldsymbol{\varphi}(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_i) \cdot \boldsymbol{\varphi}(\mathbf{x}_j)$$

- This gives a nonlinear decision boundary in the original feature space:

$$\sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

Examples of kernel functions

- Linear:

$$K(x_i, x_j) = x_i^T x_j$$

- Gaussian RBF:

$$K(x_i, x_j) = \exp\left(-\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}\right)$$

- Histogram intersection:

$$K(x_i, x_j) = \sum_k \min(x_i(k), x_j(k))$$

Questions

- What if the features are not 2D?
- What if the data is not linearly separable?
- **What if we have more than just two categories?**

Multi-class SVMs

- Achieve multi-class classifier by combining a number of binary classifiers
- **One vs. all**
  - Training: learn an SVM for each class vs. the rest
  - Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value
- **One vs. one**
  - Training: learn an SVM for each pair of classes
  - Testing: each learned SVM "votes" for a class to assign to the test example

SVM issues

- Choice of kernel
  - Gaussian or polynomial kernel is default
  - if ineffective, more elaborate kernels are needed
  - domain experts can give assistance in formulating appropriate similarity measures
- Choice of kernel parameters
  - e.g. σ in Gaussian kernel, is the distance between closest points with different classifications
  - In the absence of reliable criteria, rely on the use of a validation set or cross-validation to set such parameters
- Optimization criterion – Hard margin v.s. Soft margin
  - series of experiments in which parameters are tested

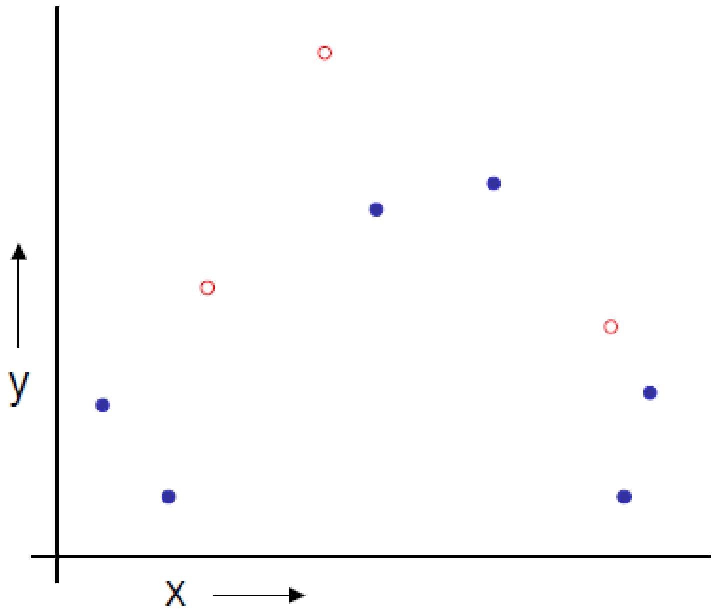
SVM as a classifier

- Advantages
  - Many SVM packages available
  - Kernel-based framework is very powerful, flexible
  - Often a sparse set of support vectors – compact at test time
  - Works very well in practice, even with very small training sample sizes
- Disadvantages
  - No “direct” multi-class SVM, must combine two-class SVMs
  - Can be tricky to select best kernel function for a problem
  - Computation, memory
    - During training time, must compute matrix of kernel values for every pair of examples
    - Learning can take a very long time for large-scale problems

