ifis

Institut für Informationssysteme

Technische Universität Braunschweig

Information Retrieval and Web Search Engines

Lecture 9: Support Vector Machines

Wolf-Tilo Balke

Muhammad Usman

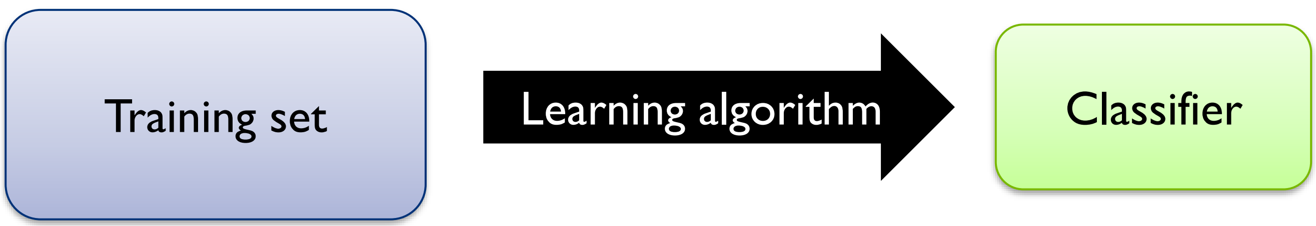
Institut für Informationssysteme

Technische Universität Braunschweig

Supervised Classification

Summary

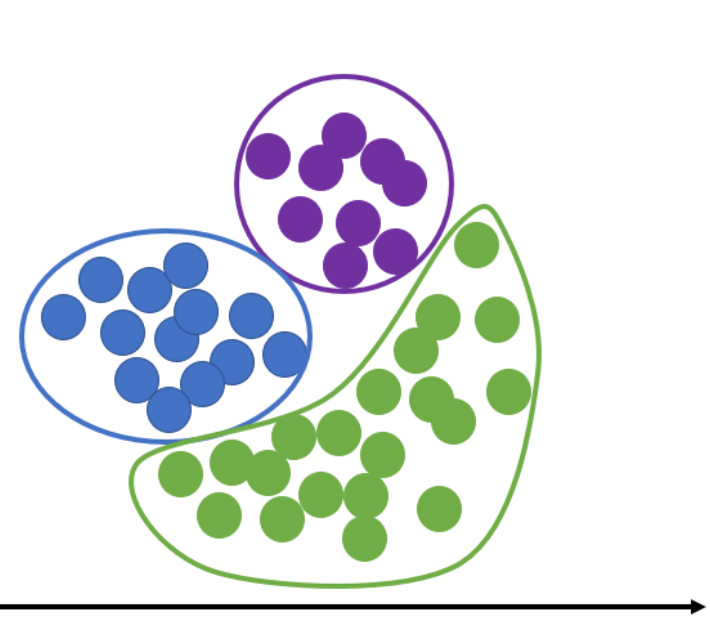
- **Supervised classification:**
  Learn by examples to assign labels to objects.
- The **learning algorithm** takes a training set as input and returns the learned **classification function**



- Some classical approaches:
  - Naïve Bayes
  - Rocchio
  - K-nearest neighbor

Feedback and Classification

1. **Linear SVMs**
2. Nonlinear SVMs
3. Support Vector Machines in IR
4. Overfitting



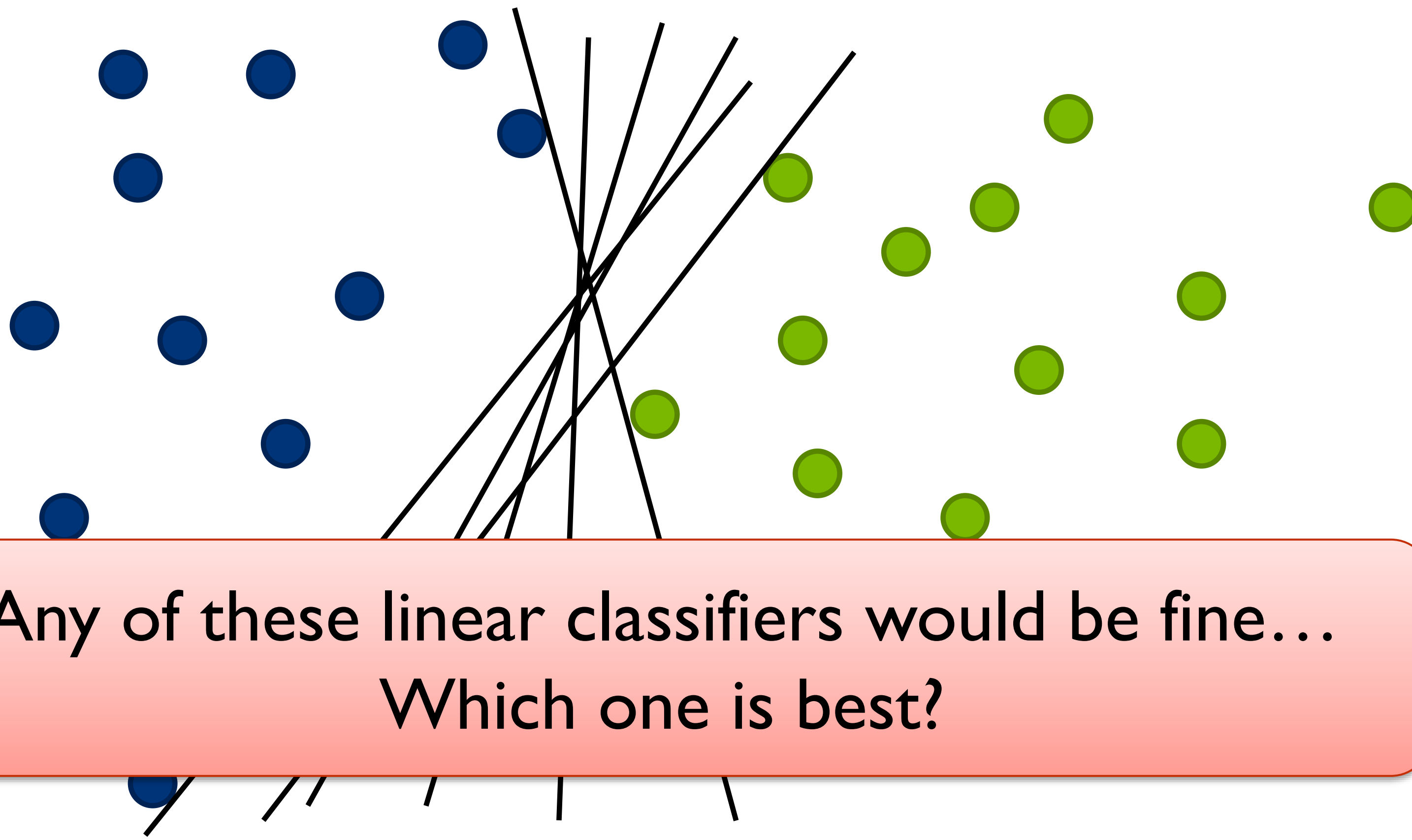
Problem Definition

- **Assumptions:**
  - **Binary classification:**
    Let's assume there are only **two classes**
    (e.g. spam/non-spam or relevant/non-relevant)
  - **Vector representation:**
    Any item to be classified can be represented as a
    ***d*-dimensional real vector**

- **Task:**
  - Find a **linear classifier** (i.e. a hyperplane) that divides the space $\mathbb{R}^d$ into two parts

Example

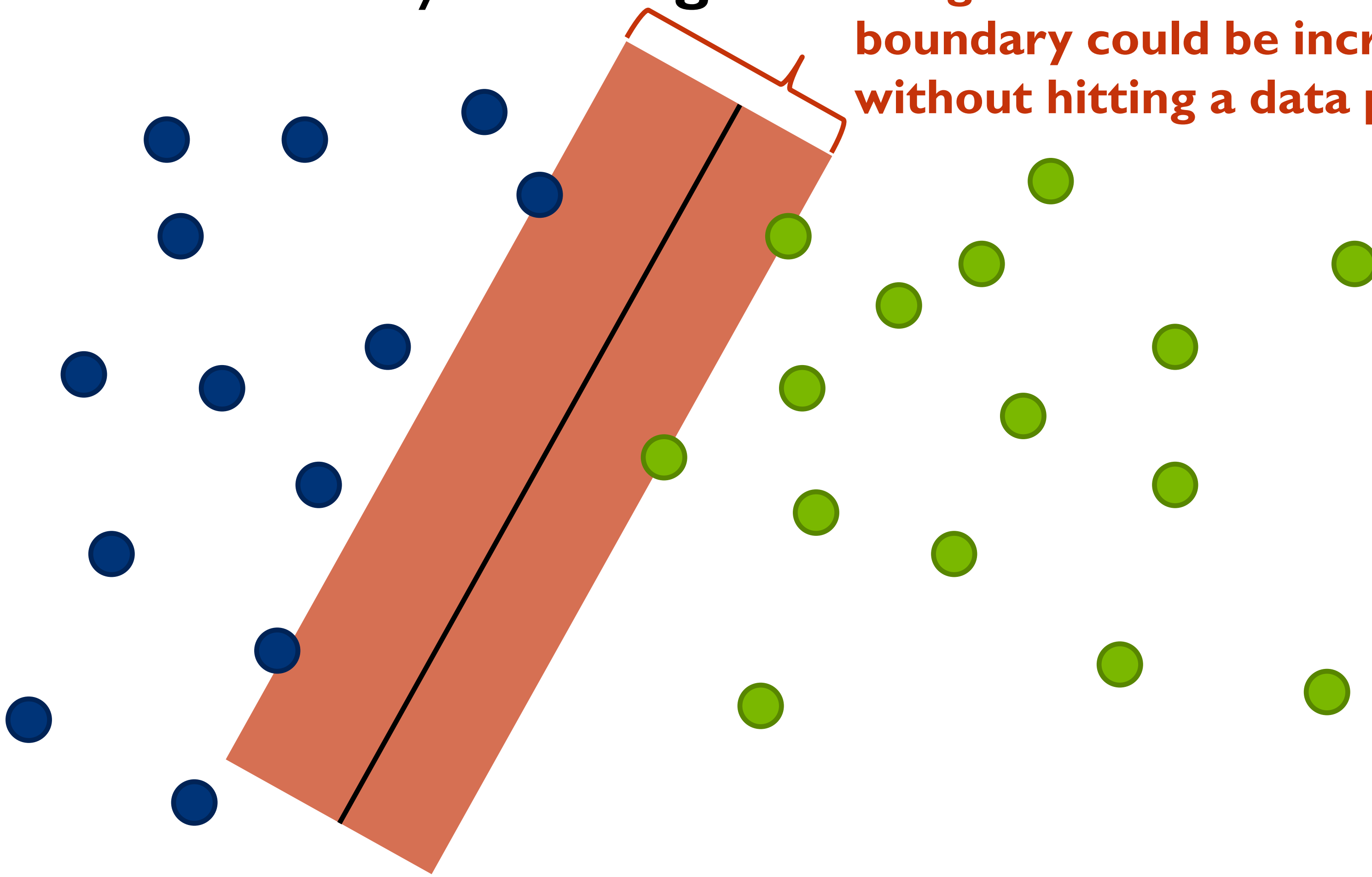
- **A two-dimensional example training set.**
- **Task: Separate it by a straight line!**



