

Chapter 8: Neural networks and support vector machines

Volodymyr Vovk

`v.vovk@rhul.ac.uk`
Office Bedford 2-20

CS3920/CS5920 Machine Learning
Last edited: December 5, 2022

Version with solutions on slides 24, 39, 41, and 67

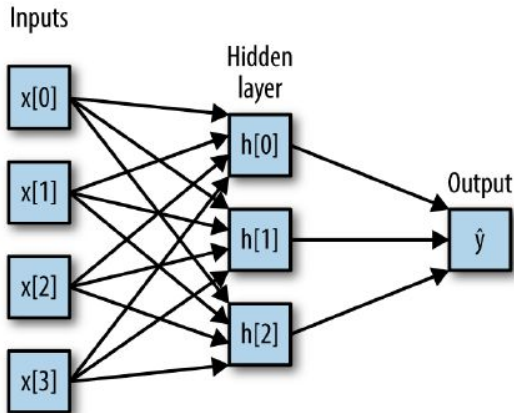
Neural nets and support vector machines

- Neural nets were used from the dawn of artificial intelligence. Motivation: this is how we think.
- SVMs were conceived by Vladimir Vapnik as competitor for neural nets (and in his book he explained that neural nets with one hidden layer can be considered to be SVMs). For a while they completely eclipsed neural nets.
- With the rise of deep learning, neural nets are in again.
- The cycle might continue. . . .

Plan

- 1 Neural networks
- 2 Maximum margin classifiers
- 3 Support vector machines
- 4 SVMs with more than two classes

Neural network with 1 hidden layer



Model

- Given the input features $x[0], \dots, x[p-1]$, the output label is computed from left to right.
- The full formula (in the case of regression) is:

$$h[0] := \tanh(w[0,0]x[0] + w[0,1]x[1] + w[0,2]x[2] + w[0,3]x[3] + b[0])$$

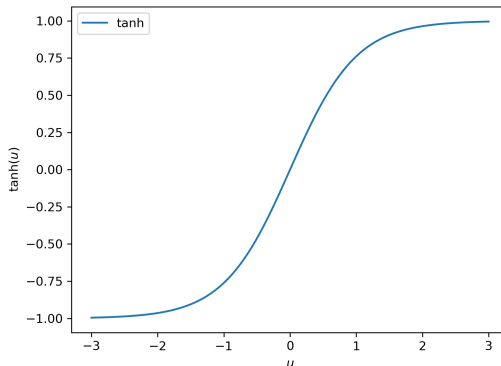
$$h[1] := \tanh(w[1,0]x[0] + w[1,1]x[1] + w[1,2]x[2] + w[1,3]x[3] + b[1])$$

$$h[2] := \tanh(w[2,0]x[0] + w[2,1]x[1] + w[2,2]x[2] + w[2,3]x[3] + b[2])$$

$$\hat{y} := v[0]h[0] + v[1]h[1] + v[2]h[2] + b.$$

Activation function `np.tanh`

Here `tanh` (**t**angens **h**yperbolicus, a function in NumPy) is the function nicely mapping the real line \mathbb{R} to $(-1, 1)$. We can replace it with a different **activation function**.



Plotting in Python (using matplotlib)

The graph on the previous slide was drawn as follows:

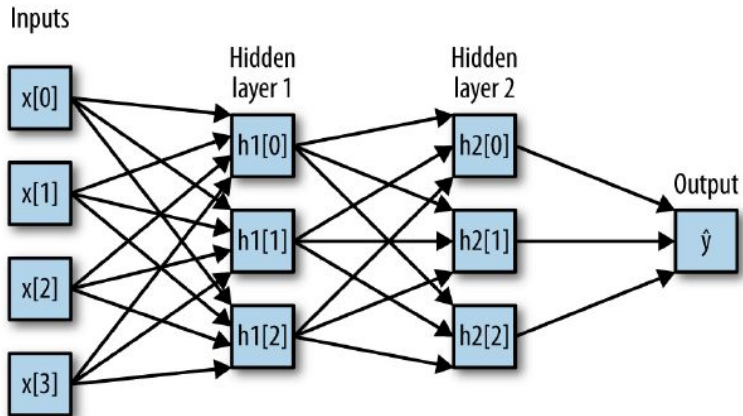
```
import numpy as np
%matplotlib notebook
import matplotlib.pyplot as plt
line = np.linspace(-3, 3, 100)
plt.plot(line, np.tanh(line), label="tanh")
plt.legend(loc="best")
plt.xlabel("u")
plt.ylabel("tanh(u) ")
```

Here the “Python magic” `%matplotlib notebook` allows you to adjust your picture.