Training - general strategy

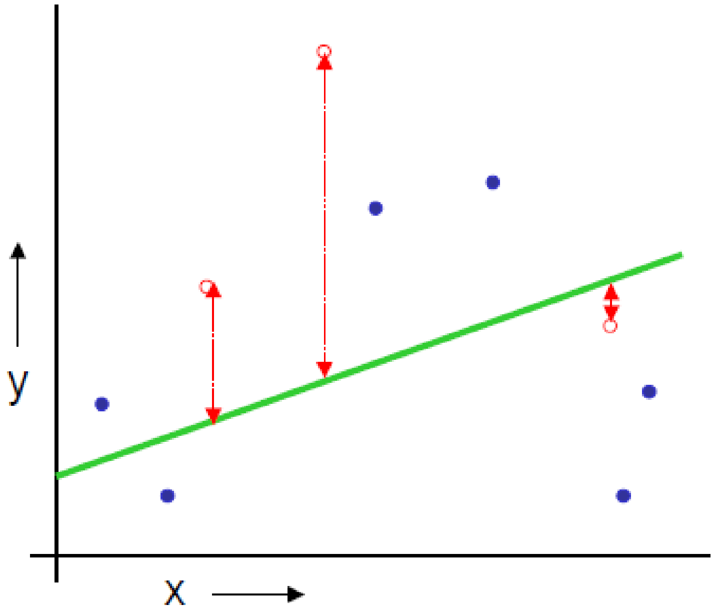
- We try to simulate the real world scenario.
- Test data is our future data.
- Validation set can be our test set - we use it to select our model.
- The whole aim is to estimate the models’ true error on the sample data we have.

Validation set method



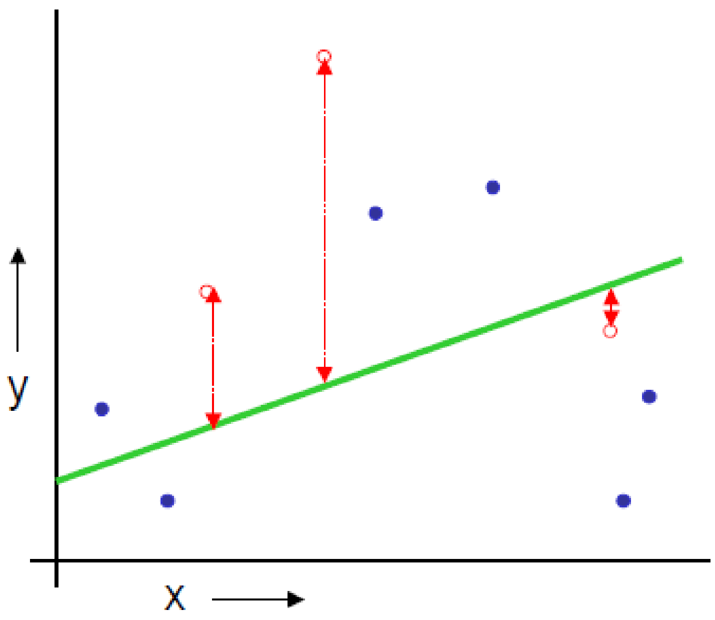
- Randomly split some portion of your data. Leave it aside as the **validation set**
- The remaining data is the **training data**

Validation set method



- Randomly split some portion of your data. Leave it aside as the **validation set**
- The remaining data is the **training data**
- Learn a **model** from the training set

Validation set method



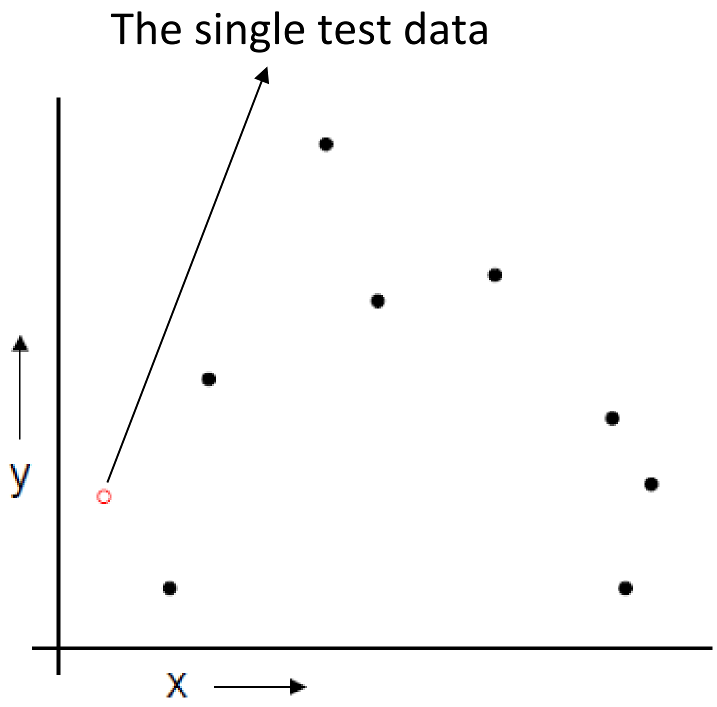
- Randomly split some portion of your data. Leave it aside as the **validation set**
- The remaining data is the **training data**
- Learn a **model** from the training set
- Estimate your future **performance** with the test data