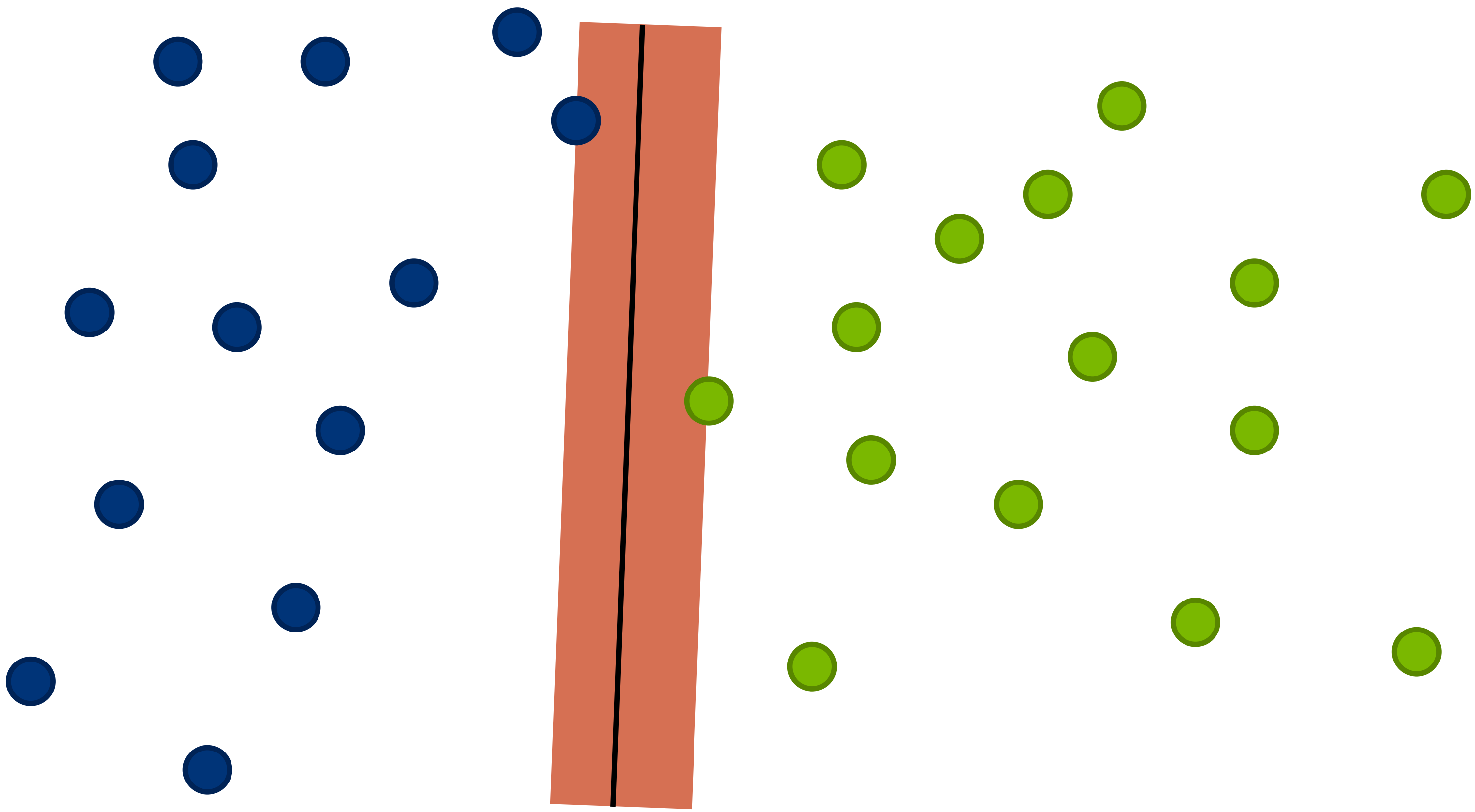
Any of these linear classifiers would be fine…
Which one is best?

Margin

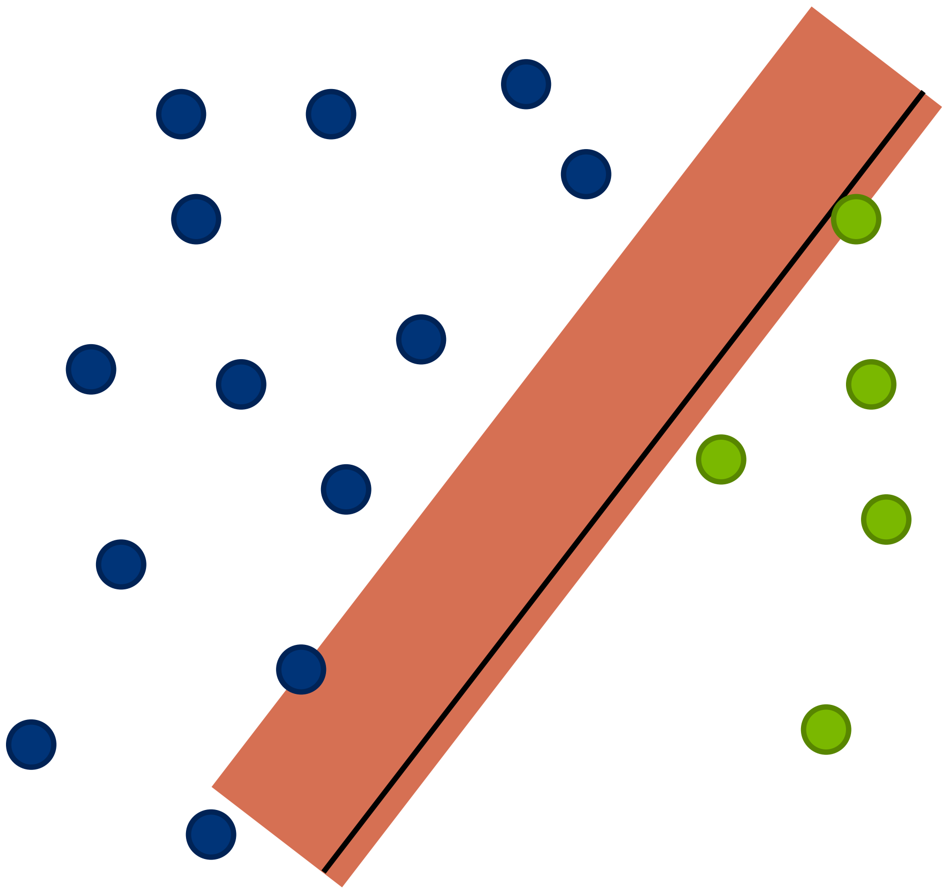
- **Idea:** Measure the **quality** of a linear classifier by its **margin!**



