TDT4173 Machine Learning

Lecture 3 – Bagging & Boosting + SVMs

Norwegian University of Science and Technology

Helge Langseth
IT-VEST 310
helgel@idi.ntnu.no

NTNU

Outline

1. Ensemble-methods
   - Background
   - Bagging
   - Boosting

2. Support Vector Machines
   - Background
   - Linear separators
   - The dual problem
   - Non-separable subspaces
   - Nonlinearity and kernels

Summary-points from last lesson

1. Decision trees
   - Representation, learning
   - Overfitting
   - Bias
2. Evaluating hypothesis
   - Sample error, true error
   - Estimators
   - Confidence intervals for observed hypothesis error
   - The central limit Theorem
   - Comparing hypothesis
   - Comparing learners
3. Computational Learning Theory
   - Bounding the true error
   - PAC learning

Ensemble methods: Motivation

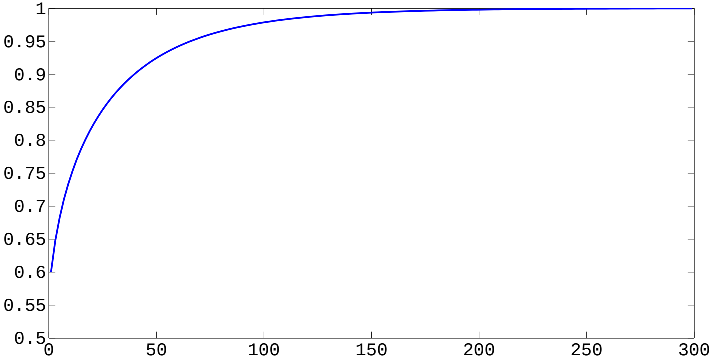
1. I ask someone a question where everyone has a $p = .6$ probability to get it right.
2. I ask $B \gg 1$ (**independent**) people the same question, and let them vote.

Which setup will give me the most reliable result?

Ensemble methods: Motivation

1. I ask someone a question where everyone has a $p = .6$ probability to get it right.
2. I ask $B \gg 1$ (**independent**) people the same question, and let them vote.

Which setup will give me the most reliable result?



Ensemble-methods

Fundamental setup

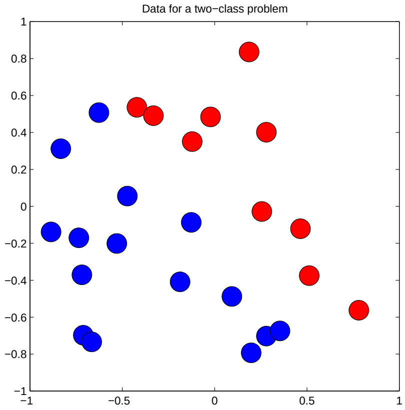
1. Generate a collection (ensemble) of classifiers from a given dataset. Each classifier can be "weak".
2. Each classifier is given a different dataset. The datasets are generated by resampling.
3. The final classification is defined by voting.

Nice property

- Each weak classifier in the ensemble may be fairly simple ("silly" and with a huge bias); we only require that it is able to do better than choosing blindly.
- Still, the ensemble of classifiers can be made quite clever!

Weak learner – example

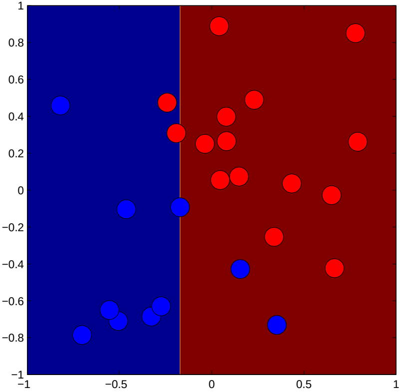
Consider a two-dimentional dataset of two classes.



What is a weak classifier in this case?