Parameters

- The parameters are w (the weights between the inputs x and the hidden layer h), v (the weights between the hidden layer h and the output \hat{y}), and the bs .
- The parameters v , w , and b are learned from data.
- An important parameter that needs to be set by the user is the number of nodes in the hidden layer.
- It is also possible to add additional hidden layers, as shown on the next slide.

Neural network with 2 hidden layers



Strengths and weaknesses (1)

- + Neural networks are able to capture information contained in large amounts of data and build incredibly complex models. Given enough computation time, data, and careful tuning of the parameters, they often beat other machine learning algorithms (for both classification and regression).
- Neural networks—particularly the large and powerful ones—often take a long time to train.

Strengths and weaknesses (2)

- They require careful preprocessing of the data (normalization is essential).
- They work best with “homogeneous” data, where all the features have similar meanings.
- Training neural networks is a very difficult art.

Plan

- 1 Neural networks
- 2 Maximum margin classifiers
- 3 Support vector machines
- 4 SVMs with more than two classes

Introduction

- One reason why neural networks are so difficult to train is that their performance (measured, e.g., by the RSS in the case of regression) is a complicated function of the parameters, with lots of local minima (and it's very easy to get stuck in one of those).
- Vapnik considered SVM an improved version of neural networks (with one hidden layer) that does not have multiple local minima.
- The reason it does not have multiple local minima is that the function it is trying to minimize is convex (details: later).

What is a hyperplane? (1)

In the p -dimensional space \mathbb{R}^p , a **hyperplane** is a flat affine (i.e., not necessarily containing 0) subspace of dimension $p - 1$.

- In two dimensions: a line.
- In three dimensions: a plane.
- In $p > 3$ dimensions, it can be hard to visualize a hyperplane.

What is a hyperplane? (2)

- In two dimensions, a hyperplane is defined by

$$w[0]x[0] + w[1]x[1] + b = 0$$

for parameters $w[0]$, $w[1]$, and b . This means: any x for which the equation holds is a point on the hyperplane.

- This can be easily extended to the p -dimensional setting:

$$w[0]x[0] + w[1]x[1] + \cdots + w[p-1]x[p-1] + b = 0$$

defines a p -dimensional hyperplane. We will write the last equation as

$$w \cdot x + b = 0$$

and always assume $w \neq 0$.

Splitting the space

Now, suppose that x does not satisfy this equation.

- If

$$w \cdot x + b > 0,$$

then this tells us that x lies to one side of the hyperplane.

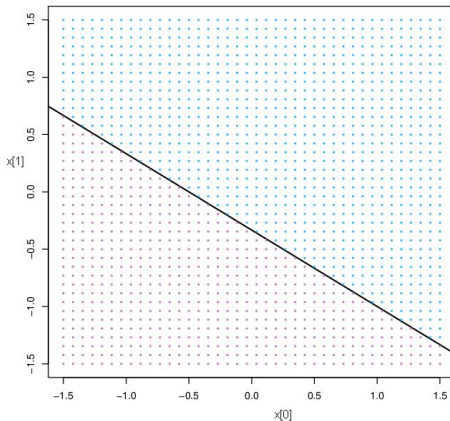
- On the other hand, if

$$w \cdot x + b < 0,$$

then x lies to the other side of the hyperplane.

We can think of the hyperplane as dividing the p -dimensional space into two halves. One can determine on which side of the hyperplane a point lies by calculating the sign of $w \cdot x + b$.

A hyperplane in two-dimensional space



Classification using a separating hyperplane (1)

- Suppose we have a training set that consists of n training samples x_1, \dots, x_n in p -dimensional space, and these samples fall into two classes:

$$y_1, \dots, y_n \in \{-1, 1\},$$

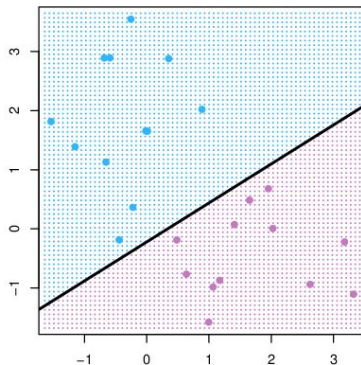
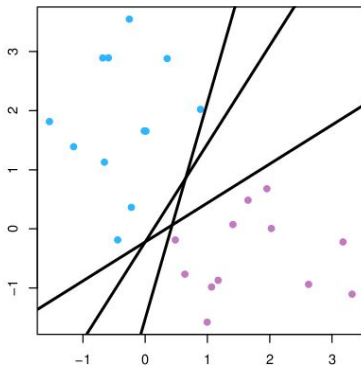
where -1 represents one class and 1 the other class.