Margin

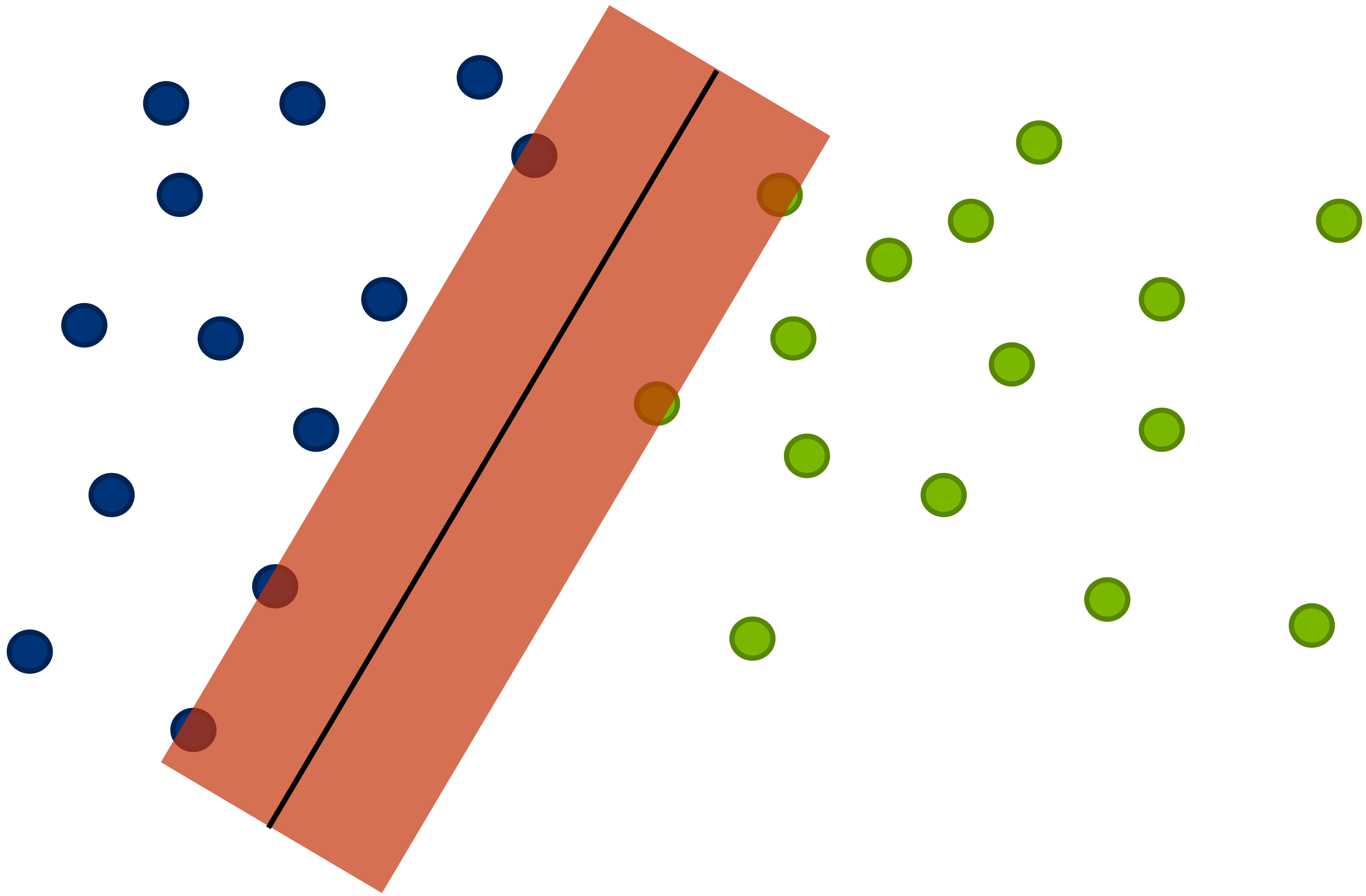


Margin



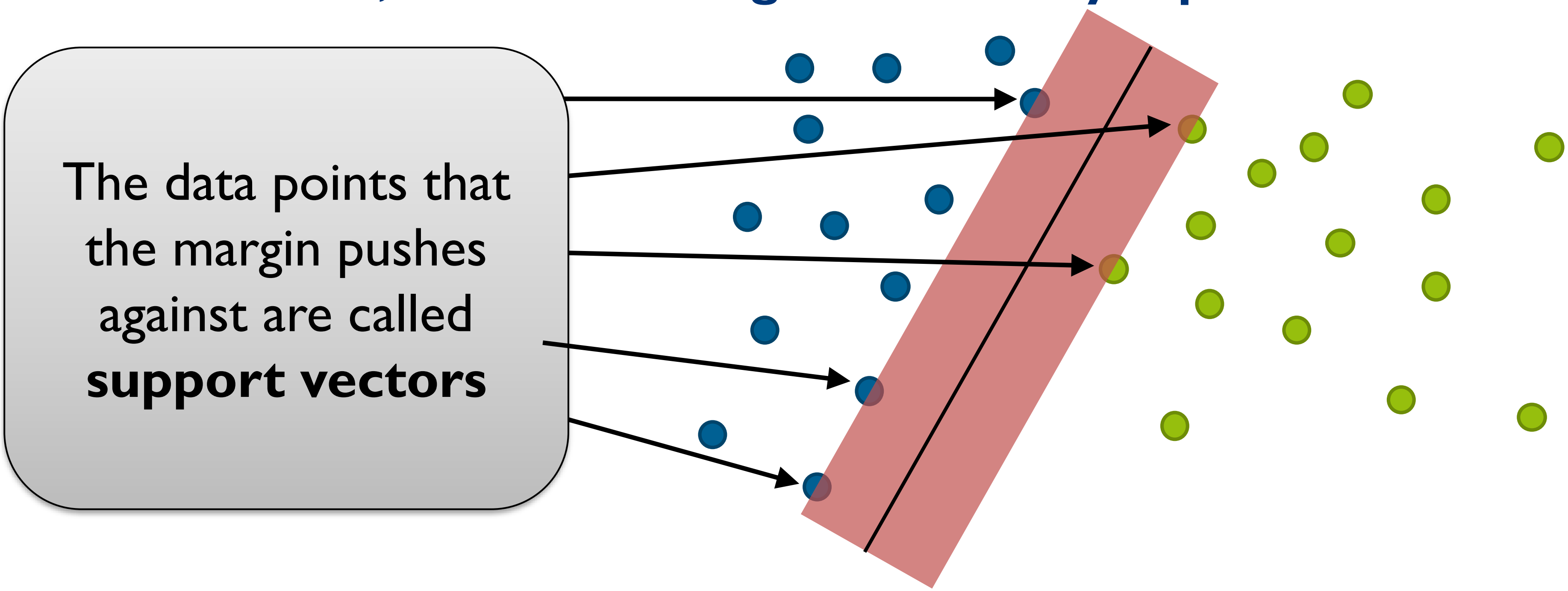
Maximum Margin Classifiers

- A **maximum margin classifier** is the linear classifier with a maximum margin



Maximum Margin Classifiers

- The **maximum margin classifier** is the simplest kind of support vector machine, called a **linear SVM**
  - **Let's assume for now that there always is such a classifier, i.e. the training set is linearly separable!**

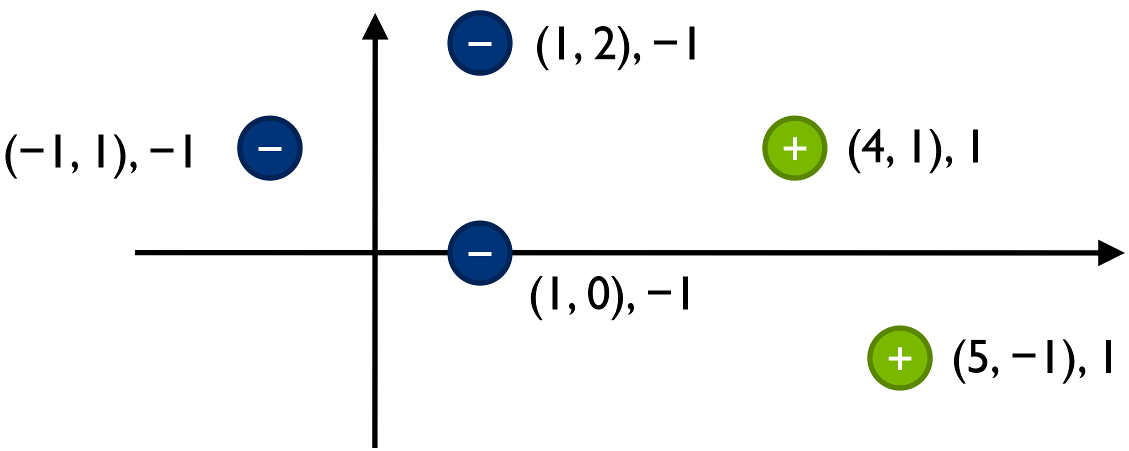


Maximum Margin Classifiers

- Why **maximum margin?**
  - It's **intuitive** to divide the two classes by a large margin
  - The largest margin **guards best against small errors** in choosing the "right" separator
  - This approach is **robust** since usually only a small fraction of all data points are support vectors
  - There are some **theoretical arguments** why this is a good thing
  - Empirically, **it works very well**

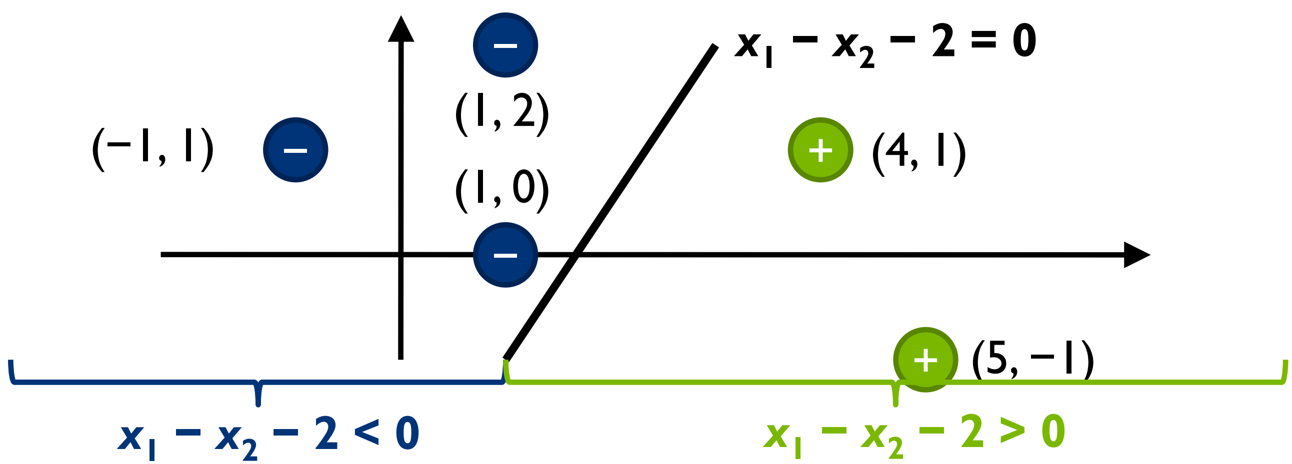
Finding MM Classifiers

- **How to formalize this approach?**
- Training data:
  - Let there be ***n* training examples**
  - The ***i*-th training example** is a **pair (y_i, z_i)**, where **y_i is a *d*-dimensional real vector** and **$z_i \in \{-1, 1\}$**
  - **"−1"** stands for the **first class** and **"1"** stands for the **second class**