Weak learner – example

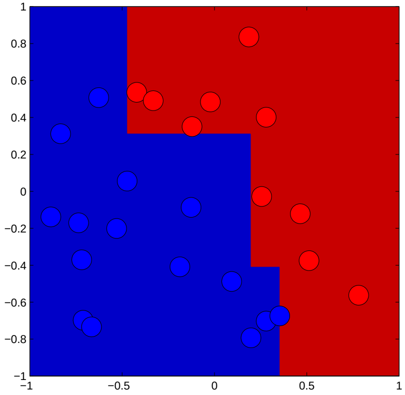
Consider a two-dimentional dataset of two classes.



One example: The half-space classifier

Weak learner – example

Consider a two-dimentional dataset of two classes.

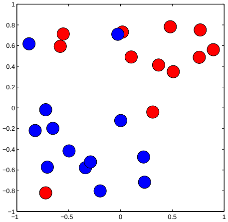


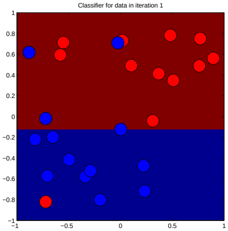
One example: The half-space classifier – Which aggregates well

Learning the half-space classifier (Naïve implementation)

Consider a dataset $\mathbf{D} = \{D_1, D_2, \ldots, D_m\}$, each D_j is an observation $((x_1, x_2), c)$. c is either `red` or `blue`.

- For d in each dimension:
  - For j over each observation:
    1. Let $b(d, j)$ be x_d from observation D_j.
    2. $r_l \leftarrow$ No. `red` observations with $x_d \leq b(d, j)$ $+$ No. `blue` observations with $x_d > b(d, j)$
    3. $\text{quality}(d, j) \leftarrow \max\{r_l, m - r_l\}$
- $(d_0, j_0) \leftarrow \text{argmax}_{d,j}\, \texttt{quality}(d, j)$
- Define the half-space classifier at $b(d_0, j_0)$





Two methods of ensemble learning

We will consider two methods of ensemble learning:

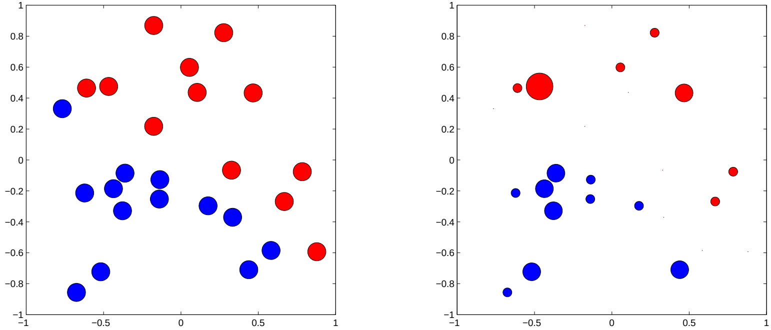
Bagging: Bagging (Bootstrap Aggregating) classifiers means to generate new datasets by drawing at random from original data (with replacement)

Boosting: Data for the classifiers are adaptively resampled (re-weighted), so that data-points that classifiers have had problems classifying are given more weight.

Bootstrapping

What is a bootstrap sample?

- Consider a data set $\mathbf{D} = \{D_1, D_2, \ldots, D_m\}$ with m datapoints
- A boostrap sample $\mathbf{D}^i$ can be created from $\mathbf{D}$ by choosing m points from $\mathbf{D}$ on random. The selection is done **with replacement**.



Bagging

- **Bagging** (= Bootstrap aggregating) produces **replications of the training set** by sampling with replacement.
- Each replication of the training set has the same size as the original set, but **some examples can appear more than once** while **others don't appear at all**.
- A classifier is generated from **each replication**.
- All classifiers are used to classify each sample from the test set using a **majority voting** scheme.