Test set method

- It is simple, however
  - We waste some portion of the data
  - If we do not have much data, we may be lucky or unlucky with our test data

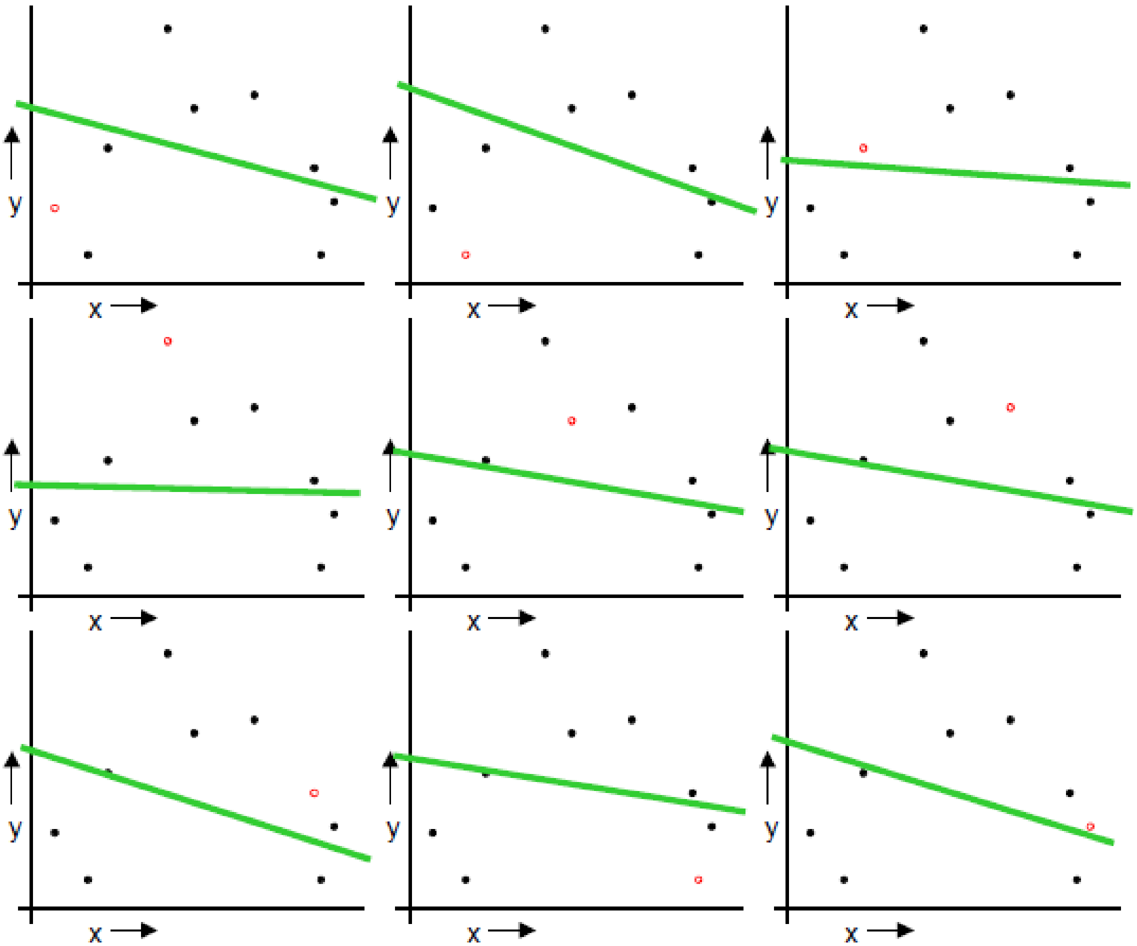
- With **cross-validation** we reuse the data