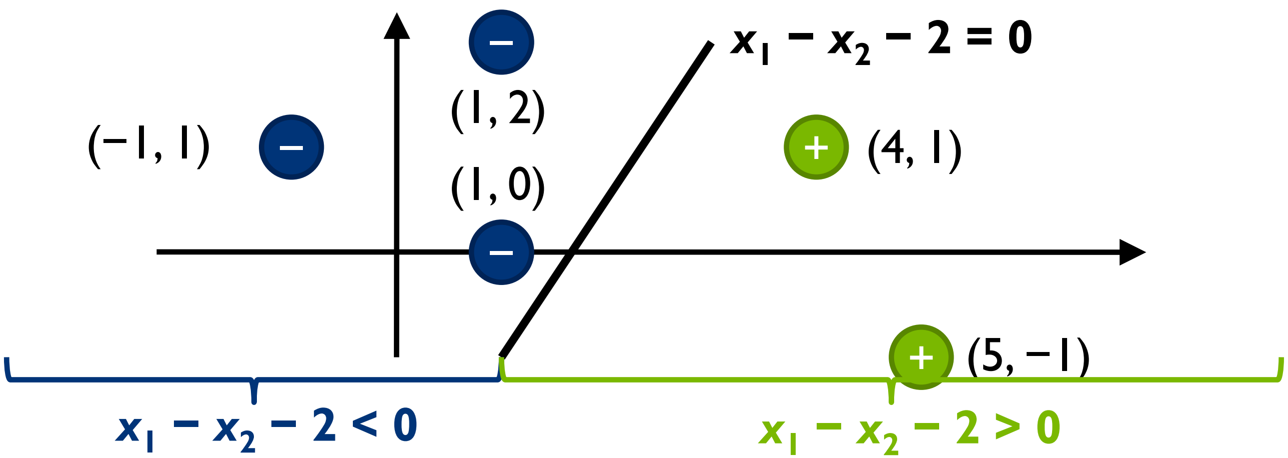
Finding MM Classifiers

- What's a valid linear separator?
- Any **hyperplane** can be defined by a real **row vector *w*** and a **scalar *b***
  - The set of points located on the hyperplane is given by $\{x \in \mathbb{R}^d \mid w \cdot x + b = 0\} = \{x \in \mathbb{R}^d \mid w_1x_1 + w_2x_2 + \cdots + w_dx_d + b = 0\}$
  - *w* is a **normal vector** of the hyperplane, i.e. *w* is **perpendicular** to it
  - *b* represents a shift from the origin of the coordinate system



Finding MM Classifiers

- Therefore, any valid separating hyperplane (w, b) must satisfy the following **constraints,** for any $i = 1, \ldots, n$:
  - If $z_i = -1$, then $w \cdot y_i + b < 0$
  - If $z_i = 1$, then $w \cdot y_i + b > 0$

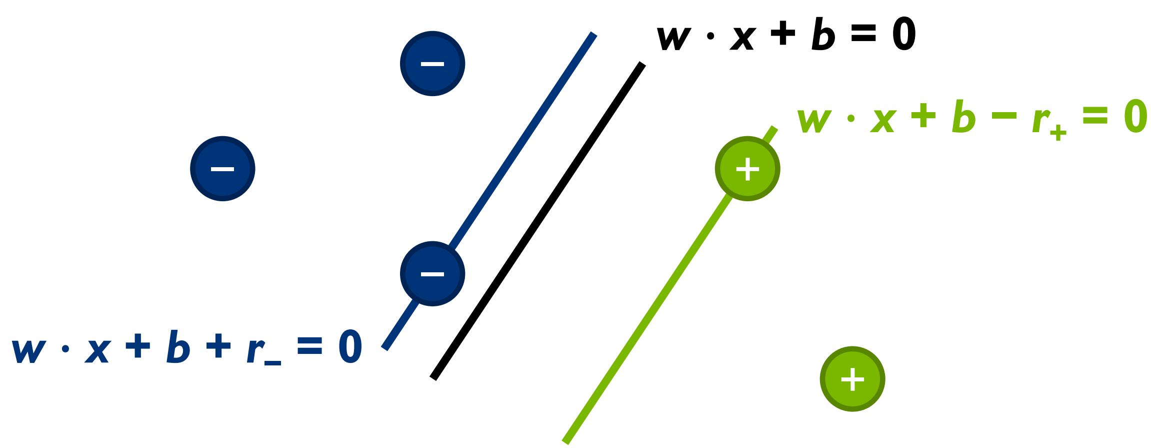


Finding MM Classifiers

- Furthermore, if (w, b) is a valid separating hyperplane, then there are scalars $r_+ > 0$ and $r_- > 0$ such that

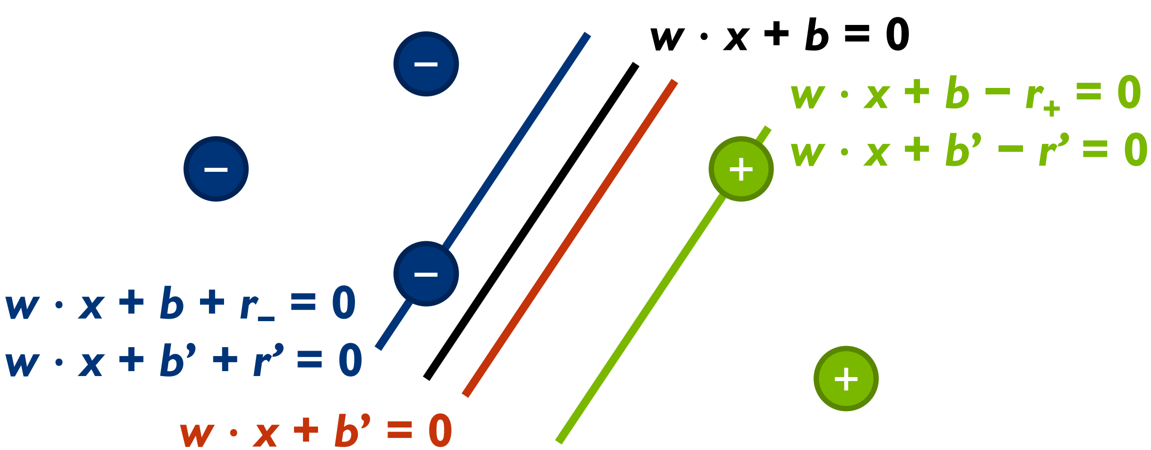
  $w \cdot x + b + r_- = 0$ and $w \cdot x + b - r_+ = 0$

  are the hyperplanes that define the boundaries to the "−1" class and the "1" class, respectively
  - **The support vectors are located on these hyperplanes!**



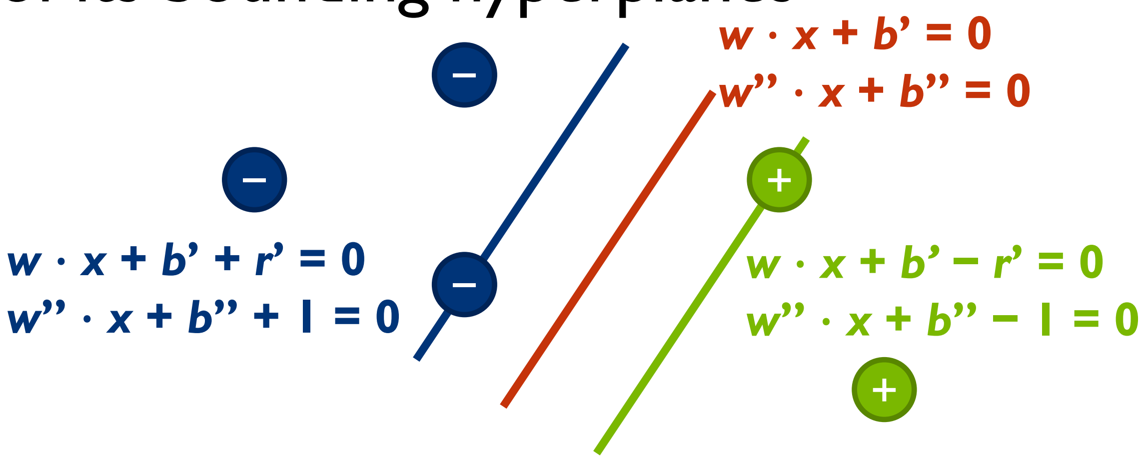
Finding MM Classifiers

- Let (w, b) be a valid separating hyperplane with scalars r_+ and r_- as defined above
- **Observation 1:**
  Define $\boldsymbol{b' = b + (r_- - r_+) / 2}$. Then, the hyperplane $\boldsymbol{w \cdot x + b' = 0}$ is a **valid separating hyperplane** with equal shift constants $\boldsymbol{r' = (r_- - r_+) / 2}$ to its bounding hyperplanes (the margin width is the same)



Finding MM Classifiers

- Now, divide *w*, *b*', and *r*' by *r*'
- This does not change any of the three hyperplanes…
- **Observation 2:**
  Define **$w'' = w / r'$ and $b'' = b' / r'$.**
  Then, the hyperplane $\mathbf{w'' \cdot x + b'' = 0}$ is a **valid separating hyperplane** with **shift constant 1** to each of its bounding hyperplanes