Bagging – more formally

1. Start with dataset $\mathbf{D}$
2. Generate B bootstrap samples $\mathbf{D}^1, \mathbf{D}^2, \ldots, \mathbf{D}^B$.
3. Learn weak classifier for each dataset: $c^i \leftarrow \mathcal{L}(\mathbf{D}^i)$; $\mathcal{L}(\cdot)$ is the weak learner; $c^i(\cdot) : \mathcal{X} \mapsto \{-1, +1\}$
4. The aggregated classifier of a new instance $\mathbf{x}$ simply found by taking a majority vote over classifiers:

$$c(\mathbf{x}) \leftarrow \operatorname{sign}\left\{\sum_{b=1}^{B} c^b(\mathbf{x})\right\}$$

DEMO

Bagging – Summary

When is it useful

- Bagging should only be used if the learner is **unstable**.
- A learner is said to be unstable if a small change in the training set yields large variations in the classification.
- Examples of unstable learners: neural networks, decision trees.
- Example of a stable learner: k-nearest neighbours, The half-space classifier.

Bagging properties

1. Improves the estimate if the learning algorithm is unstable.
2. Reduces the variance of predictions.
3. Degrades the estimate if the learning algorithm is stable.

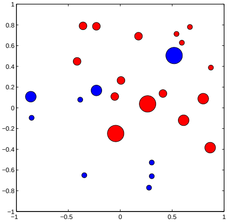
Boosting

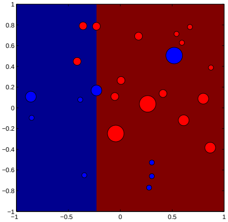
1. **Main idea as for bagging:**
   Invent new datasets, and learn separate classifiers for each dataset
2. **Clever trick:**
   The data is now **adaptively resampled**, so that previously misclassified examples count more in next dataset, previously well-classified observations are weighted less.
3. The classifiers are **aggregated** by adding them together, but **each classifier is weighted differently**

How to learn the WEIGHTED half-space classifier

Consider a dataset $\mathbf{D} = \{D_1, D_2, \ldots, D_m\}$, each D_j is an observation $((x_1, x_2), c)$. c is either red or blue.

- For d in each dimension:
  - For j over each observation:
    1. Let $b(d,j)$ be x_d from observation D_j.
    2. $r_l \leftarrow$ Sum of weight of red observations with $x_d \leq b(d,j)$ $+$ Sum of weight of blue observations with $x_d > b(d,j)$
    3. $\text{quality}(d,j) \leftarrow \max\left\{r_l, \sum_i w(i) - r_l\right\}$
- $(d_0, j_0) \leftarrow \text{argmax}_{d,j}\, \texttt{quality}(d,j)$
- Define the half-space classifier at $b(d_0, j_0)$





Boosting

- **Boosting** produces **replications of the training set** by re-weighting the importance of each observation in the original dataset.
- A classifier is generated from **each replication** using the **weighted** data-set as input.
- All classifiers are used to classify each sample from the test set. Each classifiers get a **weight** based on its calculated accuracy on the **weighted training set**.

Boosting – more formally

1. Start with dataset $\mathbf{D}$ and weights $\boldsymbol{w}_1 = [1/m, 1/m, \ldots, 1/m]$.
2. Repeat for $t = 1, \ldots, T$:
   1. c^t is the classifier learned from the **weighted** data $(\mathbf{D}, \boldsymbol{w}_t)$.
   2. ϵ_t is the **error** of c^t measured on the **weighted** data $(\mathbf{D}, \boldsymbol{w}_t)$.
   3. Calculate $\alpha_t \leftarrow \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
   4. Update weights (1):

$$w_{t+1}(i) \leftarrow w_t(i) \times \begin{cases} \exp(-\alpha_t) & \text{if } \mathbf{x}_i \text{ was classified correctly} \\ \exp(+\alpha_t) & \text{otherwise} \end{cases}$$

   5. Update weights (2): Normalize $\boldsymbol{w}_{t+1}$.
3. The aggregated classifier of a new instance $\mathbf{x}$ is found by taking a **weighted** vote over classifiers:

$$c(\mathbf{x}) \leftarrow \text{sign}\left\{\sum_{t=1}^{T} \alpha_t \cdot c^t(\mathbf{x})\right\}$$

DEMO

Boosting – Summary

Boosting properties

- The error rate of c on the un-weighted training instances **approaches zero exponentially quickly** as T increases
- Boosting forces the classifier to have at least 50% accuracy on the **re-weighted** training instances. This causes the learning system to produce a quite different classifier on the following trial (as next time, we focus on the mis-classifications of this round). This leads to an **extensive exploration of the classifier space**.
- Boosting is more clever than bagging, as it will never reduce classification performance.