- Suppose we also have a test sample x^* .
- Our goal: develop a classifier based on the training data that will correctly classify the test sample using its features.

Classification using a separating hyperplane (2)

- Suppose it is possible to construct a hyperplane that separates the training samples perfectly according to their class labels.
- Examples of three such separating hyperplanes are on the left of the following figure.
- There are two classes of observations, shown in blue and in purple, each of which has two features.

Separating hyperplanes



Mathematics of separating by a hyperplane

Let us label the observations from the blue class as $y_i = 1$ and those from the purple class as $y_i = -1$. Then a separating hyperplane has the property that

$$\begin{cases} w \cdot x_i + b > 0 & \text{if } y_i = 1 \\ w \cdot x_i + b < 0 & \text{if } y_i = -1. \end{cases}$$

Equivalently, a separating hyperplane has the property that

$$y_i(w \cdot x_i + b) > 0$$

for all $i = 1, \dots, n$.

Classifier from a separating hyperplane

If a separating hyperplane exists, we can use it as a classifier: a test sample is assigned a class depending on which side of the hyperplane it is located. Shown on the right of the previous figure. We classify the test sample x^* based on the sign of the **scoring function** (also called **decision function** in `scikit-learn`)

$$f(x^*) = w \cdot x^* + b.$$

- If $f(x^*)$ is positive, then we assign the test sample to class 1.
- If $f(x^*)$ is negative, then we assign it to class -1 .

Exercise

- Suppose we have a linear scoring function with parameters $b = 1$ and $w = (1, 0, -3)$ (in the notation of the previous slide).
- The test sample is $x^* = (-1, 0, 1)$.
- Calculate the predicted label for x^* .

Solution

- *The value of the scoring function is*

$$w \cdot x^* + b = (1, 0, -3) \cdot (-1, 0, 1) + 1 = -3.$$

- *Therefore, the prediction is -1 .*

Confidence

Moreover, we can also make use of the magnitude of $f(x^*)$.

- If $f(x^*)$ is far from zero, then this means that x^* lies far from the hyperplane, and so we can be confident about our class assignment for x^* .
- On the other hand, if $f(x^*)$ is close to zero, then x^* is located near the hyperplane, and so we are less certain about the class assignment for x^* .

The maximum margin classifier

- Notice: if our data can be perfectly separated using a hyperplane, then there will in fact exist an infinite number of such hyperplanes.
- In order to construct a classifier based upon a separating hyperplane, we must have a reasonable way to decide which of the infinitely many possible separating hyperplanes to use.
- A natural choice is the **maximum margin hyperplane** (also known as the **optimal separating hyperplane**), which is the separating hyperplane that is farthest from the training samples.

The margin (1)

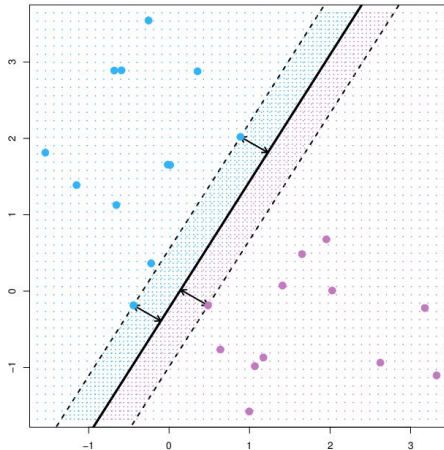
- We can compute the (perpendicular) distance from each training sample to a given separating hyperplane; the smallest such distance is known as the **margin**.
- The maximum margin hyperplane is the separating hyperplane for which the margin is largest.
- We can then classify a test sample based on which side of the maximum margin hyperplane it lies.
- This is known as the **maximum margin classifier**.