LOOCV (Leave-one-out Cross Validation)



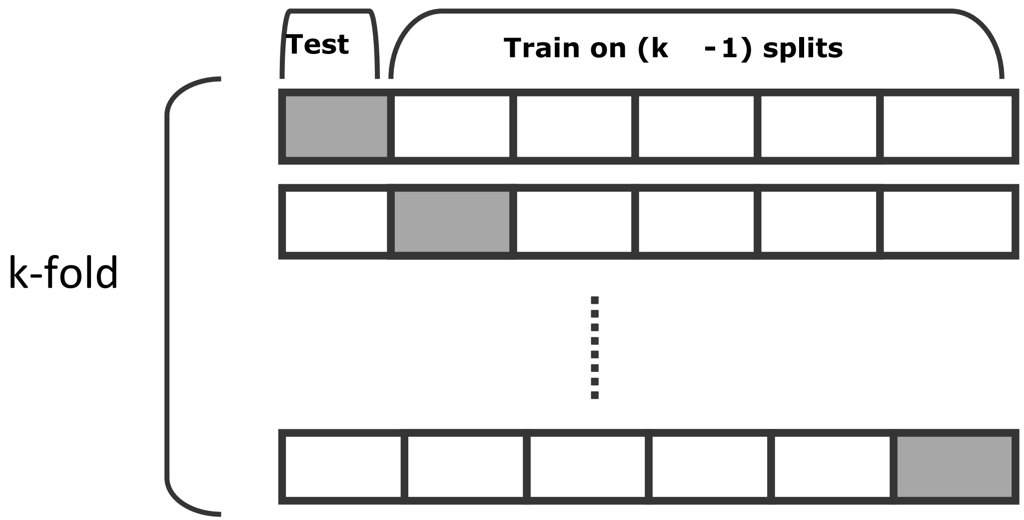
- Let us say we have N data points and k as the index for data points, k=1..N
- Let (x_k, y_k) be the k^{th} record
- Temporarily remove (x_k, y_k) from the dataset
- Train on the remaining N-1 datapoints
- Test the error on (x_k, y_k)
- Do this for each k=1..N and report the mean error.

LOOCV (Leave-one-out Cross Validation)



- Repeat the validation N times, for each of the N data points.
- The validation data is changing each time.

K-fold cross validation



In 3 fold cross validation, there are 3 runs.
In 5 fold cross validation, there are 5 runs.
In 10 fold cross validation, there are 10 runs.

the error is averaged over all runs

References

- Kristen Grauman, Discriminative classifiers for image recognition, http://www.cs.utexas.edu/~grauman/courses/spring2011/slides/lecture22_classifiers.pdf
- Jaime S. Cardoso, Support Vector Machines, http://www.dcc.fc.up.pt/~mcoimbra//lectures/MAPI_1112/CV_1112_6_SupportVectorMachines.pdf
- Andrew Moore, Support Vector Machines Tutorial, http://www.autonlab.org/tutorials/svm.html
- Christopher M. Bishop, Pattern recognition and Machine learning, Springer, 2006.
- Richard O. Duda, Peter E. Hart, David G. Stork, Pattern Classification, John Wiley & Sons, 2001