Finding MM Classifiers

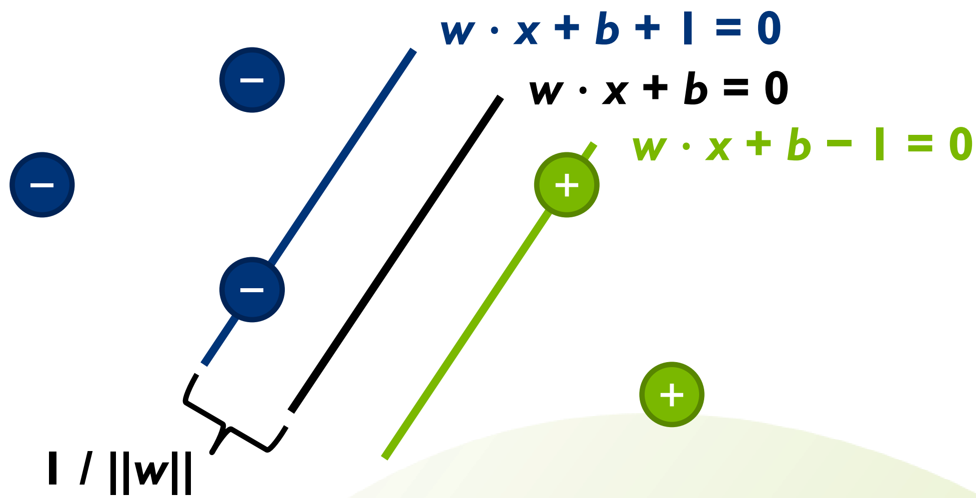
- **Corollary (normalization):**
  If there exists a valid separating hyperplane (w, b), then there always is a hyperplane (w'', b'') such that
  - (w'', b'') is a valid separating hyperplane
  - (w, b) and (w'', b'') have equal margin widths
  - the bounding hyperplanes of (w'', b'') are shifted away by 1
- Therefore, to find a maximum margin classifier, we can **limit the search** to all hyperplanes of this special type
- **Further advantage:**
  It seems to be a good idea to use a linear classifier that lies equally spaced between its bounding hyperplanes

Finding MM Classifiers

- Our search space then consists of all pairs (w, b) such that
  - $w \in \mathbb{R}^d$
  - $b \in \mathbb{R}$
  - For any i = 1, …, n:
    If $z_i = -1$, then $w \cdot y_i + b \leq -1$
    If $z_i = 1$, then $w \cdot y_i + b \geq 1$
  - There is an i such that $z_i = -1$ and $w \cdot y_i + b = -1$
  - There is an i such that $z_i = 1$ and $w \cdot y_i + b = 1$

- Now, what is the **margin width** of such a hyperplane?

Finding MM Classifiers



- **Linear algebra:**
  The distance of a hyperplane $w \cdot x + b = 0$ to the origin of coordinate space is $|b| \,/\, ||w||$
- Therefore, the margin width is $2 \,/\, ||w||$
- Consequently, our goal is to **maximize the margin width** subject to the constraints from the previous slide

Finding MM Classifiers

- We arrive at the following **optimization problem** over all $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$:

  Maximize 2 / ||*w*|| subject to the following constraints:
  - For any $i = 1, \ldots, n$:
    If $z_i = -1$, then $w \cdot y_i + b \leq -1$
    If $z_i = 1$, then $w \cdot y_i + b \geq 1$
  - There is an i such that $z_i = -1$ and $w \cdot y_i + b = -1$
  - There is an i such that $z_i = 1$ and $w \cdot y_i + b = 1$
- Note that due to the "maximize the margin" goal, **the last two constraints are not needed anymore** since any optimal solution satisfies them anyway

Finding MM Classifiers

- The problem then becomes:

  Maximize 2 / ||*w*|| over all $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$
  subject to the following constraints:

  - For any $i = 1, \ldots, n$:
    If $z_i = -1$, then $w \cdot y_i + b \leq -1$
    If $z_i = 1$, then $w \cdot y_i + b \geq 1$

- Instead of maximizing 2 / ||*w*||, we also could minimize ||*w*||, or even **minimize 0.5 $||w||^2$**
  - Squaring avoids the square root within ||*w*||
  - The factor 0.5 brings the problem into some standard form

Finding MM Classifiers

- The problem then becomes:

  Minimize 0.5 $||\mathbf{w}||^2$ over all $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ subject to the following constraints:

  – For any $i = 1, \ldots, n$:
  If $z_i = -1$, then $w \cdot y_i + b \leq -1$
  If $z_i = 1$, then $w \cdot y_i + b \geq 1$

- The two constraints can be combined into a single one:
  – For any $i = 1, \ldots, n$:
  $z_i \cdot (w \cdot y_i + b) - 1 \geq 0$

Finding MM Classifiers

- Finally:
  Minimize 0.5 $||w||^2$ over all $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$
  subject to the following constraints:
  - For any $i = 1, \ldots, n$:
    $z_i \cdot (w \cdot y_i + b) - 1 \geq 0$

- This is a so-called quadratic programming (QP) problem
  - There are many standard methods to find the solution…
- QPs that emerge from an SVM have a special structure, which can be exploited to speed up computation

Duality

- We will not discuss in detail how QPs emerging from SVMs can be solved
- But we will give a quick impression of what can be done
- By introducing **Lagrange multipliers** (already known to us from Rocchio's relevance feedback) and doing some transformations, one finally arrives at the following optimization problem:

Maximize (in $\alpha \in \mathbb{R}^n$)

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \mathbf{z}_i \mathbf{z}_j \mathbf{y}_i^{\mathsf{T}} \mathbf{y}_j$$

subject to $\alpha_i \geq 0$, for any i, and $\alpha_1 z_1 + \cdots + \alpha_n z_n = 0$

Duality

- Maximize (in $\alpha \in \mathbb{R}^n$)

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \mathbf{z}_i \mathbf{z}_j \mathbf{y}_i^{\mathsf{T}} \mathbf{y}_j$$

  subject to $\alpha_i \geq 0$, for any i, and $\alpha_1 z_1 + \cdots + \alpha_n z_n = 0$
- This problem is called the **dual optimization problem** and has the same optimal solutions as the original problem (if one ignores α); but usually it is easier to solve
- **Important property:**
  If $\alpha_i > 0$ in a solution of the above problem, then the corresponding data point y_i is a **support vector**
  - Consequence: Usually, most α_i are zero, which makes things easy

Duality

- The classification function then becomes:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i z_i y_i^{\mathrm{T}} x + b\right)$$

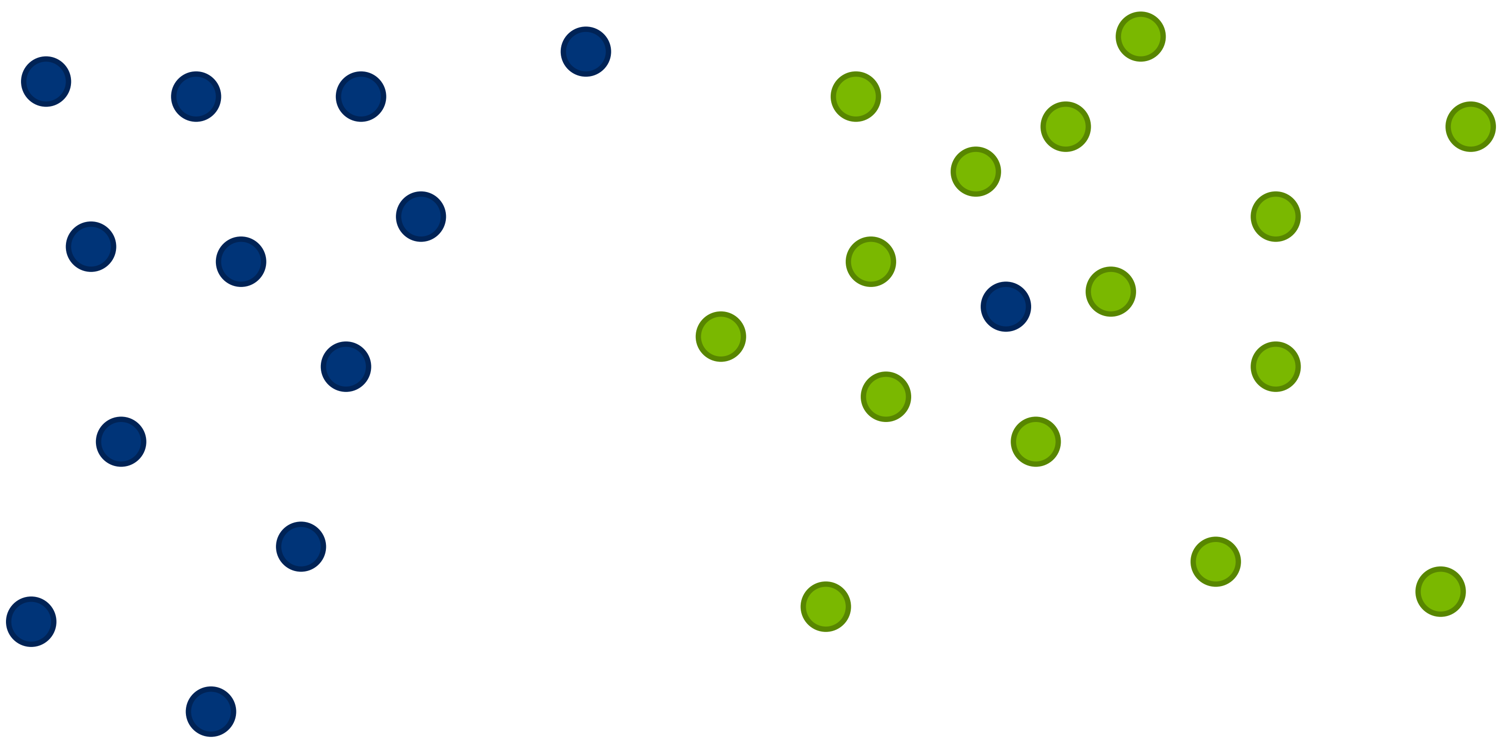
- *b* can be computed as follows, using any *i* such that $\alpha_i > 0$:

$$b = z_i - \sum_{j=1}^{n} z_j \alpha_j y_i^{\mathrm{T}} y_j$$

- Note that *f* can be directly expressed in terms of the support vectors
- Furthermore, computing *f* basically depends on **scalar products** of vectors ($y_i^{\mathrm{T}} \cdot x$), which is a **key feature** in advanced applications of SVMs