The margin (2)

- We hope that a classifier that has a large margin on the training data will also have a large margin on the test data, and hence will classify the test samples correctly.
- Although the maximum margin classifier is often successful, it can also lead to overfitting when p is large.
- If $w = (w[0], \dots, w[p-1])$ and b are the parameters of the maximum margin hyperplane, then the maximum margin classifier classifies the test sample x^* based on the sign of

$$f(x^*) = w \cdot x^* + b.$$

The maximum margin hyperplane on the data set of the previous figure



Geometry of the maximum margin hyperplane

- The maximum margin hyperplane represents the mid-line of the widest “slab” that we can insert between the two classes.
- In the previous figure, three training samples are equidistant from the maximum margin hyperplane and lie along the dashed lines indicating the width of the margin.
- These three samples are known as **support vectors**.

Support vectors

- They are called “support vectors” since they are vectors in the p -dimensional space \mathbb{R}^p ($p = 2$ in the figure) and they “support” the maximum margin hyperplane: if these points were moved slightly then the maximum margin hyperplane would move as well.
- The maximum margin hyperplane depends directly on the support vectors, but not on the other samples: a movement to any of the other samples would not affect the separating hyperplane, provided that the sample’s movement does not cause it to cross the boundary set by the margin.

Construction of the maximum margin classifier (1)

- How do we construct the maximum margin hyperplane based on a set of n training samples $x_1, \dots, x_n \in \mathbb{R}^P$ and associated class labels $y_1, \dots, y_n \in \{-1, 1\}$?
- The maximum margin hyperplane (determined by w and b) is the solution to the optimization problem

$$\|w\|^2 \rightarrow \min$$

subject to

$$y_i (w \cdot x_i + b) \geq 1, \quad i = 1, \dots, n.$$

Construction of the maximum margin classifier (2)

- The constraint

$$y_i (w \cdot x_i + b) \geq 1$$

guarantees that each sample will be on the correct side of the hyperplane with some cushion.

- The objective

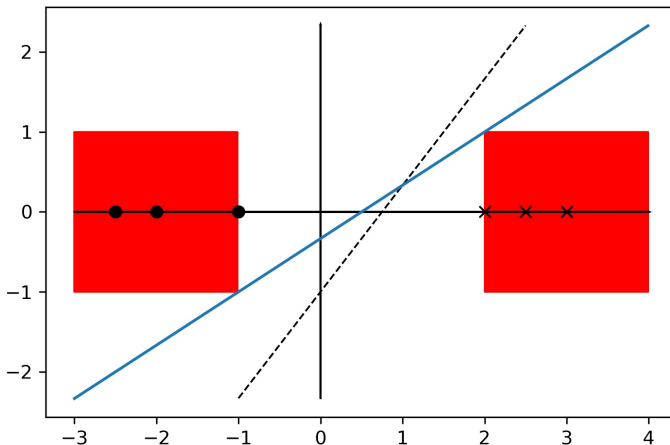
$$\|w\|^2 \rightarrow \min$$

means that we are minimizing the Euclidean length of w .

Construction of the maximum margin classifier (3)

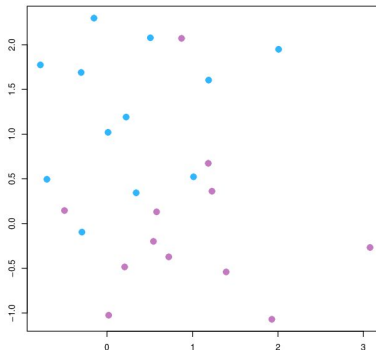
- In fact the objective $\|w\|^2 \rightarrow \min$ means that we are maximizing the margin.
- Why? Think of the 1D case, where we have only one feature, as on the next slide (where the negative and positive samples are shown as noughts and crosses).
- The optimization problem can be solved efficiently (it's nice and convex), but the details of this are outside the scope of this course.

Optimal w in 1D (slope of the blue line)



The non-separable case (1)

An example where no separating hyperplane exists (our optimization problem has no solution):



The non-separable case (2)

- We will extend the concept of a separating hyperplane in order to develop a hyperplane that **almost** separates the classes, using a so-called “soft margin”.
- The extension is the “soft margin classifier”.
- But first a few exercises.

Exercises (1)

The geometry behind the maximum margin classifier is quite intuitive.

- Suppose the training set is $\{((-1, 0), -1), ((1, 0), 1)\}$ (as always, it would be better to talk about a training sequence or a training bag). What is the optimal separating hyperplane? (Draw it and write an equation for it.)
- Suppose the training samples are:
 - positive: $(0, 1)$ and $(1, -1)$
 - negative: $(1, 0)$

What is the optimal separating hyperplane?