Description of the task

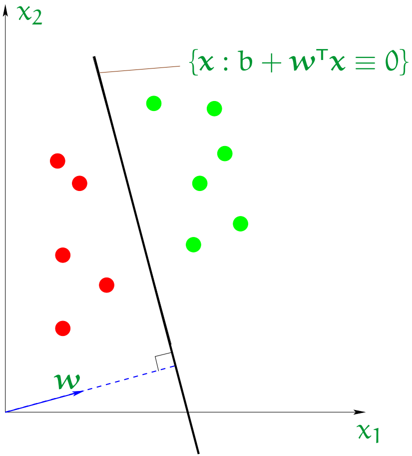
Data:

1. We have a set of data $\mathbf{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$. The instances are described by $\mathbf{x}_i$, the class is y_i.
2. The data is generated by some unknown probability distribution $P(\mathbf{x}, y)$.

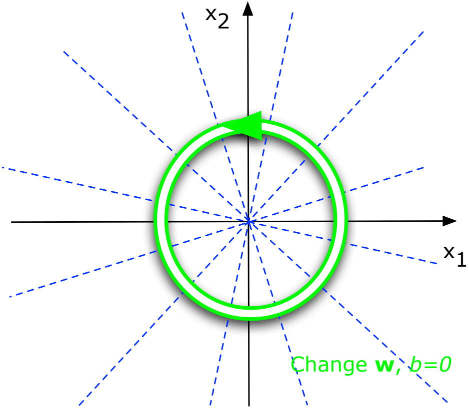
Task:

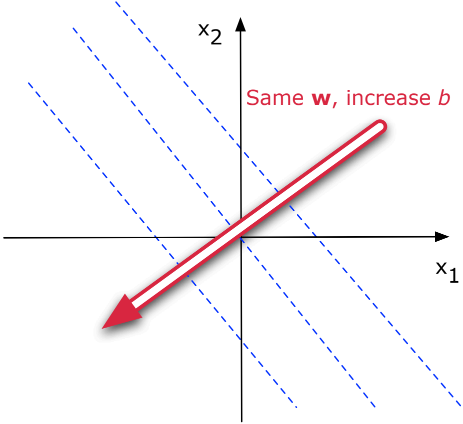
1. Be able to "guess" y at a new location $\mathbf{x}$.
2. For SVMs one typically states this as "find an unknown function $f(\mathbf{x})$ that estimates y at $\mathbf{x}$."
3. **Note!** In this lesson we look at **binary classification**, and let $y \in \{-1, +1\}$ denote the classes.
4. We will look for **linear functions**, i.e., $f(\mathbf{x}) = b + \mathbf{w}^{\mathsf{T}}\mathbf{x} \equiv b + \sum_{i=1}^{m} w_i \cdot x_i$

Describing the separating plane

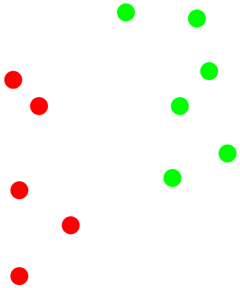


More on the representation



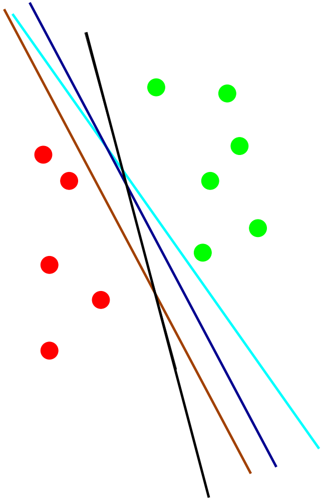


How to find the best linear separator



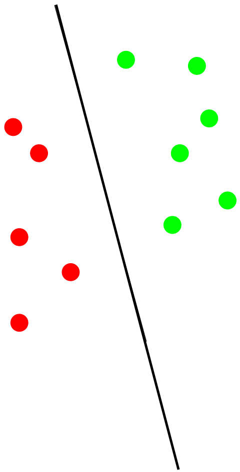
We are looking for a linear separator for this data