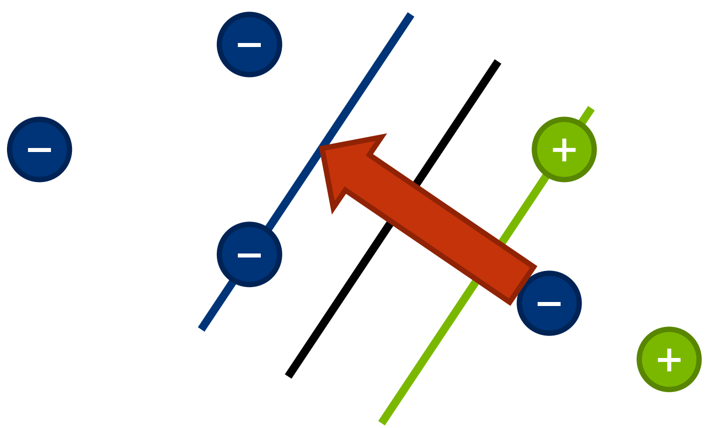
Soft Margin Classification

- At the beginning we assumed that our training data set is **linearly separable…**
- What if it looks like this?



Soft Margin Classification

- So-called **soft margins** can be used to handle such cases
- We allow the classifier to make some mistakes on the training data
- Each **misclassification** gets assigned an **error,** the **total classification error** then is to be minimized



Soft Margin Classification

- We arrive at a new optimization problem
- Minimize $0.5\ ||w||^2$ **+ C · (β_1 + ··· + β_n)** over all (w, b, $\boldsymbol{\beta}$) satisfying $w \in \mathbb{R}^d$, $b \in \mathbb{R}$, **and $\boldsymbol{\beta} \in \mathbb{R}^n$** subject to the following constraints:
  - For any $i = 1, \ldots, n$:
    $\boldsymbol{\beta_i \geq 0}$
    $z_i \cdot (w \cdot y_i + b) - 1 \geq \boldsymbol{-\beta_i}$
- If the i-th data point gets misclassified by β_i, the price we pay for it is $C \cdot \beta_i$
- C is a positive constant that regulates how expensive errors should be

Soft Margin Classification

- With soft margins, we can drop the assumption of linear separability

- The corresponding dual problem is:
  Maximize (in $\alpha \in \mathbb{R}^n$)

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \mathbf{y}_i^{\mathrm{T}} \mathbf{y}_j$$

  subject to $\mathbf{C} \geq \alpha_i \geq 0$, for any i, and $\alpha_1 z_1 + \cdots + \alpha_n z_n = 0$

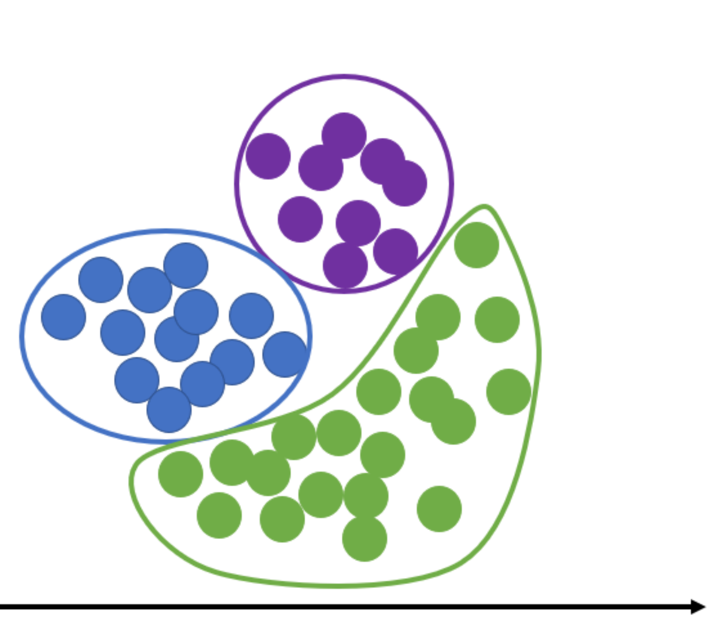
- Note that only an **upper bound on $\boldsymbol{\alpha}$** is added here
  - Still, it is possible to find solutions efficiently

Multiple Classes

- At the beginning, we also assumed that there are only **two classes** in the training set
- How to handle more than that?
- Some ideas:
  - **One-versus-all classifiers:**
    Build an SVM for any class that occurs in the training set;
    To classify new items, choose the greatest margin's class
  - **One-versus-one classifiers:**
    Build an SVM for any pair of classes in the training set;
    To classify new items, choose the class selected by most SVMs
  - **Multiclass SVMs:**
    (complicated, will not be covered in this course)

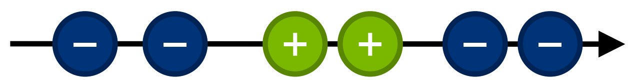
Feedback and Classification

1. Linear SVMs
2. **Nonlinear SVMs**
3. Support Vector Machines in IR
4. Overfitting

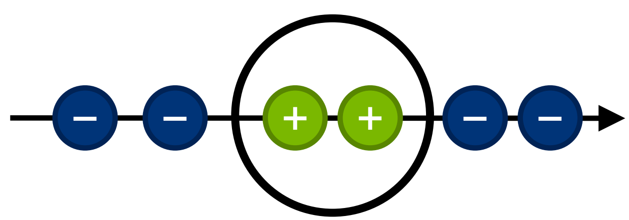


Nonlinear SVMs

- Now we are able to handle linearly separable data sets (perhaps with a few exceptions or some noise)
- But what to do with this (one-dimensional) data set?

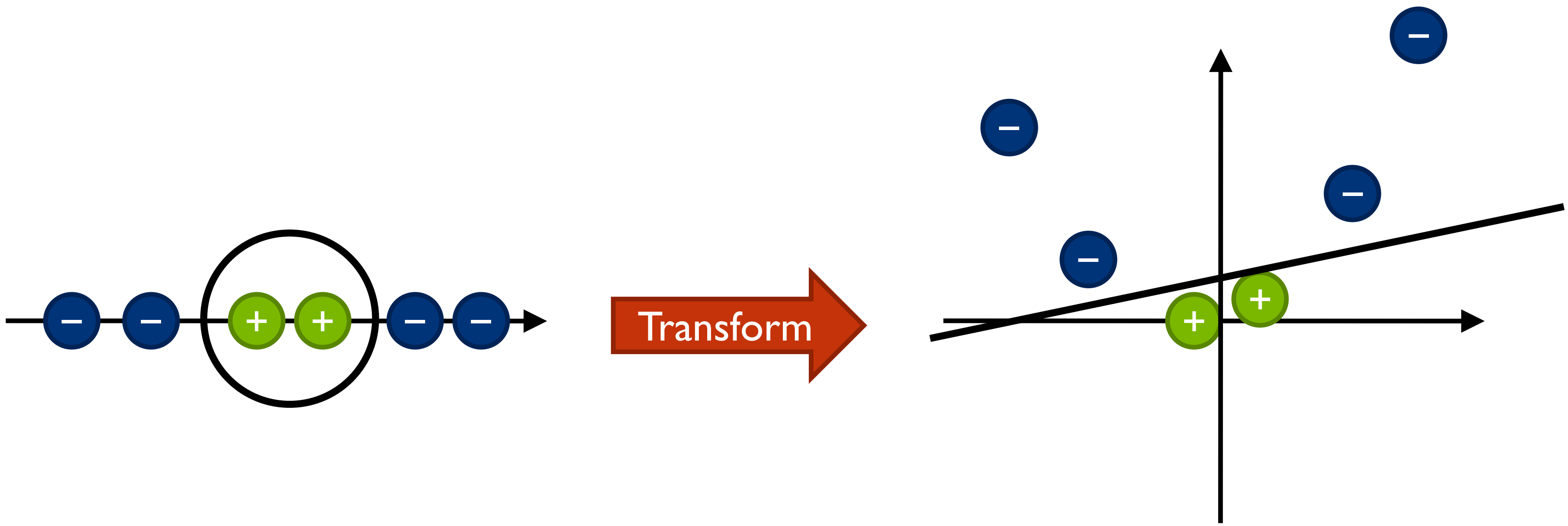


- Obviously, it is not linearly separable, and the reason for that is not noise…
- What we want to do:



Nonlinear SVMs

- **Solution:**
  Transform the data set into some **higher-dimensional space** and do a **linear classification** there…



Nonlinear SVMs

- **But…**
  When working in **high-dimensional spaces,** computing the transformation and solving the corresponding optimization problem will be **horribly difficult**
- What can we do about it?