Solutions

Let's denote the axes as $x[0]$ and $x[1]$.

- *For the training set $\{((-1, 0), -1), ((1, 0), 1)\}$, the optimal separating hyperplane is $x[0] = 0$.*
- *For the other training set, the optimal separating hyperplane is $x[1] = -2x[0] + 1.5$.*
 - *The widest slab that can be inserted between the two classes lies between the straight line connecting the two positive samples and the parallel line passing through the negative sample.*
 - *The optimal separating hyperplane is the mid-line of this slab.*

Exercises (2)

- Add another negative observation to the training set, so that the training samples are:
 - positive: $(0, 1)$ and $(1, -1)$
 - negative: $(1, 0)$ and $(0, 0)$

What is the optimal separating hyperplane now?

Identify the support vectors in all these exercises.

Solutions

- *After adding that new sample, the training set ceases to be linearly separable. Therefore, there is no solution (and no support vectors).*
- *For the two training sets on slide 38, all training samples are support vectors.*

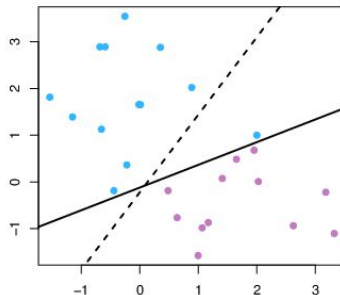
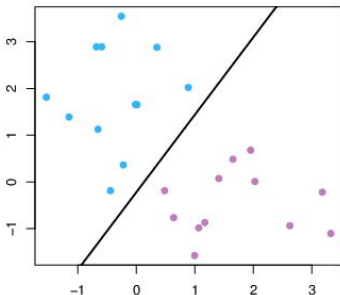
Plan

- 1 Neural networks
- 2 Maximum margin classifiers
- 3 **Support vector machines**
- 4 SVMs with more than two classes

Introduction

- Samples that belong to two classes are not necessarily separable by a hyperplane.
- And even if a separating hyperplane does exist, a classifier based on a separating hyperplane might not be desirable!
- A classifier based on a separating hyperplane will necessarily perfectly classify all of the training samples, which can lead to excessive sensitivity to individual observations.
- Example: next figure.

A dramatic change in the maximal separating hyperplane



A problem with the maximum margin classifier

- The addition of a single observation in the right-hand panel leads to a dramatic change in the maximum margin hyperplane.
- The resulting maximum margin hyperplane has only a tiny margin and so is not satisfactory.
 - As discussed earlier, the distance of a sample from the hyperplane measures our confidence that the sample was correctly classified.
 - The fact that the maximum margin hyperplane is extremely sensitive to a change in a single observation suggests overfitting the training data.

Soft margin classifier: idea (1)

Therefore, we consider a classifier based on a hyperplane that does not perfectly separate the two classes, in the interest of:

- greater robustness to individual observations, and
- better classification of **most** of the training samples.

It could be worthwhile to misclassify a few training samples in order to do a better job in classifying the remaining samples.

Soft margin classifier: idea (2)

- It allows some training samples to be on the wrong side of the margin, or even the wrong side of the hyperplane.
- The margin is **soft** because it can be violated by some of the training observations.
- Samples on the wrong side of the hyperplane are inevitable when there is no separating hyperplane.

Details of the soft margin classifier (1)

- The soft margin classifies a test sample depending on which side of a hyperplane it lies.
- The hyperplane is chosen to correctly separate most of the training samples into the two classes, but may misclassify a few samples.

Details of the soft margin classifier (2)

It is the solution to the optimization problem

$$\|w\|^2 + C \sum_{i=1}^n \zeta_i \rightarrow \min$$

subject to:

$$\begin{aligned} y_i (w \cdot x_i + b) &\geq 1 - \zeta_i, \\ \zeta_i &\geq 0, \quad i = 1, \dots, n, \end{aligned}$$

where C is a positive parameter (and the variables are $b, w[0], \dots, w[p-1], \zeta_1, \dots, \zeta_n$).