How to find the best linear separator



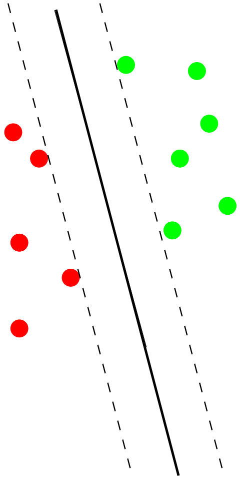
There are so many solutions. . .

How to find the best linear separator



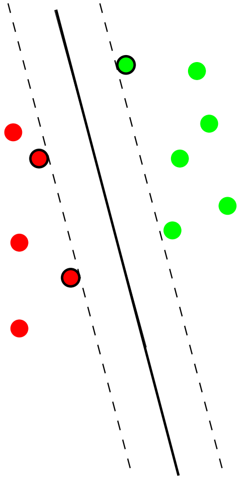
But only one is considered the “best”!

How to find the best linear separator



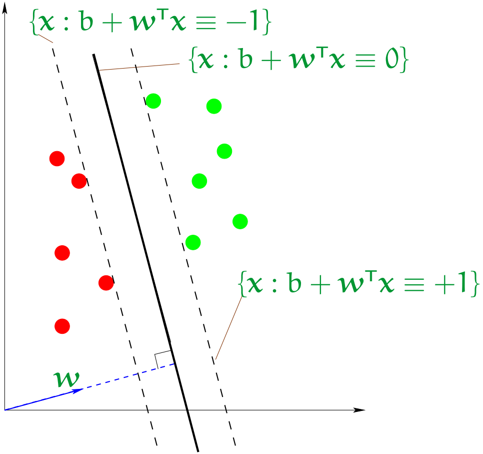
SVMs are called "large margin classifiers"

How to find the best linear separator



. . . and the data points touching the lines are the support vectors

The geometry of the problem



Note! Since one line has $b + w^\mathsf{T}x = -1$, the other has $b + w^\mathsf{T}x = 1$, the length between them is $2/\|w\|$.

An optimisation problem

Optimisation criteria:

- The distance between margins is $2/\|\mathbf{w}\|$, so that is what we want to maximise.
- Equivalently, we can minimise $\|\mathbf{w}\|/2$.
- For simplicity of the mathematics, we will rather minimise $\|\mathbf{w}\|^2/2$

Constraints:

- The margin separates all data observations correctly:
  - $b + \mathbf{w}^\intercal \mathbf{x}_i \leq -1$ for $y_i = -1$.
  - $b + \mathbf{w}^\intercal \mathbf{x}_i \geq +1$ for $y_i = +1$.
- Alternative (equivalent) constraint set: $y_i(b + \mathbf{w}^\intercal \mathbf{x}_i) \geq 1$

An optimisation problem (2)

Mathematical Programming Setting:

Combining the above requirements we obtain

$$\text{minimize wrt. } \boldsymbol{w} \text{ and } b\text{: } \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{subject to } y_i(b + \boldsymbol{w}^\top x_i) - 1 \geq 0, i = 1, \ldots, m$$

Properties:

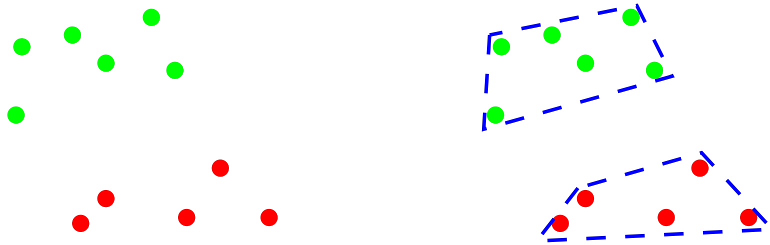
- Problem is convex
- Hence it has unique minimum
- Efficient algorithms for solving it exist