- **Observation:** There are no problems at all if we are able to compute scalar products in the high-dimensional space efficiently…

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j y_i^T y_j \qquad f(x) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i z_i y_i^T x + b \right)$$

Nonlinear SVMs

- The key technique here is called the **"kernel trick"**
- Let $\boldsymbol{h} : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ be some function that maps our original *d*-dimensional data into some *d'*-dimensional space
  - Typically $d' \gg d$ holds
- To deal with our optimization problem and be able to do classification afterwards, we must be able to quickly compute the following expressions:

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j h(y_i)^T h(y_j)$$

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i z_i h(y_i)^T h(x) + b \right) \qquad b = z_i - \sum_{j=1}^{n} z_j \alpha_j h(y_i)^T h(y_j)$$

Nonlinear SVMs

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j h(y_i)^T h(y_j)$$

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i z_i h(y_i)^T h(x) + b \right)$$

$$b = z_i - \sum_{j=1}^{n} z_j \alpha_j h(y_i)^T h(y_j)$$

- Note that we only need to compute **scalar products** in the high-dimensional space…
- If *h* is some special type of mapping (e.g. polynomial or Gaussian), there are computationally simple **kernel functions** available, which correspond to the result of scalar products in *h*'s range
- **A polynomial transformation of degree 2:**

$$h(x) \cdot h(x') = (1 + x \cdot x')^2$$

LIBSVM

Detour

LIBSVM -- A Library for Support Vector Machines

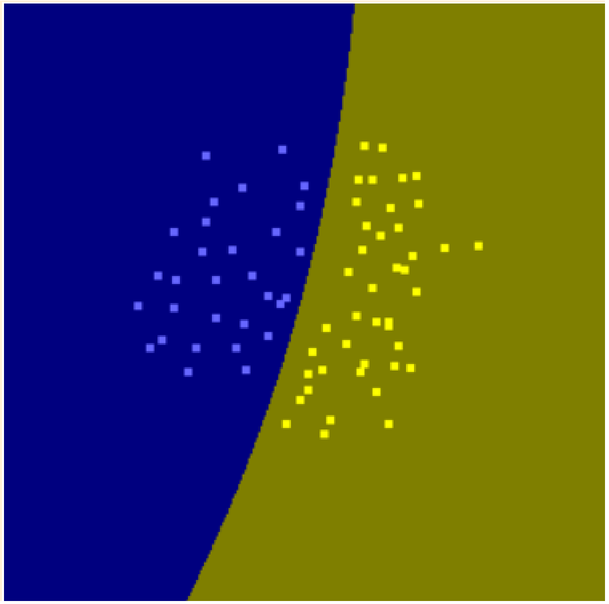
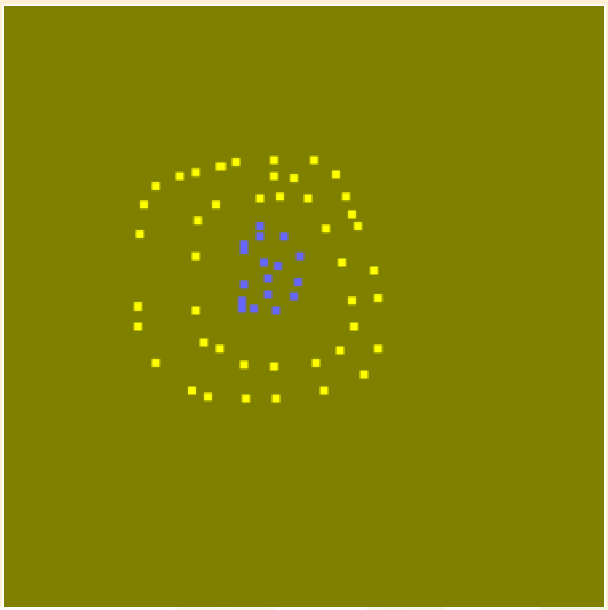
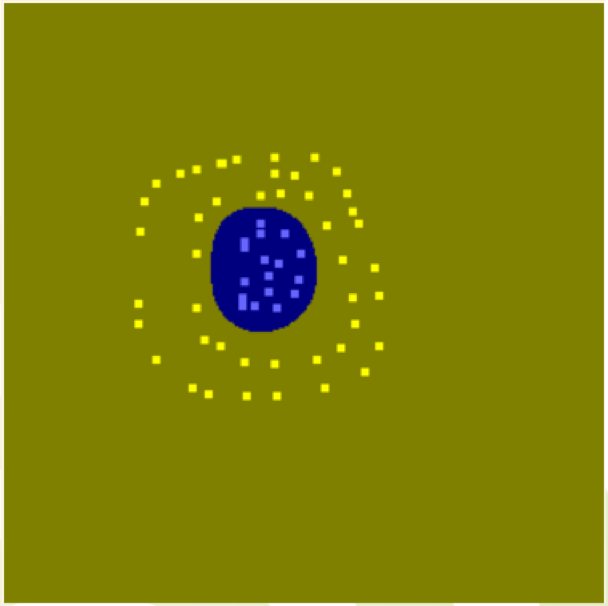
- Chih-Chung Chang and Chih-Jen Lin
- Version 3.31, 25th Feb 2023
- http://www.csie.ntu.edu.tw/~cjlin/libsvm/

LIBSVM

- Different SVM formulations
- Efficient multi-class classification
- Various kernels
- Weighted SVM for unbalanced data
- GUI demonstrating SVM classification and regression
- Python, R, MATLAB, Perl, Ruby, C#, etc.
- pip install libsvm-official

Nonlinear SVMs

Detour

Linear Data Linear Classification	Non-Linear Data Linear Classification	Non-Linear Data Non-Linear Classification
		

Nonlinear SVMs

Detour

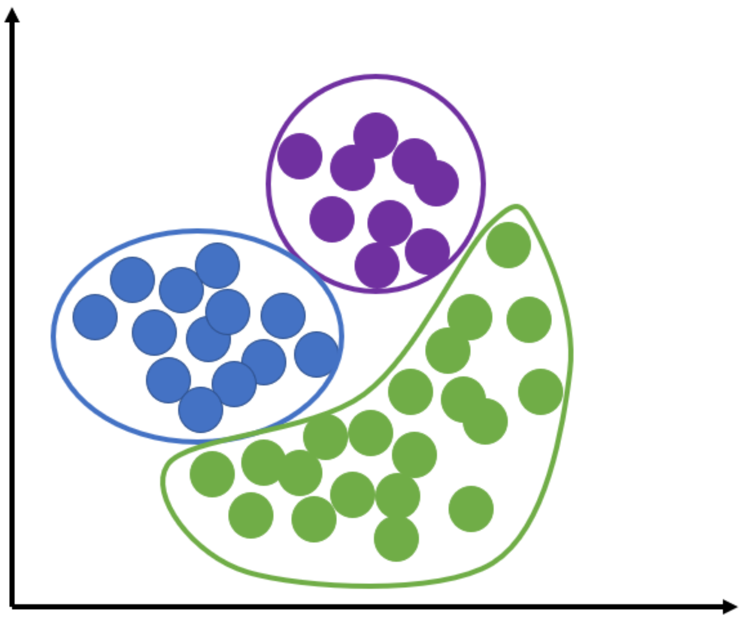
Applications:

- Speaker/speech recognition
- Predicting protein structures
- Breast cancer prognosis
- Stock forecast

http://clopinet.com/isabelle/Projects/SVM/

Feedback and Classification

1. Linear SVMs
2. Nonlinear SVMs
3. **Support Vector Machines in IR**
4. Overfitting



Text Classification

- An important application of SVMs in information retrieval is **text classification**
- Typically, this means **automatically assigning topics** to new documents based on a training collection of manually processed documents
  - But there are also many other applications, e.g. spam detection
- In SVMs, **document representations** known from the **vector space model** can be used
  - Plus additional features, e.g. document length
- Although the **dimensionality is very high** then, this usually is not a big problem since most document vectors are **very sparse**

Text Classification

- SVMs have been successfully applied in text classification on **small and medium-sized document collections**
- Some results by Joachims (1998) from experiments on the Reuters-21578 data set (F-measure with $\alpha = 0.5$)

CATEGORIES	NBAYES	ROCCHIO	DEC. TREES	KNN	LINEAR SVM		RBF-SVM
					C=0.5	C = 1.0	
EARN	96.0	96.1	96.1	97.8	98.0	98.2	98.1
ACQ	90.7	92.1	85.3	91.8	95.5	95.6	94.7
MONEY-FX	59.6	67.6	69.4	75.4	78.8	78.5	74.3
GRAIN	69.8	79.5	89.1	82.6	91.9	93.1	93.4
CRUDE	81.2	81.5	75.5	85.8	89.4	89.4	88.7
TRADE	52.2	77.4	59.2	77.9	79.2	79.2	76.6
INTEREST	57.6	72.5	49.1	76.7	75.6	74.8	69.1
SHIP	80.9	83.1	80.9	79.8	87.4	86.5	85.8
WHEAT	63.4	79.4	85.5	72.9	86.6	86.8	82.4
CORN	45.2	62.2	87.7	71.4	87.5	87.8	84.6
MICROAVG.	72.3	79.9	79.4	82.6	86.7	87.5	86.4

Learning to Rank

- A very recent application of SVM in information retrieval is called **"Learning to Rank"**
- Here, a special type of SVMs is used: **Ranking SVMs**
- The training set consists of *n* pairs of documents (y_i, y_i')
- Each such pair expresses that document **y_i is preferred to y_i'** with respect to some fixed query shared by all training pairs
- **Example** training set for query "Viagra":
  - Wikipedia's entry "Viagra" is preferred to some spam page
  - Wikipedia's entry "Viagra" is preferred to the manufacturer's official page
  - The manufacturer's official page is preferred to some spam page

Learning to Rank

- **The task in Learning to Rank:**
  Find a **ranking function** that assigns a numerical score $s(d)$ to each document d based on its vector representation such that $\boldsymbol{s(d) > s(d')}$ if and only if **document *d* is preferred to document *d'***

- A straightforward approach are **linear ranking functions**, i.e. $\boldsymbol{s(d) = w \cdot d}$, for some row vector w
- This reminds us of SVMs…

Learning to Rank

- **An SVM formulation of our task is…**

  Minimize 0.5 $||w||^2$ over all $w \in \mathbb{R}^d$
  subject to the following constraints:
  - For any $i = 1, \ldots, n$:
    $\underbrace{w \cdot y_i}_{\text{score of } y_i} \geq \underbrace{w \cdot y_i'}_{\text{score of } y_i'} + 1$