Details of the soft margin classifier (3)

- ζ_1, \dots, ζ_n are **slack variables**; they allow individual training samples to be on the wrong side of the margin or even hyperplane.
- Once we have solved the optimization problem, we classify a test sample x^* as before, by simply determining on which side of the hyperplane it lies, i.e., based on the sign of

$$f(x^*) = w \cdot x^* + b.$$

The tuning parameter C

- We have two conflicting goals: to achieve a wide margin and to classify our training samples well.
- C reflects the importance we attach to the second goal, and so it determines the number and severity of the violations to the margin (and to the hyperplane) that we will tolerate.
- If $C = \infty$ we do not tolerate any violations to the margin, and we must have $\zeta_1 = \dots = \zeta_n = 0$.
 - Our optimization problem now reduces to the old optimal separating hyperplane optimization problem.
 - An optimal separating hyperplane exists only if the two classes are separable.

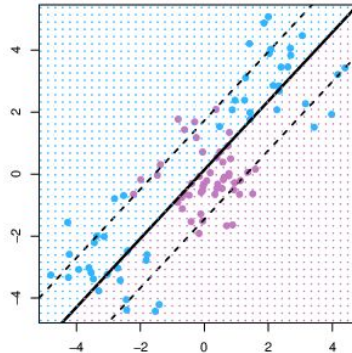
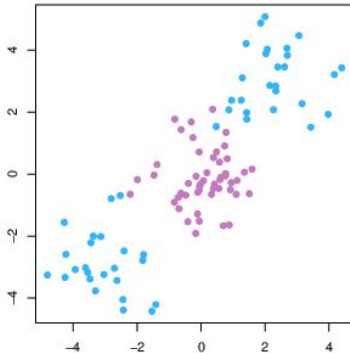
Support vectors

- It turns out that only samples that either lie on the margin or violate the margin will affect the hyperplane.
- A sample that lies strictly on the correct side of the margin does not affect the soft margin classifier.
- Changing the position of that sample would not change the classifier at all, provided that it remains on the correct side of the margin.
- Samples that lie directly on the margin, or on the wrong side of the margin for their class, are known as **support vectors**.
- Support vectors do affect the soft margin classifier.

Classification with non-linear prediction boundaries

- The soft margin classifier works well if the boundary between the two classes is linear.
- In practice we are often faced with non-linear class boundaries.
- See the next slide (the soft margin classifier shown in the right-hand panel is useless).

A non-linear boundary



Support vector machine (1)

- The crucial fact in the development of the SVM is that the soft margin optimization problem can be rewritten in such a way that it involves only the dot products of the training samples (as opposed to the samples themselves).
- To apply this solution to a test sample, we only need to know the dot products of the training samples and the test sample.
- Therefore, we can apply the kernel trick.
- This way we obtain a **support vector machine** (or SVM).

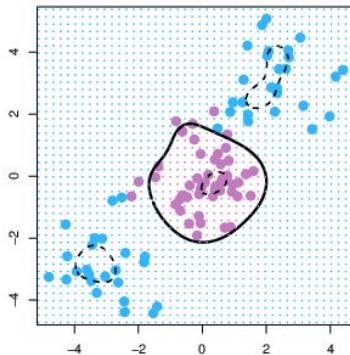
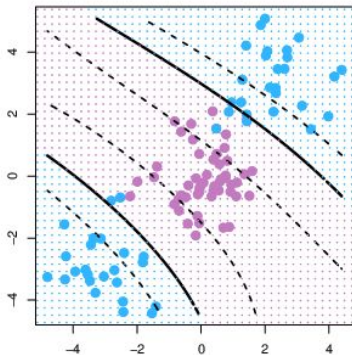
Support vector machine (2)

- You know **exactly** what is going on in the feature space: the soft margin classifier.
- In particular, there is a subset of the training set, the **support vectors**, such that the prediction for any test sample x^* depends only on the support vectors.
- Taking the linear kernel just gives us back the soft margin classifier.
- But we can use any kernel (polynomial, radial, etc.).

Support vector machine (3)

- Once again: when the soft margin classifier is combined with a kernel (especially non-linear, such as polynomial), the resulting classifier is known as a **support vector machine**.
- Example: next slide.

The non-linear boundary and non-linear kernels



An example (previous slide)

- The left-hand panel shows an example of an SVM with a polynomial kernel (namely, $d = 3$) applied to the non-linear data from the previous figure (on slide 54).
- The fit is a substantial improvement over the linear soft margin classifier.
- When $d = 1$, then the SVM essentially reduces to the soft margin classifier.
- The right-hand panel shows an example of an SVM with a radial kernel on our non-linear data; it also does a good job in separating the two classes.