The dual problem – and the convex hull

The convex hull of $\{x_j\}$:
The smallest subset of the instance space that

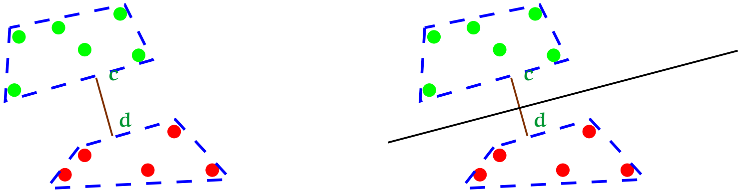
- is convex
- contains all elements $\{x_j\}$

is the convex hull of $\{x_j\}$. Find it by drawing lines between all x_j and choose the "outermost boundary".



The dual problem – and the convex hull (2)

Look at the difference between the points closest in the convex hulls. The decision line must be orthogonal to the line between the two closest points.



So, we want to minimise $\|\mathbf{c} - \mathbf{d}\|$. $\mathbf{c}$ can be written as a weighted sum of all elements in the green class: $\mathbf{c} = \sum_{y_i=+1} \alpha_i x_i$, and similarly for $\mathbf{d}$.

Note: The α_i-values are zero unless x_i is a support vector.

The dual problem – and the convex hull (3)

Minimising $\|\mathbf{c} - \mathbf{d}\|$ is (modulo a constant) equivalent to this formulation:

$$\text{minimize wrt. } \boldsymbol{\alpha}\text{: } \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \sum_{i=1}^{m} \alpha_i$$

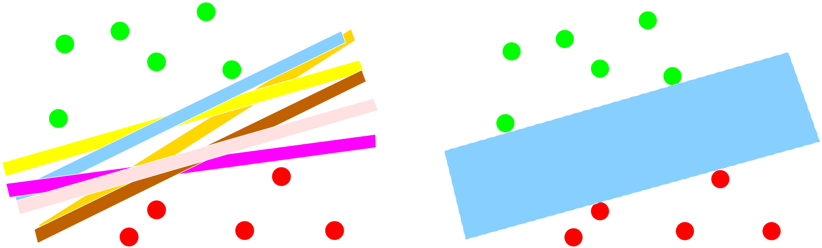
$$\text{subject to } \sum_{i=1}^{m} y_i \alpha_i = 0 \text{ and that } \alpha_i \geq 0, i = 1, \ldots, m$$

Properties:

- Problem is convex, hence has unique minimum.
- Quadratic programming problem – known solution method.
- For solution: $\alpha_i > 0$ only if $\mathbf{x}_i$ is a support vector.

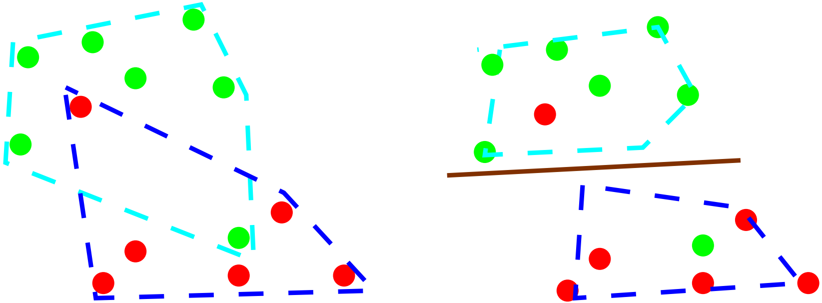
"Theoretical foundation"

1. Proofs of SVM properties available (but out of scope for us)
2. Large separators smart if we have small variations in x then we will still classify correctly
3. There are many "skinny" margin planes, only one if you look for the "fattest" plane; thus more robust.



What if the convex hulls are overlapping?