  Enforce a **standard margin** of 1
  between each pair of scores

- The constraint is equivalent to $\mathbf{w \cdot (y_i - y_i') - 1 \geq 0}$,
  which looks familiar…

- Of course, we could also use a **soft margin** or **nonlinear scoring functions** here…

Learning to Rank

- **Where to get the preference pairs from?**
- Idea from Joachims (2002):
  - Users tend to **linearly read a search engine's result lists** down from its beginning
  - If users **click the *r*-th result but do not click the (*r* − 1)-th,** then document r likely to be preferred to document *r* − 1

Learning to Rank

- Then:
  1. Compute an initial result list using some retrieval algorithm
  2. Collect user clicks
  3. Learn a ranking function
  4. Incorporate the ranking function into the retrieval process, i.e. re-rank the result list

- Of course, one could use the ranking information already in computing the initial result list
  - … if user feedback on similar queries is available
  - … if feedback from different users on the same query is available

Handwritten Digits

Detour

- Particularly popular: Recognition of **handwritten digits**

Handwritten Digits

Detour

- Results
  - Taken from Decoste/Schölkopf:

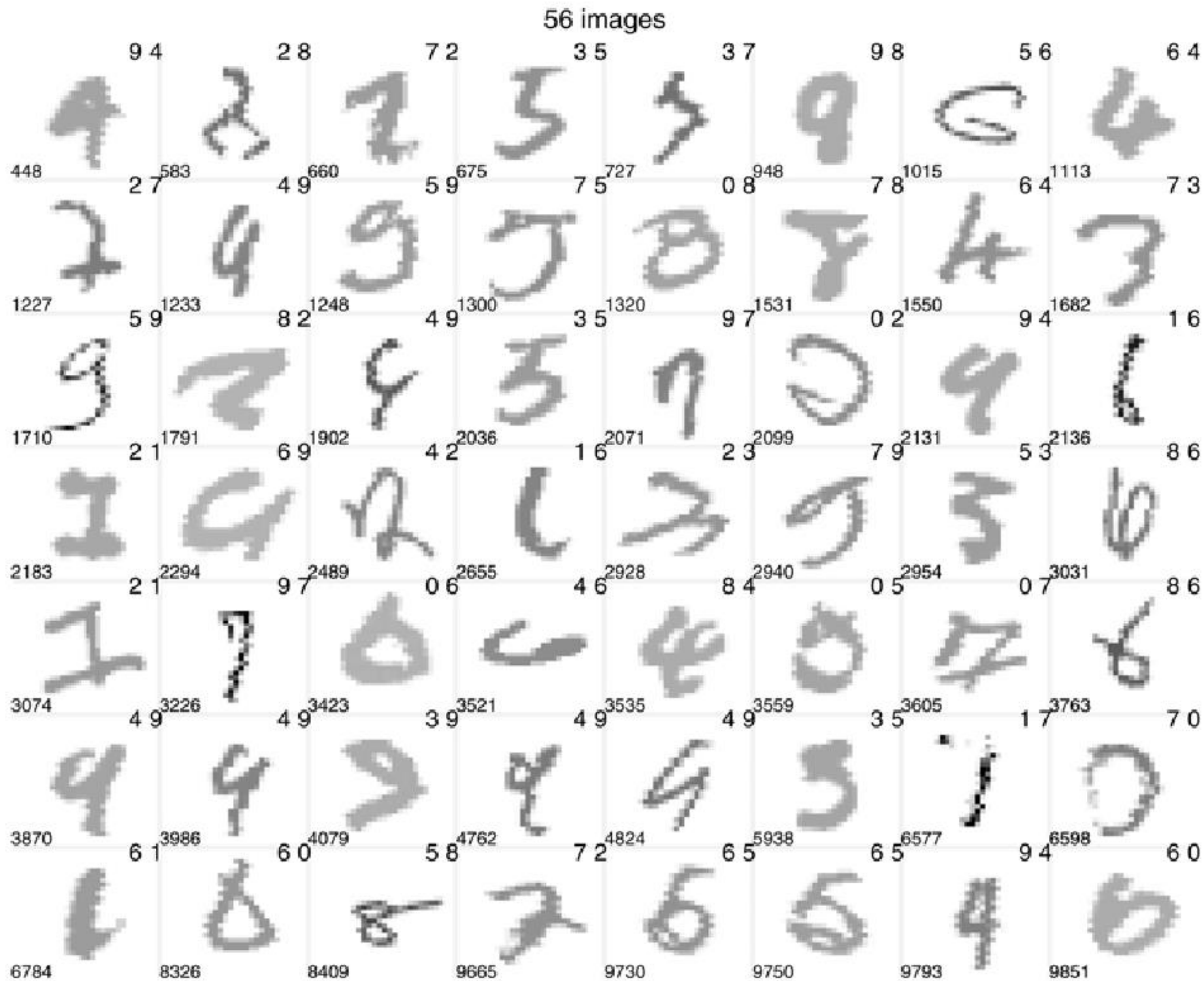
Table 7. Errors on deslanted MNIST (10,000 test examples), using VSV with 2-pixel translation.

	Misclassifications										
	Digit misclassifications										10-class
Digit	0	1	2	3	4	5	6	7	8	9	Test error rate
SV	5	5	14	12	13	10	13	13	12	25	1.22%
VSV	3	4	6	4	8	7	8	7	8	13	0.68%
VSV2	3	3	5	3	6	7	7	6	5	11	**0.56%**

Handwritten Digits

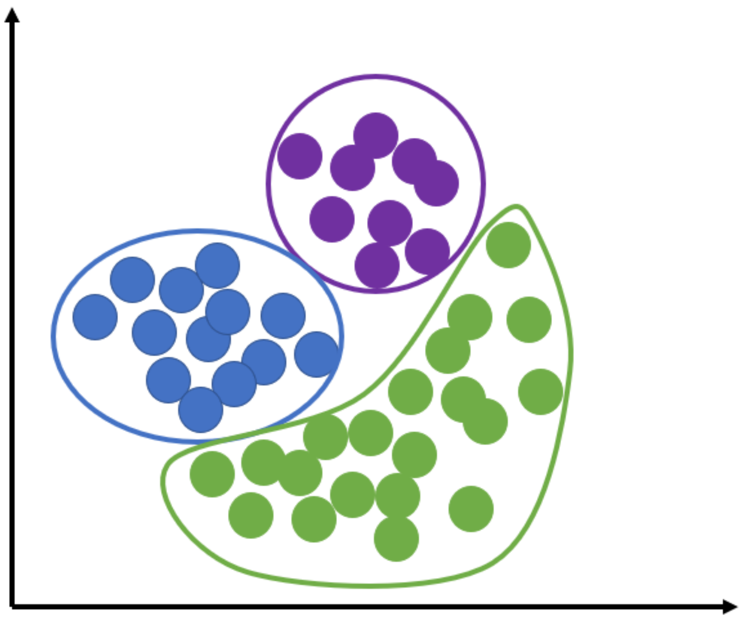
Detour

- Only 56 misclassifications in 10,000 test examples:



Feedback and Classification

1. Linear SVMs
2. Nonlinear SVMs
3. Support Vector Machines in IR
4. **Overfitting**

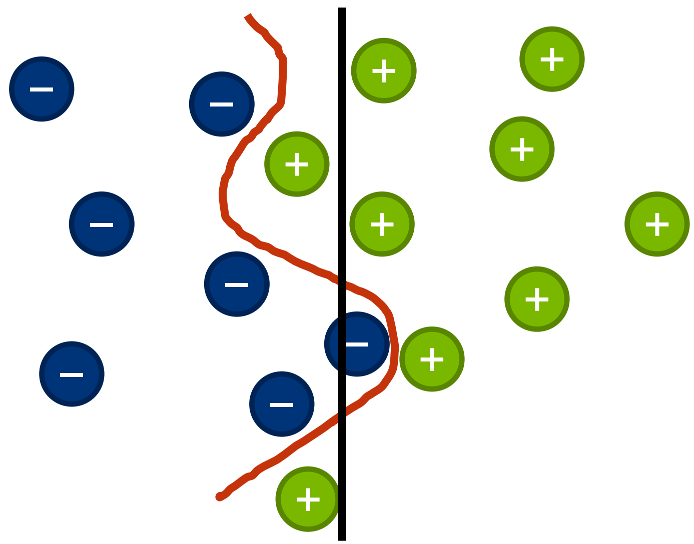


The Bias–Variance Tradeoff

- Usually, there is a tradeoff in choosing the "right" type of classifier
  - **Ignoring specific characteristics** of the training set leads to a **systematic bias** in classification
  - **Accounting for all individual properties** of the training set leads to a **large variance** over classifiers when the training set is randomly chosen from some large "true" data set
- Learning error = bias + variance
- So, what type of SVM is the "right" one?
- **Typically, you cannot have both!** (small bias & small variance)

High Variance learning methods

- If we use a mapping to a high-dimensional space that is "complicated enough," we could find a perfect linear separation in the transformed space, for any training set
- **Example:** How to separate this data set into two parts?



Overfitting

- **A perfect classification for the training set could generalize badly on new data**
- Fitting a classifier too strongly to the specific properties of the training set is called **overfitting**
- What can we do to avoid it?

1) **Cross-validation:**
   - **Randomly split the available data** into two parts (training set + test set)
   - Use the first part for **learning the classifier** and the second part for **checking the classifier's performance**
   - Choose a classifier that maximizes performance on the test set

Overfitting

2) Regularization:

- If you know how a "good" classifier roughly should look like (e.g. polynomial of low degree) you could introduce a penalty value into the optimization problem
- Assign a large penalty if the type of classifier is far away from what you expect, and a small penalty otherwise
- Choose the classifier that minimizes the overall optimization goal (original goal + penalty)
- An example of regularization is the **soft margin technique** since classifiers with large margins and few errors are preferred

Next Lecture

- Introduction to Web retrieval