Some uses of SVMs

- Health forecasting.
- Estimating risk in financial markets.
- Predicting attendance of marketing events.
- Calculating weather risks.
- Forecasting advertising revenues.

Strengths and weaknesses of support vector machines (1)

- + Support vector machines are powerful models and perform well on a variety of datasets. They allow for complex decision boundaries, even if the data has only a few features. They work well on low-dimensional and high-dimensional data (i.e., few and many features).
- They don't scale very well with the number of samples. Running an SVM on data with up to 10,000 samples might work well, but working with datasets of size 100,000 or more can become challenging in terms of runtime and memory usage.

Strengths and weaknesses of support vector machines (2)

- They require careful preprocessing of the data and tuning of the parameters.
- SVM models are hard to inspect; it can be difficult to understand why a particular prediction was made, and it might be tricky to explain the model to a nonexpert (or even an expert).
- Like neural nets, they work best with “homogeneous” data.

Plan

- 1 Neural networks
- 2 Maximum margin classifiers
- 3 Support vector machines
- 4 SVMs with more than two classes

SVMs with more than two classes

- So far, our discussion has been limited to the case of binary classification.
- It turns out that the concept of separating hyperplanes upon which SVMs are based does not lend itself naturally to more than two classes.
- Though a number of proposals for extending SVMs to the K -class case have been made, the two most popular are the one-vs-one and one-vs-rest approaches.
- The methods of this section are applicable to any (or almost any) method of binary classification.

One-vs-one classification

- Suppose that we would like to perform classification using SVMs, and there are $K > 2$ classes.
- A **one-vs-one** or **all-pairs** approach constructs $K(K - 1)/2$ SVMs, each of which compares a pair of classes.
- For example, one such SVM might compare the k th class, coded as $+1$, to the k' th class, coded as -1 .
- We classify a test sample using each of the $K(K - 1)/2$ classifiers, and we tally the number of times that the test sample is assigned to each of the K classes.
- The final classification is performed by assigning the test sample to the class to which it was most frequently assigned in these $K(K - 1)/2$ pairwise classifications.

Exercise

- Suppose the results of comparisons when running the one-vs-one procedure in a multi-class problem with 4 classes (A, B, C, and D) is:

	A	B	C	D
A		A	C	D
B			C	D
C				C
D				

For example, the A in row A and column B means that the SVM pitting A vs B predicts A.

- What is the overall prediction of the one-vs-one procedure for this test sample?

Solution

- Let us complete the table and tally the wins:

	A	B	C	D	tally
A		A	C	D	1
B	A		C	D	0
C	C	C		C	3
D	D	D	C		2

C is in majority, and so the overall prediction of the one-vs-one procedure is C.

- Equivalently, we can just count the overall number of different letters inside the original table on the previous slide.




One-vs-rest classification

- This is an alternative procedure for applying SVMs in the case of $K > 2$ classes.
- We fit K SVMs, each time comparing one of the K classes to the remaining $K - 1$ classes.
- Let f_k denote the scoring function that results from fitting an SVM comparing class k (coded as $+1$) to the others (coded as -1).
- Let x^* denote a test sample.
- In **one-vs-rest**, we assign the sample to the class for which $f_k(x^*)$ is largest (as this amounts to the highest level of confidence that the test sample belongs to class k).



Using one-vs-rest in conformal prediction

- We can use the idea of one-vs-rest to define an inductive conformity measure $A(\zeta, (x, y))$.
- Remember: $A(\zeta, (x, y))$ measures how well the labelled sample (x, y) conforms to ζ .
- Train K SVMs on ζ getting K scoring functions f_k (although we will use only one of them).
- Set $A(\zeta, (x, y)) = f_y(x)$.
- You will be asked to implement this inductive conformity measure in Assignment 3; `scikit-learn` has a method, `decision_function`, for computing it (Lab Worksheet 9, Section 3).

Further details (1)

-  M17, Chapter 2 (details of `scikit-learn` implementations).
-  H09: a lot of theory. Chapter 11: neural networks; Chapter 12: support vector machines.
-  John Shawe-Taylor and Nello Cristianini.
Kernel methods for pattern analysis.
Cambridge University Press, 2004.
See, especially, Chapters 2 and 3 (more advanced than this course).

Further details (2)

-  J21, Chapter 9: support vector machines (with a non-standard and unusually awkward statement of the optimization problem).
-  Vladimir Vapnik.
Statistical learning theory.
Wiley, 1998.
Fundamental book introducing support vector machines.