- If the convex hulls are overlapping we cannot find a linear separator
- To handle this, we optimise a criteria where we maximise distance between lines minus a penalty for mis-classifications
- This is equivalent to **scaling the convex hulls**, and do as before on the reduced convex hulls



What if the convex hulls are overlapping? (2)

The problem with scaling is (modulo a constant) equivalent to this formulation:

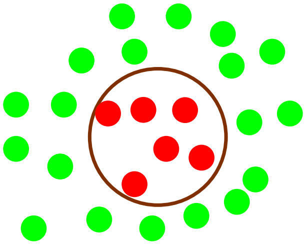
$$\text{minimize wrt. } \boldsymbol{\alpha}\text{: } \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j - \sum_{i=1}^{m}\alpha_i$$

$$\text{subject to } \sum_{i=1}^{m} y_i\alpha_i = 0 \text{ and that } 0 \le \alpha_i \le C, i = 1, \ldots, m$$

Properties:

- Problem as before, but C introduces the scaling; this is equivalent to incurring cost of misclassification.
- Still solvable using "standard" methods.
- Demo: Different values of C: `http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml`

Nonlinear problems – when scaling does not make sense



- The problem is difficult to solve when $\mathbf{x} = (r, s)$ has only two dimensions
- ... but if we blow it up to us five dimension: $\theta(\mathbf{x}) = \{r, s, rs, r^2, s^2\}$, i.e. "invent" the mapping $\theta(\cdot) : \mathbb{R}^2 \mapsto \mathbb{R}^5$, and try to find the linear separator in $\mathbb{R}^5$, then everything is OK.

Solving the problem in higher dimensions

We solve this as before, but remembering to look in the higher dimension:

$$\text{minimize wrt. } \boldsymbol{\alpha}\text{: } \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \theta(\mathbf{x}_i)^\top\theta(\mathbf{x}_j) - \sum_{i=1}^{m}\alpha_i$$

$$\text{subject to } \sum_{i=1}^{m} y_i\alpha_i = 0 \text{ and that } 0 \le \alpha_i \le C, i = 1, \ldots, m$$

Solving the problem in higher dimensions

We solve this as before, but remembering to look in the higher dimension:

$$\text{minimize wrt. } \boldsymbol{\alpha}\text{: } \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \theta(\mathbf{x}_i)^\top\theta(\mathbf{x}_j) - \sum_{i=1}^{m}\alpha_i$$

$$\text{subject to } \sum_{i=1}^{m} y_i\alpha_i = 0 \text{ and that } 0 \le \alpha_i \le C, i = 1, \ldots, m$$

Note that:

- We do not need to evaluate $\theta(\mathbf{x})$ directly, only $\theta(\mathbf{x}_i)^\top\theta(\mathbf{x}_j)$.
- If we find a "clever way" of evaluating $\theta(\mathbf{x}_i)^\top\theta(\mathbf{x}_j)$ (i.e., independent of the size of the target space) we can solve the problem easily, and without even thinking about what $\theta(\mathbf{x})$ even means.
- We define $K(\mathbf{x}_i, \mathbf{x}_j) = \theta(\mathbf{x}_i)^\top\theta(\mathbf{x}_j)$, and focus on finding $K(\cdot, \cdot)$ instead of the mapping. K is called a **kernel**.

Kernel functions

$\theta(\mathbf{x})$	$K(\theta(\mathbf{x}_i), \theta(\mathbf{x}_j))$
Degree d polynomial	$(\mathbf{x}_i^T \mathbf{x}_j + 1)^d$
Radial Basis Functions	$\exp\left(\frac{(\mathbf{x}_i - \mathbf{x}_j)^2}{2\sigma}\right)$
Two-layer Neural Network	$\text{sigmoid}\,(\eta \cdot \mathbf{x}_i^T \mathbf{x}_j + c)$

- Different kernels have different properties, and finding the "right" kernel is a difficult task, and can be hard to visualise.
- **Example:** The RBF kernel uses (implicitly) an infinitely dimensional representation for $\theta(\cdot)$.

SVMs: Algorithmic summary

- Select the parameter C (tradeoff between minimising training set error and maximising the margin).
- Select kernel function, and associated parameters (e.g., σ for RBF).
- Solve the optimisation problem using quadratic programming.
- Find the value b (the treshold) by using the support vectors.
- Classify a new point $\mathbf{x}$ using

$$f(\mathbf{x}) = \text{sign}\left\{\sum_{i=1}^{m} y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) - b\right\}$$

Demo: Different kernels
http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml