Machine Learning 1

CS4220 laboratory course manual

Version 0.9, 2020

David M.J. Tax, Marco Loog, Jesse Krijthe

Contents

Introduction **5**

1 Basics of Machine Learning **7**
- 1.1 Decision Theory 7
- 1.2 Prtools for Python 9
- 1.3 Datasets 9
- 1.4 Creating mappings and decision boundaries 11
- 1.5 Classification with Normal Densities 13
- 1.6 Density estimation using Parzen densities 14
- 1.7 The scaling problem 16

2 Linear Regression and Linear Classifiers **19**
- 2.1 Linear Least Squares without and with Intercept 19
- 2.2 Polynomial Regression and Other Feature Transformations 21
- 2.3 Fisher's Linear Discriminant 22
- 2.4 A Probabilistic Regression Model 23
- 2.5 Error Rates and Linear Classifiers 24
- 2.6 Linear Bayes Classifier 25
- 2.7 The Perceptron 25

3 Losses, Regularization, Evaluation **27**
- 3.1 Regularization 27
- 3.2 A Pseudoinverse 29
- 3.3 The Lasso 29
- 3.4 Hypothesis Classes and (Surrogate) Losses 30
- 3.5 Regularization and MAP 31
- 3.6 Probabilistic Model for Regression 31
- 3.7 Sources of Variation 32
- 3.8 Learning Curves 32
- 3.9 Cross-Validation 33
- 3.10 The Confusion Matrix 34
- 3.11 On to the Next Week. . . 35

4 Complexity **37**
- 4.1 The Support Vector Machine, `svc` 37
- 4.2 The Nonlinear Support Vector Machine, `svc` 38
- 4.3 Bias and Variance in Regression 39

4.4 Bias and Variance in Classification 40

5 Probabilistic Models & Clustering **41**
5.1 Bayes Rule Recap 41
5.2 Maximum Likelihood and A Posteriori Estimation 42
5.3 Missing Data 42
5.4 Some MAP Examples 43
5.5 That Bias and Variance Again 44
5.6 Regularization and MAP 45
5.7 The Predictive Distribution 45
5.8 Some Bayesian Network Basics 46
5.9 Inference in Bayesian Networks 47
5.10 Bayesian Networks and Learners 48
5.11 Clustering 50
5.12 Hierarchical clustering 50
5.13 Clustering with a mixture-of-Gaussians 51
5.14 Cluster validation 52

6 Reducing and Combining **55**
6.1 Some Combinatorics 55
6.2 Stuff on Scatter Matrices 56
6.3 Supervised Feature Extraction: the Fisher Mapping 56
6.4 PCA and Fisher 57
6.5 Feature Selection 60
6.6 Selection Experiments 62
6.7 Extraction and Selection 62
6.8 Combining 66
6.9 An AdaBoosting Exercise 68

Introduction

Course contents and goals

This course will provide you with a practical introduction to pattern recognition and machine learning. The techniques are discussed at a level such that you will be able to apply them in your research. The emphasis is on using the computer as a tool for pattern recognition. Starting from the basis for any pattern recognition application, measurements, the topics discussed will be:

- classification;
- evaluation;
- complexity;
- regression;
- feature selection and extraction;
- clustering.

After you have successfully completed this course, you should:

- understand pattern recognition theory to such an extent that he/she is able to read recent literature on the topic in engineering-oriented journals (e.g. IEEE Tr. on PAMI);
- know which statistical methods to apply to which problems, on which assumptions they are based and how these methods interrelate;
- be able to construct a learning system to solve a given simple problem, using existing software.

Prior knowledge

Basic working knowledge of multivariate statistics and linear algebra is required to follow the course. Next to that, it is expected that you have had the course CSE2510 Machine Learning, or something comparable.

Software Originally, this course use MATLAB to do the programming exercises. After many requests, the change to Python is made. Although both programming languages are very similar, the devil is in the details. It may be, that in some locations, still MATLAB notation is used. Please let us know when you find something!

You get get PRTOOLS for Python from `https://github.com/DMJTax/prtools`.

Notation

Most weeks contain a few optional exercises, indicated like this:

OPTIONAL

An exercise between these lines is optional. This means that you are not required to do it. It can give you some extra background and tips. Only work on it if you think the subject is interesting and if you have sufficient time left.

END OPTIONAL

Some other notational conventions are:

- Variables and code will be indicated using the `teletype font` (for example: `x`, `mean(x)`). For larger pieces of code, we will use the notation:

  ```
  >>> % A piece of test code
  >>> import numpy as np
  >>> x = np.random.rand(10,2);
  >>> np.mean(x)
  >>> np.std(x)
  ```

  Here, `>>>` is the prompt. If pieces of code are not preceded by `>>>` it will mean it's wiser to write a script.

- ☞ An alert sign like the one in the margin here indicates it is essential you read the text next to it carefully.

- X, Slides — A book sign indicates where you can read more on the theory behind the subject discussed. Numbers indicate chapters and sections in “Statistical Pattern Recognition”. “Slides” means the theory is discussed in the slides, which can be downloaded in hand-out format from the Blackboard site.

Week 1

Basics of Machine Learning

Objectives When you have done the exercises for this week, you

- should be able to mathematically derive decision boundaries given simple probability density functions,
- should be able to perform simple computations with Bayes' rule,
- should be familiar with working under Python,
- understand some PRTools commands,
- know what an object and a dataset are,
- should be able to construct, visualize and classify some simple datasets.

1.1 Decision Theory

Exercise 1.1 **(a)** Assume that we managed to represent objects from a two-class classification problem by a single feature. We know that the objects from class ω_1 have a Gaussian distribution with $\mu_1 = 0$ and $\sigma_1^2 = 1/2$, and the objects from class ω_2 have a Gaussian distribution with $\mu_2 = 1$ and $\sigma_2^2 = 1/2$. Derive the position of the decision boundary when both class priors are equal. 2.2

(b) Again, assume we have a two-class classification problem in a 1D feature space, but now assume that objects from class ω_1 have a uniform distribution between 0 and 1, and objects from class ω_2 have a uniform distribution between 2 and 3. Where is the decision boundary now?

(c) And where is the decision boundary when the objects from class ω_2 have a uniform distribution between 0.5 and 1.5? (The distribution of ω_1 did not change, classes have equal prior.)

(d) And where is the decision boundary when the objects from class ω_2 have a uniform distribution between 0.5 and 2.5? (The distribution of ω_1 did not change, classes have equal prior.)

Exercise 1.2 **(a)** Assume we represent the objects in a two-class classification problem by a single feature. We know that the objects from class ω_1 have a Gaussian distribution with $\mu_1 = 0$ and $\sigma_1^2 = 1/2$, and the objects from class ω_2 have a Gaussian distribution with $\mu_2 = 1$ and $\sigma_2^2 = 1/2$. Derive the position of the decision boundary when both class priors are equal, but we have a loss matrix of:

$$L = \begin{bmatrix} 0 & 0.5 \\ 1.0 & 0 \end{bmatrix}. \tag{1.1}$$

(b) Assume again we have a two-class classification problem in a 1D feature space, but now assume that objects from class ω_1 have a uniform distribution between 0 and 1, and objects from class ω_2 have a uniform distribution between 0.5 and 2.5. Given the loss matrix (1.1), where is the decision boundary now?

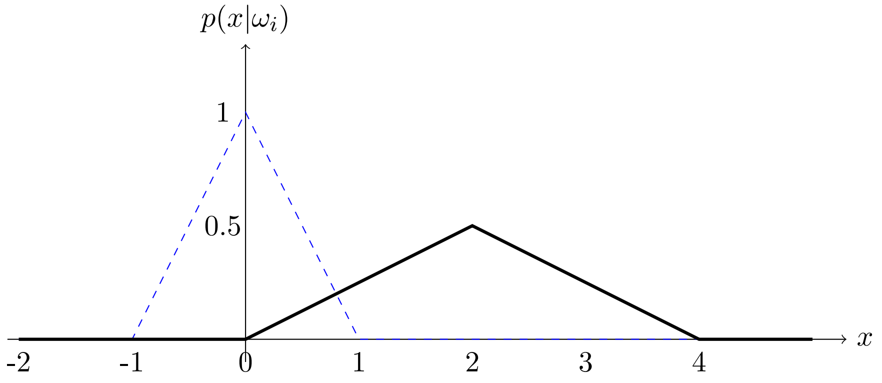


Figure 1.1: The class-conditional probabilities of two classes $p(x|\omega_1)$ (dashed blue line) and $p(x|\omega_2)$ (solid black line) in a 1-dimensional feature space.

Exercise 1.3 In figure 1.1 two triangular-shaped class conditional probability density functions are given. The first one $p(x|\omega_1)$ is indicated by a dashed blue line and the second $p(x|\omega_2)$ with a solid black line. The class priors are assumed equal here.

(a) Again, use the Bayes' rule to derive the class posterior probabilities of the following objects: $x = 3$, $x = -0.5$, $x = +0.5$. To which class are the objects therefore assigned?

(b) Which object is on the decision boundary of the Bayes classifier?

Exercise 1.4 Now assume that class ω_1 in figure 1.1 is twice as small as class ω_2. That means $p(\omega_1) = 1/3$ and $p(\omega_2) = 2/3$.

(a) Compute the posterior probabilities for $x = 3$, $x = -0.5$, $x = 0.5$.

(b) Where is the decision boundary of the Bayes classifier now?

Exercise 1.5 Compute the Bayes error for the class distributions given in figure 1.1, where the classes have equal prior.

1.2 Prtools for Python

In order to learn something about many machine learning methods, the toolbox PRTools was developed. Originally, it is a Matlab toolbox, but it has been, provisionally, translated into Python. It should mimic the most important parts of PRTools for Matlab.

Again, you get get PRTools for Python from `https://github.com/DMJTax/prtools`.

The advantages of PRTools (for Matlab or Python) are:

1. A uniform interface to datasets and mappings. It is very straightforward to wrap datasets or machine learning algorithms in mappings. This then allows for a consistent visualisation, fitting and evaluation of models.

2. This standarization also allows for the combination (in particular, the sequential stacking) of mappings. This makes it very easy to combine preprocessing steps, feature reduction methods and classfiers or regressors into one mapping. The training and evaluation of such combined mappings is identical to the training and evaluation of a single mapping.

Of course, you are not required to use PRTools. If you feel more comfortable with (for instance) `scikit-learn`, please use that. But then we will not be able to help with bugs or implementation problems.

When you are starting with Python, we assume that you already imported `numpy` and PRTools, like[1]:

```
>>> import numpy as np
>>> import prtools as pr
```

For plotting purposes we also use `matplotlib`, so also include:

```
>>> import matplotlib.pyplot as plt
```

1.3 Datasets

One key entity in Machine Learning is the idea of an *object*. We always assume we can represent any object by a set of values, often just measurements. Whatever object we are considering, in order to perform operations on an object by a computer, we have to encode this object by some numbers. An object in real life and in the computer are therefore two different things. Furthermore, in this course we will use the convention that each object is represented by a row vector of measurements (thus an $1 \times d$ array).

When you want to conclude something from data, it is often not from *one* object, but from a *set* of objects. We assume we have a set of objects from which we want to obtain some knowledge. This set is called a dataset. A dataset is a set of n objects and is stored in a $n \times d$ array.

After you have specified the problem you want to solve, you have to collect examples and do measurements on these examples. For that, you have to define what features are likely to be informative for the problem you specified.

[1]To get the Python code, please have a look at the link given on page 6.

Exercise 1.6 **(a)** Make a fake dataset containing 10 objects with 3 measurements each. Invent the measurement values yourself. Below is an example piece of code which fills the matrix `x` with 2 objects, each containing 3 measurements:

```
>>> x = np.array([[ 0.7,0.3,0.2],[2.1,4.5,0]])
```

Make your own matrix `x`.

(b) Compute the means (using `np.mean`) and standard deviations (`np.std`) of your 3 measurements of 10 objects. What is the difference between `mean(x)`, `mean(x,axis=0)` and `mean(x,axis=1)`?

Exercise 1.7 **(a)** When a dataset contains just two features per object, it can be visualized in a scatterplot (we come back to this later). Make a scatterplot by:

```
plt.scatter(x[:,0],x[:,1])
```

Looking at the data matrix you created in Exercise 1.6, find out which object is plotted where in the plot.

(b) When you look at the scatterplot, can you identify outlier objects, or structure in the data?

When the dataset is used to train classifiers, it is also required that for each object a class label is present. This indicates from which class the object originates, according to the expert (i.e. you).

Exercise 1.8 **(a)** Invent labels for the objects that you defined in the previous question. The labels can be numbers, like `1` or 2. Store them in a column vector `lab`, and create a PRTools dataset by

```
>>> lab = np.array([1,1,1,1,2,2,2,2,2,2]).T
>>> a = pr.prdataset(x,lab)
```

Check if the resulting dataset has the correct number of objects, the correct number of features and correct number of classes (correct here means: what you expect). You can do that by just typing the variable on the Matlab command line:

```
>>> print(a)
```

A scatterplot is the most simple plot you can make: it simply plots the first measurement against the second measurement. If you have three measurements, you can use 3D plots; if you have even more, you will have to select at most three of them by hand (although later we will discuss ways of visualising more measurements at once).

Exercise 1.9 Load the dataset "boomerangs" (use the function `boomerangs` and choose the number of objects to generate).

(a) Use `pr.scatterd(a)` to make a scatterplot of the first two features, and `pr.scatterd(a[:,[1,2]])` for the features 2 and 3 (note that Python starts counting with 0).

There are many other (artificial) datasets defined in PRTools. The table below lists a few of them:

`gendatb`	Generation of banana shaped classes
`gendatc`	Generation of circular classes
`gendatd`	Generation of two difficult classes
`gendats`	Generation of two Gaussian distributed classes
`gendath`	Generation of the Higleyman dataset

When you want to extract the original data matrix from a `prdataset`, you can use the '+'-operator:

```
   >> a = pr.gendatb()
   >> b = +a
   >> print(b)
```

This is sometimes useful when you want to remove the labels from a dataset:

```
   >> a = pr.gendatb()      # data with labels
   >> b = pr.prdataset(+a)  # data without labels
   >> print(b)
```

The labels are stored in `a.targets` so you can retrieve them with `lab = a.targets`.

1.4 Creating mappings and decision boundaries

On the `prdataset` you can perform operations. If the dataset is labeled, you can train a classifier. If the dataset has arbitrary real-valued targets, you can train a regressor. But you can also do feature normalisation, feature reduction, or clustering. All these operations are stored in a `prmapping`.

Assume we want to train a nearest mean classifier on the Banana dataset. In PRTools you do:

```
>>> a = pr.gendatb()
>>> w = pr.nmc(a)
>>> print(w)
Nearest mean, 2 to 2 trained mapping
```

When a mapping is trained, it can be applied to a dataset using the operator `*`:

```
>>> b = a*w
>>> print(b)
Banana dataset 100 by 2 prdataset with 2 classes: [50 50]
```

The result of the operation `a*w` is again a dataset. It is the classified, rescaled or mapped result of applying the mapping definition stored in `w` to `a`.

For mappings which change the labels of the objects (so the mapping is actually a classifier) the routines `labeld` and `testc` are useful. `labeld` and `testc` are the general classification and testing routines respectively. They can handle any classifier from any routine.

```
>>> lab = b*pr.labeld()
>>> print(lab)
[[-1]
 [-1]
 [-1]
 [-1]
   ...
 [ 1]
 [ 1]]
>>> e = b*pr.testc()
0.14
```

☞ Note that in the above examples we use the typical MATLAB conventions: everything is a matrix, and operations are often defined in terms of 'multiplication' `*`. If you prefer the PYTHON way of working, you can also call the methods explicitly:

```
>>> a = pr.gendatb()
>>> w = pr.nmc()
>>> w.train(a)
>>> b = w.eval(a)
>>> e = pr.testc(b)
```

A few of the available classifiers is listed below:

ldc	Linear discriminant analysis
qdc	Quadratic discriminant analysis
nmc	Nearest mean classifier
fisherc	Fishers linear discriminant
knnc	k-nearest neighbor classifier
parzenc	Parzen classifier
naivebc	Naive-Bayes classifier
mogc	Mixture-of-Gaussians classifier
stumpc	decision stump classifier
dectreec	Decision tree classifier
adaboostc	AdaBoost
svc	Support vector classifier

The list is not complete. Feel free to add your favorite classifier!

Some classifiers require additional hyperparameters to be specified. For instance, in the support vector classifier, you can specify the kernel, a kernel parameter, and a regularisation parameter. Or in the k-nearest neighbor classifier you can specify the number of neighbors k. You can supply that as additional input during training:

```
>>> w = pr.svc(a,('rbf',4.5,1))
```

You can also specify an *untrained* mapping beforehand with the required hyperparameters, and train it afterwards on some training set:

```
>>> u = pr.svc(('rbf',4.5,1))
>>> w = a*u
```

or

```
>>> w = pr.svc(('rbf',4.5,1))
>>> w.train(a)
```

Finally, you can also visualise the decision boundary of classifiers. This is done using the function `plotc`. In order to see the relevant region in the feature space, first a scatterplot of the (training) dataset has to be made. For example:

```
    >>> a = pr.gendath()
    >>> w = pr.parzenc(a)
    >>> pr.scatterd(a)
    >>> pr.plotc(w)
```

Exercise 1.10 Practice the use of PRTOOLS. Create some of the artificial datasets, make scatterplots of the data, train some classifiers and plot their decision boundaries.

(a) Train a classifier and plot its decision boundary (together with the training set). Can you verify that the function `testc` gives the correct classification rate on the training set as you see in the plot?

1.5 Classification with Normal Densities

Exercise 1.11 Have a look at Figures 2.3 to 2.6 in the book. 2.4.1

(a) Are the correlations that can be observed in these figures positive, negative, or zero?

(b) What would it mean for the correlation if the ellipsoidal configuration in Figure 2.6 runs from the bottom left to the top right corner instead?

Exercise 1.12 **(a)** Make a rough sketch of the shape of the ellipsoids of a normal density for which the covariance matrix has the form $\left[\begin{smallmatrix}1 & 0\\ 0 & 9\end{smallmatrix}\right]$. Take this ellipsoid, what is the ratio of the length of its long axis in comparison with its short axis?

(b) Imagine we rotate the above sketch of ellipsoids clockwise such that their orientation becomes diagonal. Qualitatively, what happens to the variances and the covariances for this new normal distributions? How do they compare to the original variances and covariances as given in the covariance matrix above?

(c) Draw a figure visualizing a 2D normal density in which the two variables/features are completely correlated, i.e., the correlation coefficient between the two variables is 1. Give the covariance matrix that matches your sketch.

Exercise 1.13 Generate 1000 random point from a 2D standard normal distribution using `randn`. (Note the discrepancy between the math, in which typically column vectors are employed, and computational software, in which feature vectors often are row vectors. This means, in this case, that the matrix with random numbers should be size 1000×2.)

(a) Turn these random points into a `prdataset a` and `scatterd` the data.

The command `w = gaussm(a)` determines estimates for a normal distribution based on the data set `a`, and stores the estimated parameters in the mapping `w`.

(b) Use `plotm(w)` to visualize the 2D density estimates on top of the scattered data.

Exercise 1.14 Consider two 2D normal distributions with different means but equal covariance matrices. Assume the latter to be equal to a multiple of the identity matrix, i.e., take Σ to be $\left[\begin{smallmatrix} c & 0 \\ 0 & c \end{smallmatrix}\right]$ for some $c > 0$. The class priors are not necessarily equal.

(a) What shapes can the decision boundary take on? Demonstrate this mathematically starting from the expressions in Section 2.4.2 in the book.

2.4.2

Exercise 1.15 Consider two 2D normal distributions with different means but equal covariance matrices.

(a) What shape does the optimal decision boundary have?

Generate 10 data points per class from a data set that fulfills the above assumptions. Create a data set `a` with these 20 points and these two classes. We can estimate linear discriminants by means of `ldc(a)` and quadratic discriminant boundaries by means of `qdc(a)`.

(b) Given the shape of the optimal decision boundary, how does the boundary look for the trained normal density based quadratic discriminant?

The PRTools command `plotc` plots the decision boundary.

(c) Scatter the data set `a` and plot the decision boundary estimated by means of `ldc(a)` and `qdc(a)`. Can you explain their difference? You might want to revisit the previous question (b).

(d) What happens when the number of points per class increases? What happens in the limit of an infinite number of points?

Exercise 1.16 Generate a 2D data set with 2 classes with 10 samples from a uniform distribution (using `rand`) and 10 samples from a normal distribution (using `randn`), respectively.

(a) Train a `ldc` and a `qdc` on this data, scatter the data, and plot the two decision boundaries (with `plotc`). Determine the how many points are on the wrong side of the boundary.

(b) With `w` a trained classifier and `a` a data set, the labels that the classifier assigns to the data points in `a` can be retrieved using `labeld`: `a*w*labeld`. Check your count from the previous question using this command.

1.6 Density estimation using Parzen densities

Next to classifiers, PRTools also has the possibility to estimate densities. In this section we are going to estimate the density using a Parzen density estimator, called `parzenm` in PRTools.

Exercise 1.17 **(a)** We start with creating a simple dataset with:

```
>>> a = pr.gendats([20,20],1,8);
```

(Type `help(pr.gendats)` to understand what type of data we have now.)

(b) We define the width parameter h for the Parzen kernel:

```
>>> h = 0.5;
```

(c) The function `parzenm` estimates a density for a given dataset. In most cases a PRTools `prdataset` is labeled, and these labels are used in the function `parzenm` to estimate a density for each class. To define a Parzen density estimator with a certain width parameter `h` on the entire dataset, ignoring labels, type:

```
  >>> a = pr.prdataset(+a)
  >>> w = pr.parzenm(a,h)
```

This mapping can now be plotted along with the data:

```
>>> pr.scatterd(a); pr.plotm(w)
```

If your graphs look a little "bumpy", you can increase the grid size PRTools uses for plotting:

```
>>> pr.plotm(w,gridsize=100)
```

and try the above again.

(d) Plot the Parzen density estimate for different values of `h`. What is the best value of `h`?

When you want to evaluate a fit of a density model to some data, you have to define an error. One possibility is to use the log-likelihood, defined as:

$$\mathrm{LL}(\boldsymbol{X}) = \log\left(\prod_i \hat{p}(\boldsymbol{x}_i)\right) = \sum_i \log\left(\hat{p}(\boldsymbol{x}_i)\right) \tag{1.2}$$

The better the data $\boldsymbol{x}$ fits in the probability density model $\hat{p}$, the higher the values of $\hat{p}(\boldsymbol{x})$ will be. This will result in a high value of $\sum_i \log\left(\hat{p}(\boldsymbol{x}_i)\right)$. When we have different probability density estimates $\hat{p}$, we have to use the one which has the highest value of LL.

Note that when we fill in different values for the width parameters h in the Parzen density estimation, we have different estimates $\hat{p}$. Using the log-likelihood as a criterion, we can optimize the value of this free parameter h to maximise LL.

To get an honest estimate of the log-likelihood, we have to evaluate the log-likelihood (1.2) on a *test set*. That means that we have to make (or measure) new data from the same distribution as where the training data came from. When we would evaluate the log-likelihood on the data on which the probability density was fitted, we would get a too optimistic estimation of the error. We might conclude that we have fitted the data very well, while actually a new dataset from the same distribution does not fit in the density at all! Therefore, if you want to evaluate the performance of an estimated $\hat{p}$, use an independent test set!

Exercise 1.18 Use the data from the same distribution as in the previous exercise to train a Parzen density estimator for different values of h. Compute the log-likelihood of this training set given the estimated densities (for different h):

```
a = pr.gendats([20,20],1,8)                  # Generate data
a = pr.prdataset(+a)
hs = [0.01,0.05,0.1,0.25,0.5,1,1.5,2,3,4,5] # Array of h's to try
LL = np.zeros(len(hs))
for i in range(len(hs)):                     # For each h...
   w = pr.parzenm(a,hs[i])                   #   estimate Parzen density
   LL[i] = np.sum(np.log(+(a*w)));           #   calculate log-likelihood

plt.plot(hs,LL);                   # Plot log-likelihood as function of h
```

(since `w` is the estimated density mapping `w`, the estimated density $\hat{p}$ for objects in a dataset `a` is given by `+(a*w)`).

(a) What is the optimal value for h, i.e. the maximal likelihood? Is this also the best density estimate for the dataset?

Exercise 1.19 **(a)** Use the same data as in the previous exercise, but now split the data into a training and test set of equal size. Estimate a Parzen density on the training set and compute the Parzen density for the test set. Compute the log-likelihood on both the training and test sets for $h = [0.1, 0.25, 0.5, 1, 1.5, 2, 3, 4, 5]$. Plot these log-likelihood vs. h curves:

```
[trn,tst] = pr.gendat(a,0.5)           # Split into trn and tst, both 50%
hs = [0.01,0.05,0.1,0.25,0.5,1,1.5,2,3,4,5] % Array of h's to try
Ltrn = np.zeros(len(hs))
Ltst = np.zeros(len(hs))
for i in range(len(hs)):                  # For each h...
   w = pr.parzenm(trn,hs[i])              #   estimate Parzen density
   Ltrn[i] = np.sum(np.log(+(trn*w)))     #   calculate trn log-likelihood
   Ltst[i] = np.sum(np.log(+(tst*w)))     #   calculate tst log-likelihood

plt.plot(hs,Ltrn,'b-')             # Plot trn log-likelihood as function of h
plt.plot(hs,Ltst,'r-')             # Plot tst log-likelihood as function of h
```

What is a good choice of h?

This week you saw the basis of machine learning: object definition, data collection, and Bayes' rule. Starting from good measurement data, the rest of the analysis (visualisation, clustering, classification, regression) becomes much easier. Starting from poorly defined objects or poorly sampled datasets with insufficient example objects, your analysis becomes very hard and no clear conclusions will be possible (except that more data is needed).

1.7 The scaling problem

In this last section we have a quick look at the scaling problem. It appears that some classifiers are sensitive to the scaling of features. That means, that when one of the features is rescaled to very small or very large values, the classifier will change dramatically. It can even mean that the classifier is not capable of finding a good solution. Here we will try to find out, which classifiers are sensitive to scaling, and which are not.

Exercise 1.20 **(a)** Generate a simple 2D dataset (for instance, using `gendatb`) and plot the decision boundaries of the six classifiers listed above. Use $k = 1$ for the `knnc`.

(b) Make a new dataset in which the second feature is 10 times larger than the dataset above. Do this by

```
newtrain = a;
newtrain(:,2) = 10*newtrain(:,2)
```

Train six classifiers `nmc`,`ldc`,`qdc`,`fisherc`,`parzenc`,`knnc` and plot the decision boundaries.

(c) Which classifiers are affected by this rescaling of the feature space? Why are they affected, and others not?

(d) Is it an advantage or a disadvantage?

Week 2

Linear Regression and Linear Classifiers

These exercises are meant to give you both some practice with the material covered in the lectures and the literature and an impression of what you are expected to know. There are a lot of exercises. So, judge for yourself which exercises you need to, want to, or should practice. There are anyway too many to go through in the 2 hours of exercise lab that we have. On another note, not all exercises are thoroughly checked. If you think something is wrong, unclear, etc., let us (i.e., any of the lecturers or TAs) know.

Objectives When you have done the exercises for this week and went through the related reading material, you should

- be able to formulate the basic least squares regression models and derive its optimal estimators,
- comprehend the idea and use of polynomial and transformed regression,
- understand how Fisher's linear discriminant is formulated in terms of standard linear regression,
- know how a perceptron classifier is optimized.
- be able to derive the expression for the posteriors in the case of logistic discrimination,
- have an idea of the basic shape of the logistic posterior.

2.1 Linear Least Squares without and with Intercept

OPTIONAL

Exercise 2.1 Consider standard linear regression with the squared loss as the performance measure:

$$\sum_{i=1}^{N}(x_i^T w - y_i)^2 = \|Xw - Y\|^2. \tag{2.1}$$

Note that in the expression above there is some confusing notation going on. The (feature) vector $x_i \in \mathbb{R}^d$ is a column vector, while all features per object in $X \in \mathbb{R}^{N \times d}$ are in rows. Y is an N-vector with all corresponding outputs.

The aim is to minimizing this sum of squared residuals between the linearly predicted and actual output over $w \in \mathbb{R}^d$.

(a) Assume that $(X^T X)^{-1}$ exists. Show that $(X^T X)^{-1} X^T Y$ gives a least squares solution to the above problem, i.e., it minimizes $\|Xw - Y\|^2$.

(b) Given that $(X^T X)^{-1}$ exists, what does that tell us about the data? More specifically, what limitation on the number of observations does this imply, what does invertibility say about the dimensionality of the (affine) subspace our data is in, and what difference does the presence or absence of the origin in this subspace make? To what extent are these limitations enough to guarantee invertibility?

Let us now allow for an intercept (or bias term), i.e., we also model a constant offset in the regression function. We do this by the trick of adding a column of ones to the matrix X. Let Z refer to this new matrix.

Consider standard linear regression (with intercept) with the squared loss as the performance measure, $\|Zw - Y\|^2$, which we want to minimize for w.

Exercise 2.2 **(a)** Assume that $(Z^T Z)^{-1}$ exists. Show that $(Z^T Z)^{-1} Z^T Y$ gives a least squares solution to this least squares problem.

(b) Given that $(Z^T Z)^{-1}$ exists, what does that tell us about how the data is scattered? Can you formulate necessary and sufficient requirements to for the existence of this inverse?

(c) Construct an example data set, for which the inverse of $X^T X$ exists, while the inverse for $Z^T Z$ does not.

Exercise 2.3 Let us consider a couple of settings in which we have very few observations. These problems may be referred to as underdetermined (do you understand the choice of adjective?).

(a) Given a data set consisting of one training point only. The input is $x = \pi$ in 1D, while the output y equals e. Sketch/describe all linear solutions with intercept that minimize the squared loss on this data set.

(b) Give a mathematical expression for the (set of) solutions of this 1D problem.

(c) Describe how the linear least squares solutions look if $X = \left(\begin{smallmatrix} 1 & 1 \\ -2 & 1 \end{smallmatrix}\right)$ and $Y = \left(\begin{smallmatrix} 1 \\ -1 \end{smallmatrix}\right)$

(d) For this last problem with 2D inputs, what is the value that the minimizer takes on? In other words, what is the sum of the squared residuals in this case?

(e) Forget about the intercept for a moment and describe how the linear least squares solutions look if $X = (1, 1)$ and $Y = \pi$.

(f) Look again at the three solution sets that you have determined for the three regression problems above. Could you come up with a good rule to single out an element from every one of these three sets? What is your reason for singling out this solution?

END OPTIONAL

Exercise 2.4 Consider a regression training data set with four 1D inputs $X = (-2, -1, 0, 3)^T$ and corresponding outputs $Y = (1, 1, 2, 3)^T$.

(a) Let us assume that we fit a linear function without intercept to this data under squared loss. Calculate the optimal function fit for the given data set.

(b) Let us now also include an intercept. Still, we stick to fitting linear functions that we fit using the squared loss. Calculate the optimal value that we find for the intercept.

(c) Think of polynomial regression (see also Section 2.2), what is the minimum polynomial degree that we need in order to fit the regression curve exactly to the training data? Is there a difference between the situation with and without intercept?

2.2 Polynomial Regression and Other Feature Transformations

The function `linearr` allows one to perform polynomial regression. Polynomial regression fits a polynomial of some maximal degree to the data in a least squares sense. Even though this results in a nonlinear function in x, the problem may still be referred to as linear regression as the regression function is, in fact, linear in the unknown parameters w estimated from the data. Enough with the confusion, let's do some experiments!

Exercise 2.5 Using `gendatr` or `prdataset`, generate data where the inputs x are drawn uniformly from the interval $[0, 1]$ and the corresponding outputs y are obtained by squaring this value and adding Gaussian noise to the inputs: $y = x^2 + \varepsilon$ with ε a random sample from the standard normal distribution.

(a) Study the behavior of polynomials of degree 0 to 3 for different training set sizes (e.g. 4, 40, and 400 samples?). You may want to have a look at the data and the fitted polynomial models. You can also estimate the squared error using a somewhat large test set.

The idea of not only using the original features, but also powers of those feature values (as in polynomial regression) can of course be applied more liberally. There is, in principle, no reason to limit oneself to polynomials. Especially if you understand what data you are dealing with, if you understand the problem you are going to crack, or if there is any type of a priori knowledge available, you could even design dedicated features.

OPTIONAL

Exercise 2.6 Assume you would like to predict the temperature in the Netherlands (output) on the basis of the specific day of the year, encoded as x_1th month in year and x_2th day in month (input). What kind of transformation(s) of this initial 2D input data would you use in order to get "linear" regression to work on this data? (We're looking for a quick-and-dirty solution here; no need to turn this into a very extensive study. Still, for inspiration you could check `http://projects.knmi.nl/klimatologie/daggegevens/selectie.cgi` where you can download actual temperature data of the Netherlands.)

END OPTIONAL

Exercise 2.7 Consider the following regression problem from 2D to 1D. The input vectors x are from a standard normal distribution in 2D. The corresponding outputs, y, are obtained through the following equation: $y = 50\sin(x_1)\sin(x_2) + \varepsilon$, where ε has a standard normal distribution as well (but in 1D of course).

(a) Visualize 10,000 samples from this regression problem and have a look at the data from different points of view.

(b) Fit a linear linear regression to these 10,000 points and measure the error on a separate test set.

(c) Fit a second degree linear regression to these 10,000 points. Again measure the error on a separate test set. Try the same for some higher degrees.

(d) Why can these higher-degree polynomials not fit this data better than the standard linear regressor? Can you figure out what seems to be happening? (If not, maybe the next question helps.)

(e) Let the input x be as in the above, but now take $y = x_1x_2$. Fit linear regressions of degree 1 and 2 and report the error they make and/or visualize the solutions in comparison with the actual training data.

OPTIONAL

Exercise 2.8 **(a)** Given that we have a data set with d features, how many monomials of degree m do we have? (Roughly, a monomial of degree m is the product of exactly m variables, where every variable can occur multiple times. E.g. x^5 and x^2z^3 are monomials of degree 5. 1 is the only zero degree monomial.)

(b) Compared with the number of features that `linearr` uses with increasing degree m, does the number of features when all cross-terms are included grow essentially faster?

(c) Can you come up with real-world classification problems where a polynomial expansion of even a moderate degree becomes infeasible?

END OPTIONAL

2.3 Fisher's Linear Discriminant

Fisher's linear discriminant (FLD, in PRTools referred to as Fisher's linear classifier , `fisherc`, and probably known under various other names as well, e.g. linear regression classifier, Fisher classifier, least squares classifier) is the classifier that can be constructed with the use of standard linear regression. We consider the two-class case, in which the input variables are simply taken to be the feature vectors in our data set, while the corresponding classes are typically encoded numerically as $+1$ and -1, i.e., $Y \in \{-1, 1\}^N$ in case of N training samples. A test sample x is then assigned to the class $\text{sign}(w^T x)$. One typically assumes that an intercept is included in the regression model.

Exercise 2.9 **(a)** Assume N is the total number of training samples and we have the same number of samples in both classes. Show that the optimal w is given by $2T^+(m_+ - m_-)$, where T is the standard (biased) sample estimate for the covariance matrix of the data and m_+ and m_- are the estimated class means for the positive and negative classes, respectively.

OPTIONAL

(b) Say we linearly transform the features of a problem by means of a nonsingular matrix A. Show that in the case that $X^T X$ is invertible, the performance in the original and the linearly transformed space is exactly the same, i.e., the classifier is invariant under nonsingular linear transformation.

(c) Consider 2 samples in 2D: one point from one class is located in $(0, 0)$, the other point from the other class is in $(2, 1)$. Draw the decision boundary of the FLD between these two point. Now perform a simple linear scaling of the first feature and divide its values by 2. Draw the new decision boundary that is obtained by retraining with the transformed data and compare it to the decision boundary one would get if the procedure would be invariant under linear transformations.

END OPTIONAL

2.4 A Probabilistic Regression Model

At times, it can be convenient to have a probabilistic regression model. For instance, because its predictions can be more easily combined with other probabilistic approaches or because it may allow one to say something about the possible spread/uncertainty of an estimate.

Exercise 2.10 Let us consider a regression problem setting with a simple 1D input and 1D output. Assume that x has a normal distribution with variance τ^2 and mean ν. In addition, given x, let y be normally distributed around $xw + w_0$ with variance σ^2.

(a) Write out explicitly the joint probability distribution for observations $(x, y)^T$.

(b) Assume ν, τ, and σ known. Consider the likelihood of this model for a data set $\{(x_i, y_i)^T\}_{i=1}^N$ and derive the maximum (log-)likelihood estimate for w and w_0.

(c) Instead of the normal distribution for x, we now take a generic distribution, say, $p(x|\theta)$, which depends on some parameter θ. When estimating the parameters, θ and w by means of maximum likelihood, why can they be optimized separately? That is, why does one not need to know the one to determine the other.

This last exercise shows why we may as well forget about the marginal distribution over x if one is merely interested in the fitting of w.

(d) Determine the ML estimates for w, w_0, and σ given that these are the free parameters of the model.

OPTIONAL

Finally, we extend our probabilistic model by means of a so-called prior probability. A prior tries to encode a priori knowledge that we may have about a certain parameter or, more generally, about the model as such. The prior knowledge we assume here is that it is more likely that the solution w we are looking for has small coefficients. We do this by assume a normal prior with mean 0 and a variance s^2 for w. Such prior is then multiplied with the likelihood to come to the overall objective function to be optimized.

(e) Consider the likelihood of this model for a data set $\{(x_i, y_i)^T\}_{i=1}^N$ and derive the estimates for w and w_0 that maximize the likelihood times the above prior.

END OPTIONAL

2.5 Error Rates and Linear Classifiers

Exercise 2.11 Given a 1D 2-class classification problem with numerical class labels $Y = \{-1, +1\}$. From this problem, we have observed three feature "vectors", 1,2, and 6, from class $y = +1$ and two samples at π and 5 from the class $y = -1$.

Using the Iverson bracket, define the hypothesis space H of linear classifiers as follows: $H = \{h(x; a) := 2[x - a > 0] - 1 | a \in \mathbb{R}\}$.

(a) Draw (or plot) the classifier for the parameter value $a = 0$. Do the same for the value $a = -1$.

(b) Draw (or plot) the error rate on the training set against the parameter value a.

(c) For which parameter values is error on the training set minimal? What is the error rate?

(d) Let's now change H into the set $\{h(x; a) := 2[x - a < 0] - 1 | a \in \mathbb{R}\}$, which $h(x; a)$ do now give you the best performance on the training set?

OPTIONAL

Exercise 2.12 We have a 2-class classification problem in 2D for which we are going to consider some specific linear classifiers. The class labels are from $Y = \{-1, +1\}$ and we have the feature vectors $0, -1$, $1, 1$, and $-1, 1$. The first belongs to class -1 the last two are from the positive class.

Let us consider two hypothesis classes. One is $H_0 = \{h(x; a) = \text{sign}(a^T x) | a \in \mathbb{R}^2\}$ and the other is $H_1 = \{h(x; a) = \text{sign}(x_1 + a_1 x_2 + a_2) | a \in \mathbb{R}^2\}$.

(a) Visualize, for both hypothesis classes, how the error rate depends on the choice of parameters. (For this visualization purpose, tt should be OK to limit a to something like $[-2, 2]^2 \subset \mathbb{R}^2$) Certainly for H_0, it is doable (and potentially instructive) to sketch the situation with pen and paper, but I can imagine you want to use some sort of 3D function plotting for this.

(b) H_0, determine the (precise) subset of $\mathbb{R}^2$ for which the error rate on this training set is minimal. What is the error rate achieved in this case?

Characterizing the set of optimal solutions in H_1 seems much more troublesome. Do have a go at it and see how far you get! Let us at least have a look at the following.

(c) What are the three corner points that you can identify in your plot or sketch from 2.12(a)? Draw the data set and indicate which three classifiers these correspond to. Give their precise parameter values.

(d) Swap the labels for the training points and redo the plot from 2.12(a) (and compare the latter to the previous visualization). What does the best training set error become and what are the corner points of the set of all optimal a now?

(e) Construct a data set for which you can find an a from H_0 that gives you an error of $\frac{1}{2}$, while in H_1 there is a perfect classifier achieving 0 error. How about the other way around

(f) Is there a data sets for which the training error rate cannot be better than 1 when using either H_0 or $H{-1}$?

2.6 Linear Bayes Classifier

Exercise 2.13 **(a)** Consider the Bayes classifier you found in Exercise 1.3. Give a parameterization of the linear classifier that coincides with the Bayes classifier.

END OPTIONAL

2.7 The Perceptron

OPTIONAL

Exercise 2.14 **(a)** Let the two samples in one class consist of the two corners of the left side of the unit square $[0,1]^2$ and let the other class take the opposite two corners as samples.

Determine the perceptron solution when you take $\eta = 1$, initialize with $w = (0,0)$, and let $\phi(x) = (x1)$ for $x \in \mathbb{R}^2$.

END OPTIONAL

Exercise 2.15 **(a)** Generate a simple, linearly separable, dataset `gendats([20,20],2,6)`. Make a scatterplot and check that the dataset is linearly separable. If it is not, generate a new dataset until it is.

Extract from this dataset the feature matrix $\mathbf{X}$ and the label vector (hint: use `getnlab`).

(b) Implement the perceptron algorithm, described on pages 192 to 194 of the book *Pattern Recognition and Machine Learning*. Don't forget to add the constant term to the features (as explained at the start of section 3.3 from the book)!

(c) Train the perceptron on the data that you generated and plot the normal vector $\mathbf{w}$ and the decision boundary (i.e. all points where $\mathbf{w}^T\mathbf{x} = 0$) in the figure. Does the weight vector make sense: does it separate the classes?

(d) What happens when the classes in the dataset are *not* linearly separable? Check what solution your implementation gives.

Exercise 2.16 Test and compare the Fisher classifier and `perlc` on the same dataset that you have used in Exercise 2.15 by plotting the decision boundary in a scatterplot. Of course, you could also measure and compare their performances on a large, new, and unseen test dataset.

Week 3

Losses, Regularization, Evaluation

These exercises are meant to give you both some practice with the material covered in the lectures and the literature and an impression of what you are expected to know. There are a lot of exercises. So, judge for yourself which exercises you need to, want to, or should practice. There are anyway too many to go through in the 2 hours of exercise lab that we have. On another note, not all exercises are thoroughly checked. If you think something is wrong, unclear, etc., let us (i.e., any of the lecturers or TAs) know.

Objectives When you have done the exercises for this week and went through the related reading material, you should

- be able to formulate the L_2-regularized least squares regression models
- be capable of identifying the trade-off between bias and variance in regression and classification and measure it,
- be able to formulate L_1 regularization and understand its feature selection abilities,
- understand how regularization influenced bias and variance,,
- appreciate the general formulation of a learning problem in terms of hypotheses, loss, and regularizer,
- can explain what sources of performance variability there are,
- understand the difference between train and test error,
- know what a learning curve is,
- explain what cross validation is.

3.1 Regularization

Consider standard linear regression with the squared Euclidean norm as the so-called regularizer (also referred to as L_2 regularization):

$$\sum_{i=1}^{N}(x_i^T w - y_i)^2 + \lambda\|w\|^2 = \|Xw - Y\|^2 + \lambda\|w\|^2. \tag{3.1}$$

The aim is to minimizing this over $w \in \mathbb{R}^d$. The regularizer tries to keep the weights small. This form of regression is also referred to as ridge regression (`ridger`).

We investigate the effect of this regularization term a bit. First, however, some (optional) math.

OPTIONAL

Exercise 3.1 **(a)** Given a data matrix X. Show that $X^T X$ is positive semidefinite (psd) matrix.

(b) Show that $X^T X + \lambda I$, with $\lambda > 0$ and I the $d \times d$-identify matrix, is always invertible.

(c) Show that $(X^T X + \lambda I)^{-1} X^T Y$ gives the least squares solution to the above problem.

(d) Assume λ is fixed to a positive value, what solution do you find if the data set grows larger and larger? Or in other words, how does the regularizer's influence change with a growing data set size?

(e) What solution is obtained in the limit of λ going to infinity?

Exercise 3.2 Set up an experiment demonstrating that `ridger` does, in fact, not regularize the intercept.

END OPTIONAL

Exercise 3.3 We consider a regression data set where the inputs x are drawn uniformly from the interval $[0, 1]$. The corresponding outputs y are obtained by adding Gaussian noise to the inputs: $y = x + \varepsilon$ with ε a random sample from the standard normal distribution. You can generate regression data sets by means of `gendatr` or `prdataset`. Ridge regression is carried out using `ridger`.

(a) Create a training data set of size 2 and a large test data set (1,000 or 10,000 samples will do). Study the regression fit to the training data for different amounts of regularization (using command like `scatterr` and `plotr`). Also check how the squared error varies for different λs (using commends like `testr`). Check a couple of new draws for the training set and also try a different set size of, for instance, of 100. Try regularization parameters varying from 0 (or something smallish like 10^{-3}) to something larger like 10^3.

(b) Determine (roughly) which value for the regularization parameter gives, on average, the best performance. Determine this optimum for a training set size of 2, of 10, and for a training set size of 100. You can limit your search to λs in the range $[10^{-3}, 10^3]$.

Exercise 3.4 Given a regression data set of fixed size. How would you tune the regularization parameter?

OPTIONAL

3.2 A Pseudoinverse

Exercise 3.5 The solution for the limit of lambda approaching from above to zero, i.e., $\lim_{\lambda \downarrow 0}(X^T X + \lambda I)^{-1}$, is equal to the pseudoinverse of $X^T X$ and denoted $(X^T X)^+$.

Reason why, among all solutions of the *unregularized* regression problem, $(X^T X)^+ X^T Y$ provides the minimum norm solution irrespective of the number of training samples. A precise mathematical proof is not expected.

Exercise 3.6 Let's say that we have two polynomial expansions $P_1(x)$ and $P_2(x)$ for a single input variable x. Assume we have a linear transformation T (i.e., just a matrix) for which $TP_1(x) = P_2(x)$.

(a) Show that, if T is invertible, for unregularized linear regression, these two representations are equivalent. That is, if w_1 and w_2 are the respective solutions, we have $w_1^T P_1(x) = w_2^T P_2(x)$ for all x.

(b) Why is the training loss using representation P_1 never smaller than that obtained with P_2? Can you give an example where it is strictly better?

(c) Construct an example, possibly numerical, that shows that if we use L_2 regularized regression, P_1 and P_2 can lead to different solutions, even if T is invertible. Why is this the case?

END OPTIONAL

3.3 The Lasso

Another way of regularizing that is popular is through an L_1 norm instead of the L_2 norm. This leads to the so-called least absolute shrinkage and selection operator or LASSO for short. What is nice about the LASSO is that it often also results in a selection of features (feature selection is also implemented through some other approaches as we will see later). That is, it automatically leads to a reduction of the number of features that the final regressor depends on. Here, we investigate that behavior a bit.

Exercise 3.7 Consider a 2D regression problem where $x \in \mathbb{R}^2$ is standard normally distributed, while $y = x_1 + \varepsilon/5$, where ε is also from a standard normal distribution.

(a) Take a few samples from the above problem, say 20 or so. Visualize the shape of the objective function for regression with no intercept and *standard L_2 regularization* for different values of λ. Inspect/determine visually for which value of λ at least one of the entries of the minimizing w becomes 0. Does any of the two entries of the optimum ever become zero really?

(b) As in 3.7(a), take the same number of samples from the above regression problem. Visualize the shape of the objective function for different values of λ but now use L_1 regularization. What shape does the objective take on when $\lambda = 0$ and when λ is very, very large? Inspect/determine visually around which value for λ one of the entries of the optimal w becomes 0. Should it really become zero for some λ in this setting?

3.4 Hypothesis Classes and (Surrogate) Losses

In an attempt to develop a general approach to machine learning, a learning problem may be defined in somewhat abstract terms as consisting out of four components. Next to the training data sets D, we have a hypothesis class H, which is the set of all possible models that are considered, a loss, which is a function that tells us how well a hypothesis $h \in H$ fits the data D, and a regularization term. The assumption then is that, what we are looking for, is the hypothesis from H that *minimizes* the loss on the training data.

Exercise 3.8 **(a)** In the linear regression setting above, what is the hypothesis class, what is the loss, what is the regularization term?

(b) Let's say we fit a Gaussian distribution to a data set D by means of maximum likelihood. Give a hypothesis class and a loss. Do we have a regularization term?

In the typical settings that we encounter, the measure of fit can often be expressed as the sum over individual elements in the data set. That is, we define a loss per element $(x, y) \in D$, $\ell(x, y|h)$ and calculate the overall or, more often, the expected loss as the average loss over the data set: $\frac{1}{N}\sum_{i=1}^{N} \ell(x_i, y_i|h)$. Note that this is often what loss refers to: the loss per data point (and this is also how we will typically use it). The expected loss is also referred to as the (empirical) risk. Just mentioning risk most often refers to the expected true loss for a given h: $\int p(x, y)\ell(x, y|h)dxdy$—the integral becomes a sum whenever appropriate, e.g. in the classification setting y is, of course, a discrete label.

OPTIONAL

Exercise 3.9 **(a)** Write down the loss (per data point) for linear regression in case we fit linear functions.

(b) What loss[1] is used when fitting a model to i.i.d. data under log-likelihood?

(c) Can you give a loss that cannot be expressed in terms of, or that is not equivalent to, a sum over individual element in a data set?

(d) Can you give a learning methods for which seems to be difficult to formulate in terms of hypothesis class plus loss?

Exercise 3.10 In classification, we often consider so-called margin-based loss functions. In this setting it is typical to only look at two-class problems and that the class labels are encoded as $+1$ and -1. Margin-based loss functions are loss functions for which there exists a function $v : \mathbb{R} \to \mathbb{R}$ such that $\ell(x, y|h) = v(yh(x))$.

(a) Consider the log-likelihood objective that logistic regression (in d dimensions) optimizes (check your own notes, the book by Bishop, or refer back to the notes used in ML0 by Ng: `http://cs229.stanford.edu/notes/cs229-notes1.pdf`). Assume H is the class of linear mappings. Rewrite the optimization of the log-likelihood in terms of a minimization of a sum and show that every term can be expressed by means of a margin-based loss. Give an explicit expression for v.

[1]It can take on negative values, which may mean that some will not accept is as a loss.

(b) Consider Fisher's linear discriminant. Rewrite its objective function in the form that uses a margin-based loss function. How does v look now?

(c) Can you determine how the margin-based loss that corresponds to the SVM looks like?

(d) What is the (obvious? logical? canonical?) hypothesis class for LDA (or `ldc`) and what is the loss typically used? Do you think that LDA can be formulated in terms of a margin-based loss?

END OPTIONAL

3.5 Regularization and MAP

Let us relate some regularizers to specific forms of MAP estimation.

Exercise 3.11 **(a)** Reconsider the solution found in 2.10(e). With which form of L_2 regularized linear regression do we get to exactly the same solution?

(b) Consider standard multivariate linear regression with squared Euclidean norm regularization as in 3.1. What prior should one assume over w to let the MAP solution coincide with the minimizer of Equation (3.1)?

(c) With what prior do we get to the lasso as discussed in Section 3.3?

OPTIONAL

3.6 Probabilistic Model for Regression

Exercise 3.12 Assume a linear relation $y = w^T x$ between input and output and assume that, given the input, the output is normally distributed with mean 0 and unknown variance σ.

(a) Given a data matrix $X = (x_1, \dots, x_N)^T$ and corresponding output vector $Y = (y_1, \dots, y_N)^T$. Determine the likelihood for w and show that its maximizer is equal to the standard least squares estimator that we already saw in Section 2.1.

(b) Determine the maximum likelihood estimator for σ.

(c) Assume, additionally, a Gaussian prior over w with mean 0. The posterior for w in this setting is Gaussian as well. Determine its mean and covariance using (2.116) from Bishop's book or, for instance, the technique of completing the squares.

(d) Determine the MAP estimator for w.

END OPTIONAL

3.7 Sources of Variation

Exercise 3.13 In this exercise we investigate the difference in behavior of the error on the training and the test set. Generate a large test set and study the variations in the classification error based on repeatedly generated training sets:

```
>> t = gendath([500 500]);
>> a = gendath([20 20]); t*ldc(a)*testc
```

Repeat the last line e.g. 30 times.

(a) What causes the variation in the error?

Now do the same for different test sets:

```
>> a = gendath([20 20]);
>> w = ldc(a);
>> t = gendath([500 500]); t*w*testc
```

Repeat the last line e.g. 30 times.

(b) Again explain what causes the variance observed in the results.

3.8 Learning Curves

The function `cleval` allows you to calculate so-called learning curves. These are curves that plot classification errors against the number of points in the training data set. (Check `help cleval` for the details.)

In this section we are going to study some learning curves for different data sets and different classifiers. Some of the plots and number that will be generated demonstrate certain salient and/or noteworthy behavior.

After reading a question, and before just implementing the things to be implemented, try to think about the outcome to be expected. Eventually, say at the end of this course, you should not be surprised by many of these things anymore!

3.8.1 Staring at Learning Curves

Exercise 3.14 Generate Highleyman classes (`gendath`) with a 1000 samples per class. Enlarge the feature dimensionality of this set by adding 60 dimensions of class independent randomness, i.e., plain noise. Use `cleval` to generate learning curves for `nmc`, `ldc`, and `qdc` using 64, 128, 256, and 512 objects in the training set (make sure that you repeat often enough. . .). Use `plote` to plot these curves in a single figure (check the help).

(a) Can you explain the overall behavior of these curves?

(b) Explain why the curves intersect. Which classifier performs best?

(c) What do you expect the limiting behavior of learning curves is? That is, if we were able to train on more and more data?

Exercise 3.15 (a) Take your favorite data set and study a learning curve for the first nearest neighbor classier (1NN). What can be expected of the learning curve of the apparent error?

OPTIONAL

Exercise 3.16 Redo the experiments from Exercise 3.14 but substitute `gendath` with `gendats` and `gendatd`. Don't forget to add the 60 noise features.

(a) What kind of typical differences do you see when comparing behaviors for different data sets? Explain these differences?

3.8.2 Dipping

Exercise 3.17 Study the learning curve for `nmc` in combination with `gendatc` for very, very, very small training set sizes.

(a) What seems to be happening? And is this expected?

3.8.3 I Challenge You

Exercise 3.18 This more an challenge than an exercise. You might consider skipping it and move on to the next exercise. Then again, you also might take on this challenge! (Anyway, do keep track of time somewhat.)

(a) Create a two-class problem in which `ldc`, on average, outperforms `qdc` for large sample sizes.

END OPTIONAL

3.8.4 Feature Curves

Like learning curves that typically plot the classification error against the number of training set samples, making feature curves can also be informative. The latter studies how the classification error varies with varying number of feature dimensionality. We can use the `clevalf` command to perform such study.

Exercise 3.19 Load the data set `mfeat_kar` and make a feature curve for 4, 8, 16, 32, and 64 features using 50% of the data for training based on `qdc`.

(a) When redoing this experiment, would you expect the same curve? Why? Or why not?

(b) What, generally, happens to the learning curve when 40% or 80% of the data is used for training? Can you explain this?

3.9 Cross-Validation

The `prcrossval` function allows you to perform a cross-validation on a given data set using a particular classifier (or even a whole bunch of them in one go).

Exercise 3.20 Generate a small data set using `gendatb`, say, with 10 objects per class.

(a) Using n-fold cross-validation, make plots for the error rates for kNN and 1NN over different values of n. Also calculate the standard deviation of the error estimate, e.g., by performing the cross-validation 10 time (check the help).

(b) What do you notice about the estimated error rates? What is the general trend (maybe you should redo the data generation and the cross-validation a couple of times).

(c) What happen to the variance of the estimates for varying n? Again, we are interested in the general trend.

(d) How would the observations change if one would repeat the experiments with much larger dataset? Would they change?

OPTIONAL

Exercise 3.21 **(a)** Let us look back at exercise 1.19 where a Parzen density is estimated on some data, and use the same data in this exercise. Now let PRTools find the optimal h using leave-one-out cross-validation. This can simply be performed by not specifying h, i.e. calling `w = parzenm(+a);`. To find out what value of h the function actually uses, you can call `h = parzenml(a);`. Does the h found correspond to the one you found to be optimal above?

3.10 The Confusion Matrix

A confusion matrix is of the size $C \times C$, where C is the number of classes. An element $C(i, j)$ encodes the confusion between the classes i and j. More precisely, $C(i, j)$ gives the number of objects that are from i, but are labeled as j. Nothing confusing about it.

Exercise 3.22 To create a confusion matrix you may use something like the following code.

```
>> lab = getlab(test_set);
>> w = fisherc(train_set);
>> lab2 = test_set*w*labeld;
>> confmat(lab,lab2)
```

An alternative is:

```
>> confmat(test_set*w)
```

(a) What is stored in `lab` and what does `lab2` contain?

Exercise 3.23 Load the digits data sets `mfeat_zer` and `mfeat_kar`. Split the data up in a training and a test set (using `gendat`).

(a) What are the error rates on the test sets?

(b) Use `confmat` to study the error distributions more closely. Where do the larger parts of the errors stem from?

(c) Can you explain which of the two feature sets is insensitive to image rotations?

Exercise 3.24 Given a confusion matrix $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ for a two-class problem.

(a) Give an estimate of the expected cost (per object) when misclassifying class 1 as class 2 is twice as expensive as the other way around and assuming that a correct classification incurs no cost.

END OPTIONAL

3.11 On to the Next Week...

Exercise 3.25 Make a learning curve for `fisherc` for sample sizes ranging from 1 per class to about 50 per class for `gendats` in 40 dimensions. Is this the learning curve that you had expected? Can you explain it?

Week 4

Complexity

Objectives When you have done the exercises for this week, you

- should know the fundament of the support vector classifier (i.e. maximum margin),
- should be able to kernelize a nearest mean classifier,
- optimise a hyperparameter using crossvalidation.

In order to keep the code text simple, we may assume that we imported `Prtools` as

```
>> from prtools import *
```

4.1 The Support Vector Machine, `svc`

Exercise 4.1 Consider the following 2D two-class data set. Class one contains two points: $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 3 \end{bmatrix}$. Class two has a single data point: $\begin{bmatrix} 2 \\ 0 \end{bmatrix}$.

(a) Determine the classifier that maximizes the margin on this classification problem, using a graphical/geometrical reasoning (probably you cannot do the minimization of the support vector error function by hand). How many support vector are obtained.

Shift the first point above, $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$, to $\begin{bmatrix} 0 \\ -1 \end{bmatrix}$.

(b) How does the new maximum margin classifier look? What happened to the number of support vectors?

Exercise 4.2 **(a)** Demonstrate, possibly graphically/geometrically, that the support vector classifier is sensitive to feature scaling. Hint: this can be done in 2D based on a training set of size 3 (like in 4.1(a) and 4.1(b)) and a single test point.

Exercise 4.3 **(a)** Reconsider the 2D configurations from 4.2 above and compare the solution of the LDA classifier to those obtained by means of an SVM. In what cases do they differ? Do you see the pattern?

OPTIONAL

Exercise 4.4 Make exercise 3.17 in the book.

Section 3.8

END OPTIONAL

4.2 The Nonlinear Support Vector Machine, `svc`

Exercise 4.5 **(a)** Assume we have two objects, represented by 1-dimensional feature vectors x and χ. Find a feature mapping ϕ that leads to the inner product $\exp(-(x-\chi)^2)$. Hints: expand the term $-(x-\chi)^2$ and write $\exp(2x\chi)$ as a series based on the Taylor series of the exponential.

(b) What is the dimensionality of the space that ϕ maps a 1-dimensional feature vector to?

Exercise 4.6 Let's kernelize `nmc`.

(a) Express the distance to any class mean in terms of regular inner products between the test point x and, say, the N_C samples x_i^C from class C.

(b) Kernelize the nearest mean classifier by mean of the Gaussian kernel, $K(x,\chi) = \exp(-\frac{\|x-\chi\|^2}{2\sigma^2})$. Can you show that this boils down to something like a Parzen classifier? You may limit yourself to the two-class case.

Exercise 4.7 The function `svc` can be used to both construct linear and non-linear support vector machines. The following kernels K are defined:

`'linear'`	linear kernel (default)
`'poly'`	polynomial kernel with degree `par`
`'rbf'`	RBF or Gaussian kernel with width `par`

To define the kernel in `svc`, supply a second input argument with a list of kernel type, kernel parameter, and tradeoff parameter C: `svc(a,(kernel_type,par,C))`.

(a) On `a = gendatb([20,20])`, train an `svc` with a `rbf'` kernel, i.e., the Gaussian kernel, for kernel widths that vary from fairly small (0.1?) to fairly large (10?). Check with a large (enough) independent banana test set how the performance varies for the different choices of kernel widths.

(b) How does the kernel width of `parzenc` relate to the width of the radial basis function?

(c) Why can the `svc`, potentially, perform much faster at test time than the Parzen classifier?

Exercise 4.8 The `prcrossval` function allows you to perform a cross-validation on a given data set using a particular classifier. You should supply a dataset, an untrained mapping and, optionally, the number of folds. The function then returns an error estimate for each of the folds:

```
>> e = prcrossval(a,svc([],('rbf',2.0,1.)),k=10)
```

Given the banana dataset with 200 objects per class, `a=gendatb([200,200])`, we want to optimise the hyperparameter of an RBF support vector classifier.

(a) Choose a range of possible hyperparameter-values for the radial basis kernel (something like `s = [0.2,0.5,1.0,2.0,5.0,7.0,10.0,25.0]`?). Estimate for each `s` the crossvalidation error. Which `s` is optimal?

OPTIONAL

Exercise 4.9 **(a)** Use the fact that $(A + BB^T)^{-1}B = A^{-1}B(I + B^TA^{-1}B)^{-1}$ to show that $X^T(XX^T + \lambda I_N)^{-1} = (X^TX + \lambda I_d)^{-1}X^T$, with I_a the $a \times a$ identity matrix.

(b) Using the identity from 4.9(a), show that estimating the y to an unobserved x through ridge regression can be expressed completely in terms of inner products between input vectors and values from Y.

Define $k(x, z) = (x^Tz + c)^2$. We will see that k is a kernel by explicitly finding the mapping ϕ that takes the feature vectors x and z to a new space where the inner product equal $k(x, y)$, i.e., you will find ϕ such that $\phi(x)^T\phi(z) = k(x, z)$.

(c) First take x and z to be 1-dimensional and write out/expand $k(x, y)$.

(d) Do the same for 2-dimensional x and z.

(e) See the pattern.

Unfortunately, there is no kernelized ridge regression in PRTools. So, for the next experiment, you have to implement it yourself. The function `proxm` could come in handy.

(f) You may want to check that kernel ridge with a very small λ and the above kernel k (with c fixed) basically solves the problem in 2.7(e). (But maybe you should not do the experiment with too large a training set...)

(g) Why does the choice of c does basically not matter in the foregoing experiment as long as it is not set to 0 (and one has enough training samples)?

4.3 Bias and Variance in Regression

Exercise 4.10 This is a restatement and continuation of Exercise 2.5.

Using `gendatr` or `prdataset`, generate data where the inputs x are drawn uniformly from the interval $[0, 1]$ and the corresponding outputs y are obtained by squaring this value and adding Gaussian noise to the inputs: $y = x^2 + \varepsilon$ with ε a random sample from the standard normal distribution.

(a) Study the behavior of polynomials of degree 0 to 3 for different training set sizes (e.g. 4, 40, and 400 samples?). You may want to have a look at the data and the fitted polynomial models. You can also estimate the squared error using a somewhat large test set.

(b) Estimate the bias and the variance of the prediction using the four different polynomial regressors and a couple of different training set sizes. Compare these outcomes, both across degree and across training set sizes. How do the bias and variance change? Did you expect this in the light of the the bias-variance tradeoff?

Exercise 4.11 This is a restatement and continuation of Exercise 3.3.

We consider a regression data set where the inputs x are drawn uniformly from the interval $[0, 1]$. The corresponding outputs y are obtained by adding Gaussian noise to the inputs: $y = x + \varepsilon$ with ε a random sample from the standard normal distribution. You can generate regression data sets by means of `gendatr` or `prdataset`. Ridge regression is carried out using `ridger`.

(a) Create a training data set of size 2 and a large test data set (1,000 or 10,000 samples will do). Study the regression fit to the training data for different amounts of regularization (using command like `scatterr` and `plotr`). Also check how the squared error varies for different λs (using commends like `testr`). Check a couple of new draws for the training set and also try a different set size of, for instance, of 100. Try regularization parameters varying from 0 (or something smallish like 10^{-3}) to something larger like 10^3.

(b) Determine (roughly) which value for the regularization parameter gives, on average, the best performance. Determine this optimum for a training set size of 2, of 10, and for a training set size of 100. You can limit your search to λs in the range $[10^{-3}, 10^3]$.

(c) Estimate the bias and the variance of the prediction using the optimal regularization parameters for the three training set sizes and compare their values to the ones obtained by the unregularized solution.

4.4 Bias and Variance in Classification

Exercise 4.12 Generate a small Banana data set, using `gendatb`, say, with 10 samples per class. Let us use a linear classifier on this data set, for instance, LDA, logistic regression, or the NMC.

(a) Determine which parts of this data set are as good as always misclassified. Is this a manifestation of the bias or the variance?

(b) Determine for which parts of the feature space the classifiers often disagree. Is this a manifestation of the bias or the variance?

END OPTIONAL

Week 5

Probabilistic Models & Clustering

Objectives for this week:

- be able to formulate ML and MAP estimators;
- understand how to come to an expression for the posterior predictive distribution;
- know how to check for (conditional) (in)dependencies based on simple Bayesian networks;
- to learn to use hierarchical clustering on several datasets;
- to learn about the limitations of hierarchical clustering;
- to get familiar with mixture-of-Gaussian clustering;
- to learn how to use cluster validity measures.

In what follows, p is used for pmfs as well as pdfs. Likewise, the term probability may also refer to a value that is actually a density. This should not lead to any (real) confusion. Inform us, however, if you think it does. Admittedly, we generally may use somewhat sloppy notation. E.g. we may not be talking in terms of random variables there where it might actually be more appropriate to do so. In a similar vein, we often just talk about the distributions $p(a)$ and $p(\mu, b)$ over a and (μ, b), respectively, without making very explicit, for instance, that these are of course different p.

5.1 Bayes Rule Recap

You want to know if you are one of the few people that knows all there is to know about machine learning. To find out, you decide to go through a host of exams and other tests. The overall score you receive indicates that you are indeed one of the happy(?) few. Knowing so much about ML, however, you do not jump to conclusions and you check with your local expert how accurate these scores are. "Well," the person in the lab coat tells you, "of course they are not perfect, but these test will correctly identify 99% of the people that know all of ML and will be incorrect in a mere 0.1% when people do not belong to this select group."

Exercise 5.1 **(a)** So, what is the probability that you do indeed know all of ML?

5.2 Maximum Likelihood and A Posteriori Estimation

Exercise 5.2 Given N i.i.d. observations x_i, with $i \in \{1, \dots, N\}$, from a true distribution p (e.g. a pdf or a pmf). Assume we want to fit a distributional model $p(\cdot|\theta)$ parameterized through θ for such observations. So, $p(\cdot|\theta)$ is a distribution for every choice of θ, which takes x_i's as input variable.

(a) Consider a fixed parameter value θ and assume that this is the correct model. Write down the probability (or, in the continuous setting, the probability density) of observing the sample of N x_i's given this parameter value.

(b) How is the solution to the previous question also referred to if we consider it as a function of the parameter?

Exercise 5.3 The maximizer of the likelihood, gives us the maximum likelihood estimator for the parameter. You could say that this parameter choice maximizes the probability[1] of observing our i.i.d. training data set $\{x_1, \dots, x_N\}$ among all possible parameter choices. Another way to come to an estimate for our parameter θ is to aim for the most probable choice of parameter given the data.

(a) Use Bayes' theorem to write the probability of a specific parameter θ, given the observed training data, in terms of the likelihood of θ.

The probability distribution over the parameter space that comes in through Bayes' theorem, gives us the possibility to encode our a priori expectations about the possible solution. Once we have decided on this so-called prior, we find another estimate of our parameter by maximizing the probability of θ, given the observed training data. This estimator is called the maximum a posteriori estimator or MAP estimator for short.

(b) When maximizing the a posteriori probability, why do we not need to bother ourselves with the data marginal $\prod p(x_i)$?

(c) What solution does the MAP estimator give us if we assume $p(\theta)$ to be constant, i.e., we assume a so-called flat prior?

(d) Say we are not only interested in the optimal estimator, but we also care about the actual probability $p(\theta|x_1, \dots, x_N)$. Assuming $p(x|\theta)$ to be the true model for an observation x, can you come to an expression for $\prod p(x_i)$ in terms of θ, among other terms? Hint: from the joint distribution over x and θ, we can of course derive any marginal...

5.3 Missing Data

There are many classification settings in which one has both labeled data and data for which only the input feature vector has been observed and not the corresponding labels. This situation may especially arise in the case that labeling one's data is relatively expensive compared to collecting the inputs.

OPTIONAL

[1]Though in the continuous setting, these are not probabilities of course.

Exercise 5.4 **(a)** We want to fit a model $p(x, y|w)$. Give the log-likelihood of a single labeled observation (x_i, y_i) and a single unlabeled observation x_j.

Here's a taste of what the clustering lectures will have in store for us (see Section 5.13).

(b) Say our model is QDA, show that if we only have unlabeled data available, the log of the model likelihood on the training data is equal to the objective function used when fitting a mixture of Gaussians.

(c) What is (probably) the most-used technique to estimate the parameters of such a model with so-called unobserved latent variables? How would you get to the likelihood estimates?

END OPTIONAL

5.4 Some MAP Examples

Exercise 5.5 In a two-class problem, labels $y \in \{-1, +1\}$ can be modeled through a Bernoulli distribution. Let the parameter q be the probability of observing $y = -1$. Assume the a priori distribution for q equals $p(q) = q^{-1}(1-q)^{-1}$.

(a) Determine the maximum a posteriori probability for q given that we observed one $+1$ and one -1.

(b) The prior used is referred to as the Haldane prior. It is a so-called improper prior. Which properties of a pdf does this improper prior fulfill and which doesn't it?

OPTIONAL

Exercise 5.6 Let us a priori assume that the mean m of a 1D distribution that we want to fit is within an interval $[-a, +a]$ around the origin. Within this interval, the probability for every mean is equal.

(a) What kind of distribution do we have on $[-a, a]$?

(b) Let us assume in addition that the 1D model distribution that we are fitting is normal and that there is no prior assumption on the variance (or, equivalently, that the prior on the variance is flat). Given N observations $x_1, \ldots, x_N$ from $\mathbb{R}$, determine the MAP estimates for the mean and the variance. (No need to derive this very formally and precise, but your solution should be correct of course and you should be able to properly defend it.)

(c) Let us now assume that the 1D model distribution that we are fitting is actually uniform between l and u. There is no prior assumption on any of the other parameters except for the mean. Given N observations $x_1, \ldots, x_N$ from $\mathbb{R}$, determine the MAP estimates for l and u.

END OPTIONAL

Exercise 5.7 Consider a two-class classification problem in a feature space of one dimension. Let N_i be the number of observations from class i. Let (x_i, y_i) $(i \in \{1, \dots, N\})$ be $N = N_1 + N_2$ pairs of observations, i.e., we have N different training samples with feature value x_i and corresponding class label y_i. Assume now we want to train a linear discriminant classifier (LDA) that assumes Gaussian class-conditional distributions for which the two variances (and standard deviations) are equal. That is, we model the classes by normal distributions with equal (co)variance structure.

(a) Write down the full density $p(x, y)$ for this two-class Gaussian model.

(b) Use the logarithmic loss to write down the (maximum) likelihood risk functional for this model. Indicate clearly which parameters have to be estimated, i.e., how are the two priors, the two means, and the single standard deviation denoted?

(c) Write down the expression for the empirical likelihood risk functional for the two-class Gaussian model and express it in the N different training samples (x_i, y_i) $(i \in \{1, \dots, N\})$.

(d) Derive the maximum likelihood estimates for the priors, means, and variances for this model. The derivation doesn't have to be given in every single detail, but should contain sufficient formulas and/or explanatory text to be able to follow the proof.

Assume now, we have some prior knowledge about the mean of class 1 that we can encode by means of a prior over this variable. Assume this prior to be a Gaussian distribution with zero mean and variance λ^2, i.e., we expect the mean of class one to be near the origin.

(e) Given again the N observations, write down the expression for the posterior probability of the five parameters considered earlier. You may want to use Bayes theorem and the earlier obtained expression for the (empirical) likelihood risk functional in order to come to an expression for this posterior.

(f) Determine the MAP solution for this posterior probability.

(g) What happens in the extreme cases that λ goes to either 0 or ∞?

(h) Consider the following additional extreme cases: $\lim_{N_1 \to \infty}$, $\lim_{N_1 \downarrow 0}$, $\lim_{s \downarrow 0}$ (where in the last expression s denotes the standard deviation of the Gaussian classes).

(i) For which of the above five extreme cases, do we find the ML estimate back? In which cases does the prior decide on the optimal MAP solution?

Exercise 5.8 In case we are dealing with a 1D scale parameter s, like the standard deviation in a normal distribution, a standard prior to use for this parameter is $\frac{1}{s}$.

(a) Why is this not a proper prior?

(b) Will the MAP estimator for such scale parameter typically be larger or smaller than the ML estimator, or will they be exactly the same?

5.5 That Bias and Variance Again

Exercise 5.9 (a) Go though the above examples once again. Does the prior typically increase or decrease the bias of the model or is this difficult to say? If the answer is the last option, why is it difficult to say? Ask yourself the same about the variance?

(b) Can we see these priors as regularizers?

5.6 Regularization and MAP

Let us relate some regularizers to specific forms of MAP estimation.

Exercise 5.10 **(a)** Reconsider the solution found in 2.10(e). With which form of L_2 regularized linear regression do we get to exactly the same solution?

(b) Consider standard multivariate linear regression with squared Euclidean norm regularization as in 3.1. What prior should one assume over w to let the MAP solution coincide with the minimizer of Equation (3.1)?

(c) With what prior do we get to the lasso as discussed in Section 3.3?

5.7 The Predictive Distribution

When doing actual predictions, especially in the Bayesian setting, one typically is not interested in the estimates of the parameters of a distribution as such. In the regression setting, for instance, one rather cares about the output value for a particular input given all of the observed data.

Exercise 5.11 A random variable Y takes on the value $+1$ with probability θ and -1 with probability $1 - \theta$. A random draw lead us to make the observation $Y = +1$.

(a) Determine the maximum likelihood estimate for θ.

(b) Take the a prior probability for θ to be uniform on $[0, 1]$, determine its MAP estimate.

(c) Assume Y^* is an i.i.d. draw from the same distribution as Y. Determine the predictive distribution for $p(Y^*|Y = +1)$.

We are going to take a new (i.i.d.) draw Y^* from the same distribution as Y. The idea is, however, that we should guess at the outcome of this draw beforehand.

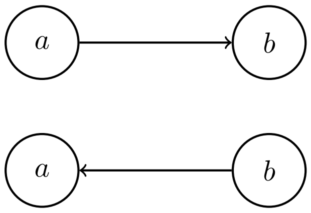
(d) Assume the loss to be minimized with our guess is the error rate (or the 0-1 loss), what label should we guess for this new draw according to the maximum likelihood estimate, i.e., which guess minimizes the expected error? What would our guess be if we rely on the MAP estimate? And what if we use the predictive distribution?

(e) Assume guessing $+1$, while the actual draw will give -1, costs us 1, while the cost of make the other mistake, guessing -1, while the actual draw will give $+1$, is a nonnegative variable c. Relying on the predictive distribution, for which cost c does it become beneficial to guess -1? At this same cost, what guess would we prefer according to the ML and MAP?

(f) What guess should we do for Y^* in case, the loss considered is the squared loss? Again, go through the above three scenarios: ML, MAP, and predictive distribution. Note that, a priori, we are of course not limited to choosing -1 or $+1$.

5.8 Some Bayesian Network Basics

Exercise 5.12 Is there an essential difference between the dependence of the two variables a and b as defined by the two Bayesian networks below?



Exercise 5.13 (a) With three nodes, how many essentially different DAGs can be constructed?

(b) With three different random variables, how many essentially different (in the sense of conditionally independence relations) Bayesian networks can be constructed?

(c) What if all three random variables are basically the same, what is the number of really differently behaving Bayesian networks in the case?

OPTIONAL

Exercise 5.14 Given 2 variables for which we cannot make any independence assumptions. In this case, we can make 2 different decompositions of the joint distribution in terms of individual distributions (i.e., distributions for single variables that can, however, be conditioned on any number of other variables). In particular, we have $p(x_1)p(x_2|x_1)$ and $p(x_2)p(x_1|p_2)$.

(a) Given 3 variables for which we cannot make any independence assumptions, how many different decompositions does the joint distribution allow in terms of individual distributions?

(b) Can you come up with the general solution for N variables?

Exercise 5.15 Given a Bayesian network for the variables x_i, $i \in \{1, \ldots, N\}$. Denote by $P(x_k)$ the subset of variables that are parents of x_k. Define the product of conditional probability density functions

$$\prod_{i=1}^{N} p(x_i|P(x_i)).$$

Show that this is a probability density function.

END OPTIONAL

Exercise 5.16 Assuming that $p(a|bc) = p(a|b)$, show that $p(c|ab) = p(c|b)$.

Exercise 5.17 Consider a problem with three variables a, b, and c and discrete observations.

(a) Show by constructing a counter example that $p(a,b) = p(a)p(b)$ does not generally imply that $p(a,b|c) = p(a|c)p(b|c)$.

(b) Also show that the implication in the other direction generally does not hold. Again, try to do so by means of a simple counterexample.

Exercise 5.18 We are given the following DAG.

$$\begin{tikzcd}
a \arrow[r] & b \arrow[d] & c \arrow[l] \\
 & d &
\end{tikzcd}$$

(a) Assume $X, Y, Z \in \{a, b, c, d\}$ and X, Y, and Z different from each other. Suppose we want to consider paths between X and Y. How many possible, essentially different, graphs do we have to consider?

(b) For which of the graphs of X, Y, and Z are all paths from X to Y blocked by Z?

(c) For which of the graphs do we have that $X \perp Y|Z$?

(d) Assume now that there is also an arrow from c to d. How do the answers to the above questions change?

OPTIONAL

Exercise 5.19 = Exercise 2.39 from *Pattern Recognition* (fourth edition) by Theodoridis and Koutroumbas.

END OPTIONAL

5.9 Inference in Bayesian Networks

Exercise 5.20 The a priori probability $p(s = 1)$ that Student X studies (properly) for CS4220 is 0.9. The prior probability $p(l = 1)$ that the s/he attends the lectures equals 0.8. In addition, for passing the exam ($e = 1$), we have the following conditional probabilities.

$$\begin{aligned}
p(e = 1|s = 1, l = 1) &= 0.9 \\
p(e = 1|s = 1, l = 0) &= 0.7 \\
p(e = 1|s = 0, l = 1) &= 0.2 \\
p(e = 1|s = 0, l = 0) &= 0.1
\end{aligned}$$

The corresponding DAG is as follows.

$$\begin{tikzcd} s \arrow[r] & e & l \arrow[l] \end{tikzcd}$$

(a) Determine $p(e = 1)$, i.e., the a priori probability of passing the exam.

(b) Calculate $p(e = 1|l = 1)$ and $p(l = 1|e = 1)$.

(c) What value does $p(l = 1|e = 1, s = 1)$ take on?

(d) Given that somebody passed the exam, is it more likely that the person studied (properly) or that s/he attended the lectures? What is the difference in probability?

OPTIONAL

Exercise 5.21 = Exercise 2.41 from *Pattern Recognition* (fourth edition) by Theodoridis and Koutroumbas.

END OPTIONAL

5.10 Bayesian Networks and Learners

Exercise 5.22 Consider a naive Bayes classifier and the independence assumptions it makes about the features and the labels. Draw the Bayesian network that reflects these assumptions.

Exercise 5.23 **(a)** Given that the independence assumption made in a particular Bayesian network are indeed correct, why would this typically lead to improved performance over a statistical model that does not make these assumptions?

(b) Even if the assumptions are only correct to some degree, why can they help in getting to improved performance?

Let us now compare `ldc` and a naive Bayes version of `ldc` that treats all features as independent given the class label. (The latter you may have to implement yourself.)

(c) Generate a data set in 10 dimensions of size 5000 using `gendats` and compare the learning curves for 10, 20, 30, . . . , 100 training samples of `ldc` and its naive version. Qualitatively explain their behavior and their performance differences.

(d) How do those learning curves look if `gendatd` is used rather than `gendats`? Explain the differences we now see.

OPTIONAL

Exercise 5.24 Consider the polynomial regression setting in Subsection 8.1.1 of Bishop. Specifically, look on page 264 in Figure 8.6.

(a) What does it mean that x_n is modeled with a small, solid dot?

(b) What would it mean if these x_n are modeled by means of a shaded node?

(c) Can you come up with situations where one would consider these x_n to be deterministic and situations where these are stochastic?

(d) In most classification settings one considers, would the input feature vector typically be taken stochastic or deterministic?

Exercise 5.25 Draw a Bayesian network for LDA in which both its parameters, the feature and label data (say, N samples to train on), and their relations are shown. Are there any (conditional) independency assumptions that can be made?

END OPTIONAL

5.11 Clustering

11 In the rest of this week's exercises, we will discuss the problem of clustering; this practical session is intended to familiarise you with the different techniques that were discussed, more specifically hierarchical clustering, the mixtures-of-Gaussians and clustering evaluation.

5.12 Hierarchical clustering

This week we will focus on which *objects* belong together in groups or clusters. This clustering process enables us to extract structure from the data, and to exploit this structure for various purposes such as building a classifier or creating a taxonomy of objects.

The most difficult part of clustering is to determine whether there is *truly* any structure present in the data and if so, what this structure is. To this end, we will also employ cluster validity measures to estimate the quality of the clustering we have obtained.

13 In the lectures we discussed hierarchical clustering at length. There are basically two choices that need to be made in hierarchical clustering in order to construct a dendrogram:

1. the dissimilarity measure;
2. the type of linkage.

In this course, we will only employ the Euclidean distance between two samples as a measure of dissimilarity. As you will recall from the lecture, there are three types of linkage: complete, single and average linkage. Once the dendrogram has been constructed, we need to cut the dendrogram at an appropriate level to obtain a clustering.

Exercise 5.26 Start with the `hall` dataset, an artificial dataset with clear structure. This dataset can be loaded into a `prdataset` by using `read_mat("hall")`.

(a) Load the dataset and use `scatterd` to visualise it. How many clusters are visible in the plot?

(b) What is the most suitable clustering?

Exercise 5.27 Load the `rnd` dataset and make a scatterplot to visualise it. This is a uniformly distributed dataset, with no apparent cluster structure. We will hierarchically cluster this dataset to get an idea of what a dendrogram looks like when there is no structure in the data.

(a) Plot the dendrogram with complete linkage using `dendro(+a, "complete")`. What is apparent?

(b) Perform hierarchical clustering with `hclust` on the `rnd` dataset with complete linkage. The function `hclust` is a mapping that can be trained on dataset `a` using complete linkage, to get 3 clusters, by doing:

```
lab = hclust(a,(3,'complete'))
```

You can now relabel the original data with the new labels, and plot:

```
b = prdataset(+a,lab)
scatterd(b)
```

(c) Repeat this for single and average linkage. Do you observe the same behavior as with complete linkage?

Exercise 5.28 **(a)** Perform hierarchical clustering on the `hall` dataset with complete linkage: what do the lengths of the vertical stems in the dendrogram tell us about the clustering?

(b) Cut the dendrogram at different levels, i.e. experiment with different numbers of clusters when calling the `hclust`. Can you think of ways in which a good clustering can be defined?

(c) Can you devise a simple rule-of-thumb (in terms of vertical stem lengths) for finding a good clustering in the dendrogram?

(d) Now perform single linkage hierarchical clustering. Do you notice any significant differences with the complete linkage dendrogram?

(e) Do you notice any differences with the average linkage dendrogram?

5.13 Clustering with a mixture-of-Gaussians

During the lectures, the concept of clustering based on the quality of a mixture-of-Gaussian density fit to the data was discussed. The operation of the Expectation-Maximisation (EM) 14.2 algorithm, which is employed to estimate the parameters of the mixture model, was also discussed in detail.

In the following exercise, mixture model clustering will be explored with the aid of the `mog` function. This function performs the Expectation-Maximization procedure to find the parameters of a Mixture of Gaussians, trained on some dataset `a`:

```
w = mog(a,(3,'full',0.001))
scatterd(a)
plotm(w)
```

The function `mog` has three input parameters: (1) `k` for the number of clusters, (2) the shape of the covariance matrix of each of the clusters, (3) a regularisation parameter that regularises the inverse covariance matrix.

When you fitted the mixture, the trained mapping outputs a probability density for each of the mixture components. So if you ask for `k=3` clusters, each input object `x`, will return a vector of 3 values. You can now compute the (log-)likelihood of some dataset by first summing the probabilities of all clusters, then take the logarithm, and then add all log-probabilities of the full dataset:

```
pred = a*w
logL = numpy.sum(numpy.log(numpy.sum(+pred,axis=1)))
print(logL)
```

Exercise 5.29 **(a)** Load the `triclust` dataset and play around with the function `mog`. Vary the number of Gaussians employed in the mixture model, and also vary the type of Gaussian employed. Relate the `type` (`'full'`,`'diag'`,`'sphr'`) of the Gaussian to its covariance matrix.

(b) On the `cigars` dataset, fit an unconstrained Gaussian (`type = 'full'`) mixture model using the function `mog`. For the number of clusters k, assume the following values: $k = 1, 2, 3, 4, 5$. Which k do you expect to be the best? .

(c) Now try clustering the `messy` dataset. What is the best shape to employ for the Gaussians? What is the optimal number of clusters?

5.14 Cluster validation

This part of the session is intended to familiarise you with different cluster validity measures.
16 We will achieve this by:

1. employing the fusion graph as a simple, intuitive measure of clustering tendency in hierarchical clustering;

2. coding and applying the Davies-Bouldin index to various algorithms and datasets.

5.14.1 Fusion graphs

The *fusion graph* plots the *fusion level* as a function of the number of clusters (g). For example, the fusion level at $g = 2$ represents the (single, complete, average) link distance between the clusters that are merged to create two clusters from three clusters. A simple heuristic to determine the number of clusters in hierarchical clustering is to cut the dendrogram at the point where we observe a large jump in the fusion graph.

Exercise 5.30 **(a)** Why is this a reasonable heuristic to employ?

The following three exercises focus on the estimation of the number of clusters based on the fusion graph.

Exercise 5.31 **(a)** Load the `triclust` dataset. Perform single linkage hierarchical clustering and display its fusion graph by using the function: `fusion_graph`. Where do you observe the largest jump?

(b) Now perform complete linkage hierarchical clustering. Does the fusion graph give a clear indication of the number of clusters in the data? If not, why not?

Exercise 5.32 **(a)** Load the `hall` dataset. Perform single linkage hierarchical clustering and display the fusion graph. What do you observe in the fusion graph?

Exercise 5.33 **(a)** Finally, load the `messy` dataset. Perform single linkage hierarchical clustering. According to the fusion graph, where should the dendrogram be cut?

(b) Does a satisfactory clustering result from cutting the dendrogram at this point? Motivate.

(c) Now perform complete linkage clustering. Is the clustering suggested by the fusion graph better or worse than the clustering obtained with single linkage clustering?

5.14.2 The Davies-Bouldin index

D.L. Davies and D.W. Bouldin[2] introduced a cluster separation measure which is based on both the within-scatter of the clusters in a given clustering and the separation between the clusters. Formally, this measure is known as the Davies-Bouldin index (DBI). It assumes that clusters are spherical, and that a desirable clustering consists of compact clusters that are well-separated.

Suppose we wish to compute the DBI for a clustering consisting of n objects assigned to g clusters. We can compute a score for every possible pair of clusters in this clustering, which is inversely proportional to the distance between the cluster means and directly proportional to the sum of the within-scatters in the pair of clusters. This score is given by

$$R_{jk} = \frac{\sigma_j + \sigma_k}{\|\boldsymbol{\mu}_j - \boldsymbol{\mu}_k\|}, \quad j,k = 1,2,\ldots,g; \quad k \neq j. \tag{5.1}$$

Here $\boldsymbol{\mu}_j$ is the mean of all the objects in cluster j and σ_j is the within scatter of cluster j, given by:

$$\sigma_j = \sqrt{\frac{1}{n_j} \sum_{\boldsymbol{x}_i \in C_j} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2}, \tag{5.2}$$

where C_j is the set of objects associated with cluster j and n_j is the number of objects in cluster j. The score, R_{jk}, is small when the means of clusters j and k are far apart and the sum of the within-scatter for these clusters is small. Since cluster j can be paired with $g-1$ other clusters, resulting in $g-1$ pair-scores, $R_{jk}, j = 1,2,\ldots,g; k \neq j$, a *conservative* estimate of the cluster score for cluster j, when paired with all other clusters, is obtained by assigning the maximal pair-score with cluster j:

$$R_j = \max_{k=1,2,\ldots,g;\ k\neq j} R_{jk}. \tag{5.3}$$

The Davies-Bouldin index of the complete clustering is then determined by averaging these maximal pair-scores for all clusters:

$$I_{DB} = \frac{1}{g}\sum_{j=1}^{g} R_j. \tag{5.4}$$

[2] IEEE Transactions on Pattern Analysis and Machine Intelligence 1, pp. 224–227, 1979.

Exercise 5.34 We will employ the Davies-Bouldin index to evaluate the clustering produced by hierarchical clustering. We will do so for a range of clusters in order to determine the optimal number of clusters, i.e. the best level to cut the dendrogram at. To achieve this we employ the function `dbi`.

(a) The function `dbi` takes the dataset features and the labels predicted by the clustering method as inputs. It computes, for each clustering, the DBI. Familiarize yourself with the operation of this function.

(b) Load the `triclust` dataset and make a scatterplot. What do you expect the DBI values as a function of the number of clusters to look like?

(c) Now apply `dbi` to this dataset with complete linkage clustering, starting at 2 clusters and stopping at 10. What is evident from the DBI values?

(d) Apply `dbi` to the `hall` dataset with complete linkage clustering, starting at 2 clusters and stopping at 20. What do you observe? Can you explain your observation?

Week 6

Reducing and Combining

These exercises are meant to give you both some practice with the material covered in the lectures and the literature and an impression of what you are expected to know. Judge for yourself which exercises you need to, want to, or should practice. On another note, at this point, the lecturer (ML) cannot claim that all exercises are thoroughly checked. If you think something is wrong, unclear, etc., let us (i.e., me (ML) , any of the other lecturers, or any of the TAs) know.

Objectives When you have done the exercises for this week and went through the related reading material, you should

- be able to do some basic combinatorial computations,
- appreciate the computational complexity of feature selection,
- understand the approximative nature of and the interaction between the selection criteria and the search strategy,
- know some of the basic properties of scatter matrices,
- be able to reproduce and interpret the objective functions of LDA and PCA,
- know how to kernelize PCA,
- appreciate the reasons why one may want to combine,
- be able to explain how bagging is carried out,
- understand the underlying idea of boosted

6.1 Some Combinatorics

Exercise 6.1 Given a data set with 100 features. Say you want to find the 5 best features. How many feature sets do you need to check for this in case we have to rely on an exhaustive search?

6.2 Stuff on Scatter Matrices

Exercise 6.2 Given a set of feature vectors $\{x_i\}$ for which the sample covariance matrix equals C and given a linear transformation A. Show that the covariance matrix for the transformed feature vectors $\{Ax_i\}$ equals ACA^T.

OPTIONAL

Exercise 6.3 Note: this might be a tiresome exercise but if you have too much time on your hands... One way or the other, you should know that $S_m = S_b + S_w$ for a fact!

(a) Show that $\Sigma_i = E[(x - \mu_i)(x - \mu_i)^T] = \frac{1}{2}E[(x - x')(x - x')^T]$ in which x and x' are equally distributed, i.e., according to the distribution of class i.

(b) Exercise 5.12 Ex. 5.12 from the book. Show that the mixture scatter matrix is the sum of the within-class and between-class scatter matrices. The fact proven in (a) can be very helpful here.

Exercise 6.4 **(a)** Variation to Exercise 5.18 from *Pattern Recognition* by Theodoridis and Koutroumbas (where the book is slightly inaccurate). Convince yourself that if the number of classes equals M that the rank of the between scatter matrix S_b is *at most* $M - 1$. You can either try to proof this, which might be a bit though, or you can develop some insight be staring at the formula or doing some experiments in, say, Matlab.

(b) In which (two) different cases can the rank of S_b be smaller than $M - 1$? Note that this applies similarly to any estimate of a covariance matrix based on M samples.

Exercise 6.5 Make Exercise 5.20 from *Pattern Recognition* by Theodoridis and Koutroumbas. (Note that S_{xw} actually equals S_w?)

Exercise 6.6 **(a)** (Another variation to Exercise 5.18 from *Pattern Recognition* by Theodoridis and Koutroumbas.) Convince yourself that if the number of classes equals M that the rank of the between scatter matrix S_b is *at most* $M - 1$. You can either try to proof this, which might be a bit though, or you can develop some insight be staring at the formula or doing some experiments in, say, Matlab.

(b) In which (two) different cases can the rank of S_b be smaller than $M - 1$? Note that this applies similarly to any estimate of a covariance matrix based on M samples.

END OPTIONAL

6.3 Supervised Feature Extraction: the Fisher Mapping

Exercise 6.7 **(a)** Consider three arbitrary points in a 2-dimensional feature space. One is from one class and the two others from the other. That is, we have a two-class problem. What (obviously?) is the 1-dimensional subspace that is optimal according to the Fisher mapping/LDA? What value would the Fisher criterion take on in this case? Explain your answers.

(b) Consider an arbitrary two-class problem in three dimensions with three points in one class and one in the other. How can one determine a 1-dimensional subspace in this 3D space that optimizes the Fisher criterion? (Like in (a), a geometric view maybe seems the easiest here.)

(c) Again we take a two-class problem with four objects in 3D but now both classes contain the same number of points. Again determine the 1D subspace that Fisher gives.

(d) When dealing with two points from two different classes in three dimensions, how many essentially different Fisher optimal 1D subspaces are there?

6.4 PCA and Fisher

Exercise 6.8 Assume we have a d-dimensional data set for which we have a covariance matrix C.

(a) Say that we map all the data point linearly to 1 dimension by mean of the d-vector v. Show that the data variance in this 1D space equals $v^T Cv$.

(b) Assume that v has norm 1, i.e., $||v|| = 1$. Show that the v maximizing $v^T Cv$

OPTIONAL

Exercise 6.9 Assume that the overall mean of a data set is zero and that all feature vectors are stored in the rows of the matrix X.

(a) Show that the eigenvector of $X^T X$ with the largest eigenvalue is equal to the largest principal component.

(b) Given that v is a (column) eigenvector with eigenvalue λ of XX^T, show that $X^T v$ is an eigenvector of $X^T X$. What is the eigenvalue associated to the eigenvector $X^T v$?

(c) Show that a (row) feature vector x from the original space can be mapped onto the first PC by using $xX^T v$.

(d) Describe now how one can kernelize PCA. That is, describe the procedure to do the actual calculation. All ingredients are there. (Hint: realize what information XX^T and xX^T contain.)

Exercise 6.10 Consider 1000 samples from a 3D Gaussian distribution. Assume this sample has zero mean and a 3×3-covariance matrix C given by

$$C = \begin{pmatrix} 99 & 0 & 0 \\ 0 & 99 & 0 \\ 0 & 0 & 124 \end{pmatrix} .$$

(a) Which direction gives the first principle component for this data set?

It turns out, that these 1000 samples are in fact the class means of 1000 different classes of bird species described by three different features. Assume that all class priors are $\frac{1}{1000}$ and that all these classes are normally distributed with covariance matrix equal to the identity matrix.

(b) Give the total covariance matrix (also called the mixture scatter matrix) for this data set.

(c) The Fisher mapping determines the Eigenvectors with the largest Eigenvalues of the matrix $\Sigma_W^{-1}\Sigma_T$ (or, if you prefer, $\Sigma_W^{-1}\Sigma_B$.) Σ_W, Σ_B, and Σ_T are the within-class covariance, the between-class, and the total covariance matrices, respectively. Which direction gives the optimal Fisher mapping if we want to reduce the feature space to one dimension?

In a next step, a linear feature transformation T is applied to the original 3D space. The 3×3-matrix describing this transformation is given by

$$T = \begin{pmatrix} 1 & \frac{1}{2} & 0 \\ \frac{1}{2} & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} .$$

(d) Recall your answer under b. and show that the total covariance matrix of the data after the transformation equals

$$\begin{pmatrix} 125 & 100 & 0 \\ 100 & 125 & 0 \\ 0 & 0 & 125 \end{pmatrix} .$$

(e) Are the first two features correlated and, if so, are they positively or negatively correlated?

(f) Determine the first principal component for this new feature space.

(g) Let v the Fisher vector you found under exercise c. How can you calculate the optimal Fisher mapping in Matlab for the transformed data set, given you can use standard functions from Matlab, but only v and T as arguments?

Exercise 6.11 Consider the following two-class problem in three dimensions. Class one has a normal distribution with unit covariance matrix centered in the origin. Class two lies symmetrically around the first class and consists of two normal distributions (a mixture of Gaussians if you like) of which the two means are located in $(-3, -3, 2)$ and $(3, 3, -2)$. The classes have equal priors. We are going to study dimensionality reductions for this classification problem.

(a) Consider the Fisher mapping (`fisherm`). Give a formula for the Fisher criterion that this procedure optimizes. Make sure that all variables etc. are properly clarified. Explain what this criterion aims to do and how it works.

(b) Explain why you would expect to find better subspaces with the Fisher mapping in case you are dealing with small training set sizes rather then large training set sizes.

(c) What can you say about the subspaces that you will find in case the training set is very, very large? Is it a good subspace or a bad subspace and why is this so?

(d) How would PCA work under different sizes of training sets on this data set? Would it work well in comparison to the Fisher mapping? Explain your answer.

(e) For a particular training set, the covariance matrix has the form

$$\begin{pmatrix} 3 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 3 \end{pmatrix}$$

Give the direction, i.e., a 3D vector, of the first principal component.

Exercise 6.12 Let us consider a two-dimensional, two-class problem in which the two classes are normally (Gaussian) distributed with both covariance matrices equal to $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ and with equal class priors. The two class means are given by $\left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} 2 \\ 0 \end{smallmatrix}\right)$, respectively. We are going to qualitatively study the learning curve of the true error of a classification procedure that first extracts a single feature from a sample of this data set using PCA and subsequently builds a nearest mean classifier (NMC) in this 1-dimensional space.

(a) Show that the total covariance matrix, which is used in PCA, for this data set equals $\left(\begin{smallmatrix} 2 & 0 \\ 0 & 1 \end{smallmatrix}\right)$.

(b) Show that $\left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right)$ is the first principle component of the total covariance matrix. Show that $\left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right)$ is the second.

(c) If we reduce the original data by means of the first principle component as given in b., does the Bayes error increase, decrease, or stay exactly the same?

(d) What happens if we estimate the covariance matrices and calculate the first principle component based on a small training set? How does the expected true error, using PCA for feature extraction and NMC for classification, for the small training set compare to the true error rate for a very large training data set?

(e) How does the learning curve for this problem with the above classification procedure look qualitatively?

(f) How does the learning curve look qualitatively if the class covariance matrices equal $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 4 \end{smallmatrix}\right)$?

(g) How does the learning curve look qualitatively if the class covariance matrices equal $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 2 \end{smallmatrix}\right)$?

END OPTIONAL

Exercise 6.13 We have a 4-class problem in 3 dimensions. All classes are normally distributed with the identity matrix as the covariance matrix (lucky us). The four class means are at $(-5, 0, 0)$, $(5, 0, 0)$, $(0, 0, 3)$, and $(0, 0, -3)$. The classes have equal priors. We are going to study dimensionality reductions for this classification problem.

(a) Assume we have an infinite amount of training data; so we basically know the class distributions perfectly. What will be the first principal component? Briefly explain your answer. You can use explicit calculations in this, but there is no need for this.

(b) Now consider the Fisher mapping and let us reduce the dimensionality to 1 (one) to start with. In which direction of the following three will the optimal 1D subspace be: $(1,0,0)$, $(0,1,0)$, or $(0,0,1)$? Explain your answer. Again: explicit calculations are not needed, but if you prefer to, you can of course provide them.

(c) We are still in the infinite training set size setting. Explain that if we reduce the dimensionality to 2 (two), that PCA and Fisher mapping both provide the same subspace.

(d) We now rescale the first feature by dividing all its values by 4. Does the 2D PCA subspace change? Does the 2D Fisher subspace change? Explain if, why, and how they change.

(e) Assume now we are back in the situation of question c. but we are now in the more realistic setting in which we carry out PCA and Fisher on the basis of only a handful of samples. E.g. three or four per class. We again are going to reduce to 2 dimensions. Which method will turn out to work best, given that we will use the 2D representation for classification?

6.5 Feature Selection

Exercise 6.14 We are dealing with a 20-dimensional classification problem with three classes and 10 samples per class. We would like to reduce the dimensionality of this initial 20-dimensional space to only 5. As feature subset evaluation criterion you have taken the classification performance of your favorite classifier.

(a) How many criterion evaluations are necessary to come to a choice of 5 features when you rely on sequential feature forward selection?

(b) How many criterion evaluations are necessary when you use sequential feature backward selection instead?

(c) How many criterion evaluations does one have to make when one would search for the optimal subset of 5 features? Optimality is of course measured in terms of the evaluation criterion chosen.

(d) Which one of the three feature subsets obtained above would you prefer to use for your actual classification problem? Explain your answer.

OPTIONAL

Exercise 6.15 Say we want to solve a high-dimensional classification problem (e.g. 1000 features or more maybe) with relatively few training objects (say, 100) by selecting a few features (say, about 30) from the large, original collection.

(a) Pick three feature selection strategies (e.g. feature forward selection etc.). Write these down and provide a brief description of how they work for every one of them.

(b) From the three selection strategies you picked, give the best performing and the worst performing strategies (w.r.t. the expected true error rate). Explain your answer properly and provide convincing reasons for your choices.

(c) Which two selection strategies would you pick if the speed of the selection process matters the most? Which one will be the fastest and which one the slowest?

Exercise 6.16 We consider feature selection based on the Bayes error: the absolutely minimal error that can be achieved on the classification problem at hand.

(a) Construct a 3-dimensional 2-class classification problem for which feature forward selection (using the Bayes error as criterion) will outperform individual feature selection if we want to have a subspace of dimension 2. Make clear that the subspaces that will be obtained are indeed different and that the better subspace is the one that feature forward selection obtains. Also be clear in the description of the distributions that you choose for the two classes (drawings could be fine, but may be hard to correctly interpret).

(b) Show that, when using the Bayes error as the selection criterion, individual feature selection can never outperform feature forward selection when reducing a 3-dimensional problem to 2 dimensions.

(c) Construct a 2-class classification problem (that should at least be 4D according to b!) for which individual feature forward (using again the Bayes error as criterion) will outperform feature forward selection if we want a subspace of dimension 2. Again, make clear that the subspaces that will be obtained are indeed different and that the better subspace is the one that feature forward selection obtains. Be clear in the description of the distributions that you choose for the two classes.

Exercise 6.17 You are dealing with a 10-dimensional classification problem with two classes and 100 samples per class. You decide to study the nearest mean classifier (NMC) and the Fisher classifier (fisherc) in order to solve this classification problem. You also decide to perform a feature selection to 3 (three) features before you train your final classifier. As feature subset evaluation criterion you take the classification performance the respective classifier has on the training set.

(a) How many criterion evaluations are necessary to come to your choice of 3 features (out of the 10) when you use sequential feature forward selection?

(b) How many criterion evaluations are necessary to come to your choice of 3 features (out of the 10) when you use sequential feature backward selection?

(c) How many criterion evaluations do you need to make sure you find the overall optimal combination of 3 features?

(d) How many criterion evaluations do you need to make sure you find the overall optimal combination of features?

(e) Say it is allowed for features to be chosen more than once. How many criterion evaluations do you need to make sure you find the overall optimal combination of 3 features (so they don't have to be unique) for the Fisher classifier? And how many for the NMC? [Yes, this is a tricky one.]

END OPTIONAL

Exercise 6.18 Given a data set with 100 features. Say you want to find the 5 best features. How many feature sets do you need to check for this in case we have to rely on an exhaustive search?

6.6 Selection Experiments

Exercise 6.19 Check the help of the standard search strategies for features selection `featself`, `featselb`, and `featseli`.

(a) Create 100 samples from a 10D "difficult" distribution using `a = gendatd(100,10)`. Apply a few of these feature selection strategies together with the `'maha-s'` criterion (what does `'maha-s'` do?).

(b) Which features are the best two and do the algorithms pick them out? And if not, do they agree among each other on the optimal choice?

Exercise 6.20 Load the diabetes data set using `a = diabetes`. This data set is a real-world data set (though an old one). The aim is, or rather, was to predict the presence of diabetes mellitus in people based on the eight measurements made. Some of the features included are diastolic blood pressure, triceps skin fold thickness, and age.

(a) Split up the data set in 50% for training and the rest for testing. Based on the training set, what is the optimal 5-dimensional subspace for the fisher classifier? (Hint: you can loop you way out of this but it might be more handy to use some feature from `nchoosek`; read the help carefully.)

(b) What is the optimal 5-dimensional space for a 1NN classifier?

(c) What is the error attained on the test set for these classifiers with their respective optimal feature spaces.

(d) Check some search strategies and selection criteria (`help feateval`) and see whether they are able to pick out the same 5-dimensional subspace. If not, do at least the error rates on the test set come close in case one uses the same classifier?

(e) Do another 50-50 split of the original data set. When redoing some of the feature selections, do you find back the same feature sets?

OPTIONAL

Exercise 6.21 This code you get for free:

(a) Run it a couple of times and have a look at the output, both in the figure and in the command window, and try to understand it. Also have a look at the code of course. E.g. what does `randperm` do in the code and what is `rp` used for? What does `featseli` do? And `'eucl-m'`?

(b) Can you find a selection strategy, which optimal performance is, by and large, better than the best performing one in (a)? Recall that there are not only so-called filter approaches.

6.7 Extraction and Selection

Exercise 6.22 Consider the three equally probable classes X, Y, and Z in a 2-dimensional feature space. See Figure 6.7. All have a uniform and circular distributions of diameter

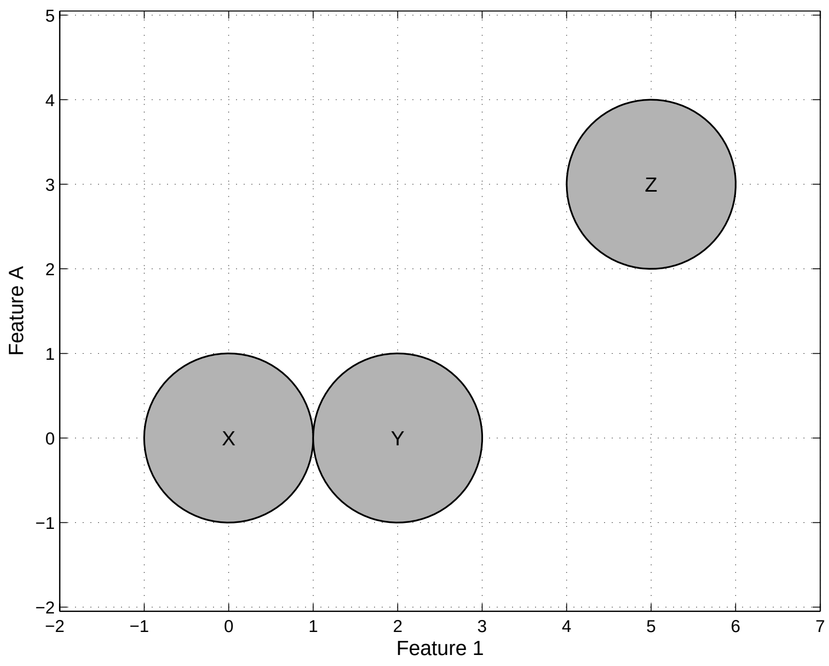


Figure 6.1: Three equally probable classes X, Y, and Z in a 2-dimensional feature space.

2 and the three class means are located in $(0,0)$, $(2,0)$, and $(5,3)$, respectively. In addition, the distributions are such that all class covariance matrices, as well as the within-class covariance matrix $\mathbf{S}_W$, are equal to the identity matrix $\left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)$.

(a) In a forward selection scheme, what feature is selected when the selection criterion used is the sum of the (squared) Mahalanobis distances[1] between the classes? (You may also use the multi-class Fisher criterion—often denoted J_{F}—as this leads to the same result.)

(b) Using the same criterion, what feature is selected when a backward approach is used?

(c) Use the given class means to determine the between-class scatter, or between-class covariance matrix, and check that both $(3,2)$ and $(-2,3)$ are eigenvectors.

(d) Use the previous between-class matrix, together with the within-class covariance matrix, to determine the 1-dimensional optimal solution based on the Fisher criterion.

(e) Which of the three previous solutions performs worst? How much class overlap is attained when relying on the feature forward selection procedure?

(f) Change one or more of the current three class means and modify the problem such that the feature extraction performs better (i.e., gives lower class overlap) than the two feature selection methods. Stated differently: provide three class means and show that the feature extraction approach above now outperforms the feature selection methods.

[1]Mahalanobis distances are the same as the Euclidean distance, but they have a correction based on some covariance. Specifically for this exercise, the Mahalanobis distance between two classes based on the two class mean m_1 and m_2 is given by $(m_1 - m_2)^T \mathbf{S}_W^{-1}(m_1 - m_2)$.

(g) Finally, assume that the class distributions are squares aligned with the feature axes instead of the circular distributions and that the three class means are still the same. In this setting, do the solutions to questions 1, 2, and 3 change? And if so, how?

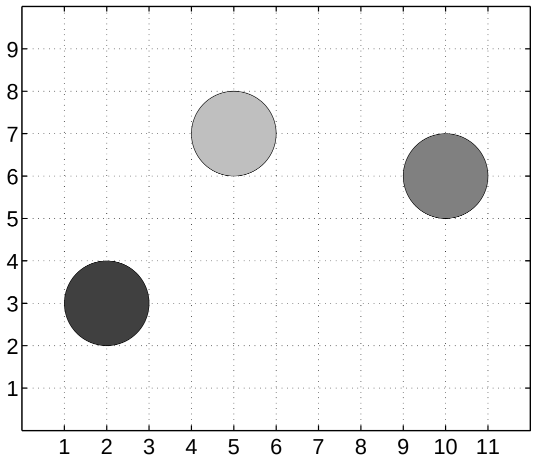


Figure 6.2: Example configuration of three classes.

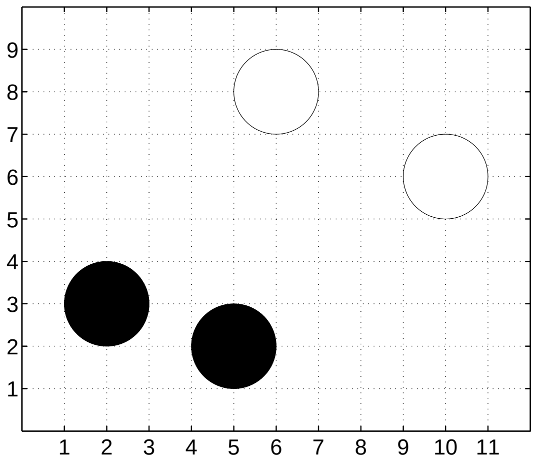


Figure 6.3: Example configuration of two classes (both of which in turn consist of two clusters).

Exercise 6.23 Let us consider a two-dimensional, three-class problem in which all classes are uniformly distributed discs with radius 1. All class means are variable but differ so that there is no class overlap in the 2D feature space. The prior probabilities of all classes are equal. Figure 6.2 above gives *an example configuration* in which the three class means are given by $(2, 3)$, $(5, 7)$, and $(10, 6)$, respectively.

(a) Let us reduce the feature dimensionality from 2 to 1 by means of the Fisher mapping (`fisherm` in terms of good ol' Matlab's PRTools). Configure the three class means such that in the 2D space there is *no overlap* but in the 1D space found by the

Fisher mapping two of the three classes *completely* overlap. You are only allowed to put the class means in the *grid points* given in the figure. Provide the three coordinates of your solution.

(b) Is it possible to give three means for which there is no overlap in 2D but for which all three classes overlap completely in 1D when using the Fisher mapping?

We now consider a similar setting as exemplified in Figure 6.2, but now we have *two* classes (one black one white) both consisting of two clusters. All four clusters are uniformly distributed discs with radius 1. (The four clusters have equal priors.) An example configuration is given in Figure 6.3.

(c) Configure the four class means such that if one would look at either of the two features, both classes would perfectly *overlap*, while in the original 2D space the two classes are perfectly non-overlapping. In other words, construct an example for which selecting a single feature would give a Bayes error of 0.5, while the Bayes error in the original space is 0. Again, you are only allowed to put the means in the *grid points* given in the figure. Provide the four coordinates of the clusters of your solution. Make sure you make clear where the black clusters go and where the white go.

(d) Would a linear feature extraction technique be able to provide a better one-dimensional subspace for the problem created in c. than feature selection is able to do? That is, can feature extraction find a 1D feature for which the two classes do not fully overlap?

(e) Is it possible to create a classification problem, again positioning the four clusters, such that feature selection will outperform any kind of feature extraction? Explain your answer.

Exercise 6.24 Figure 6.4 displays a three-class problem in which all classes are uniformly distributed on a 2×2 square. The three class centers are located in $(-1, 6)$, $(0, 0)$, and $(2, 0)$, respectively. The prior probabilities of all classes are equal.

(a) What is the Bayes error for this two-dimensional classification problem?

We now are going to reduce the dimensionality of this classification problem from the original two features to a single one.

(b) Using the means of the classes, what is the sum of squared Euclidean distances for this problem if one would only consider Feature A? What if one would only consider Feature B? If we would perform a feature selection based on this error, which one of the two features would we choose and why?

(c) What is the Bayes error for this problem if one would only consider Feature A? What if one would only consider Feature B? If we would perform a feature selection based on this error, which one of the two features would we choose and why?

Assume now that the sizes of the squares of all three classes increase from 2×2 to 4×4 (the class centers indeed remain the same).

(d) Similar to b., again using the means of the classes, what is the sum of squared Euclidean distances for this problem if one would only consider Feature A? What if one would only consider Feature B?

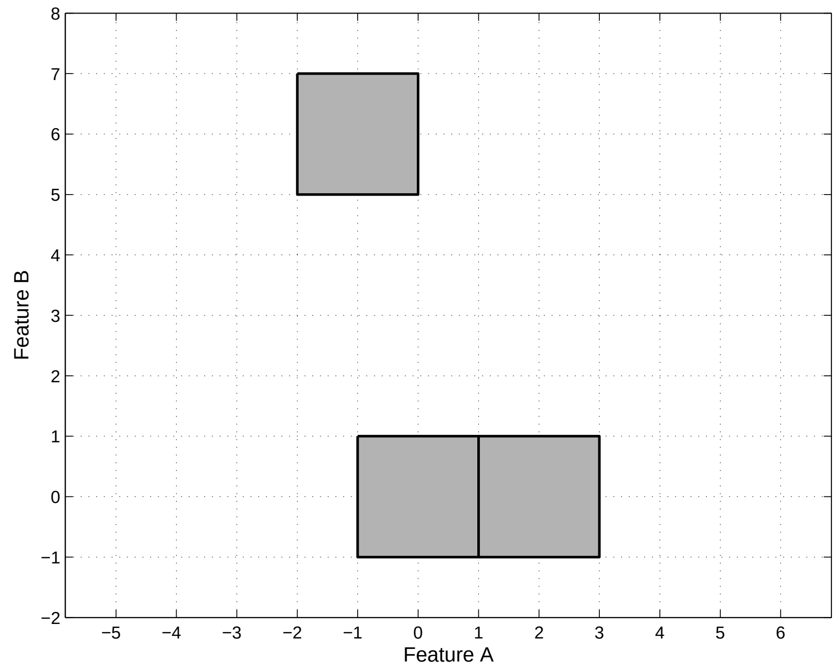


Figure 6.4:

(e) Same as in c., what is the Bayes error for this problem if one would only consider Feature A? What if one would only consider Feature B?

Instead of feature selection, we could also have resorted to feature extraction. In order to perform PCA, we need the covariance matrix of the data.

(f) Determine the between-class covariance matrix (or the between class scatter) for the three class problem we consider.

(g) Assume that the covariance matrix of the squares is given by $\left(\begin{smallmatrix} 2 & 0 \\ 0 & 2 \end{smallmatrix}\right)$. Use this in combination with your answer in f. to determine the total covariance.

END OPTIONAL

6.8 Combining

Exercise 6.25 We briefly consider the use of bagging.

(a) Compare 1NN with its bagged version on `gendats` with 10 samples in every class. For the bagged version you need something like `baggingc(trn,knnc([],1))`. Compare in terms of test error as well as visually.

(b) You may want to check if using `gendatd` instead of `gendats` improvements can be obtained.

OPTIONAL

Exercise 6.26 We are going to combine classifiers trained on different representations of hand written digits. We compare these to the individual classifiers and the classifier trained on the data set having all features.

First we construct the various data sets:

```
a1 = mfeat_kar, [b1,c1,I,J] = gendat(a1,0.3);
a2 = mfeat_zer, b2 = a2(I,:); c2 = a2(J,:);
a3 = mfeat_mor, b3 = a3(I,:); c3 = a3(J,:);
```

The `kar`, `zer`, and `mor` indicate the type of features we are considering.

(a) Train three nearest mean classifiers on the different data sets:

```
w1 = nmc(b1)*classc; w2 = nmc(b2)*classc; w3 = nmc(b3)*classc;
```

(The `classc` is a trick to make sure that classifiers output posteriors.) Estimate the three corresponding error rates on the corresponding `c`-data sets, e.g., for the first data set: compute `c1*w1*testc`.

(b) Train and test the same classifier on the data sets `b = [b1 b2 b3]` and `c = [c1 c2 c3]`, which contain all features from the three data sets above. Compare its error rate to those from (a).

Given the trained individual classifiers `w1`, `w2`, and `w3`, a fixed combiner can be constructed as follows: `v = [w1; w2; w3]*meanc`. This is the mean combiner. Other combiners that PRTools allows are `minc`, `maxc`, `prodc`, `medianc`, and `votec` of which it should be clear what kind of combining rule these implement.

(c) Test some combiners, by means of `[c1 c2 c3]*v*testc`, and compare the results to the four earlier error rates you estimated.

(d) Looking back at the exercise, do you understand why the special form of the `gendat` command, in which the `I` and `J` occur, had to be used? What would we have been doing if we would just have generated the `b` and `c` data sets by applying `gendat` to the three `a` data sets individually? Obviously, you may want to check `help gendat` for an initial clue and to see what `I` and `J` contain.

Exercise 6.27 **(a)** Generate a data set by means of `gendath` and train two differently behaving linear classifiers on it. Take, for instance, `w1` equal to `nmc` and `w2` to`ldc`. Generate posterior probabilities from these by computing `p1 = a*w1*classc` and `p2 = a*w2*classc` and combine them in one data set by `p = [p1 p2]`.

How many features does this new data set have and why is this so?

(b) Scatter plot the first and third dimension of `p`. What do these dimensions correspond to? Try to understand that the original classifiers correspond to horizontal and vertical lines at 0.5 in this same scatter plot.

(c) Straight lines not parallel to any of the two axes actually combine the posteriors from the two classifiers. Are there combiners possible that indeed seem improve upon the two individual classifiers?

Exercise 6.28 As simply as fixed combiners can be constructed, one can construct trained combiners. An example is `v = [nmc*classc ldc*classc]*fisherc`, which takes the two classifier outputs of `nmc` and `ldc` and combines them through training a `fisherc` on top of it. All is simultaneously trained by executing `w3 = a*v` on a data set `a`.

(a) Take the data set from Exercise 6.27, `gendath`, construct a training and a test set, and compare `nmc`, `ldc`, and the trained combiner.

END OPTIONAL

6.9 An AdaBoosting Exercise

Exercise 6.29 The AdaBoost combining scheme, `adaboostc` in PRTools, enables one to experiment with different AdaBoost schemes based on different base classifiers. The classical one is the so-called decision stump (`stumpc`, check the help).

(a) Generate a standard banana data set `a = gentdatb` and visualize the decision boundaries that are obtained by means of boosting when you combine 1, 10, 100, and 1000 base decision stumps, respectively. N.B. You might want to use the following form to suppress annoying messages, while `adaboostc` is training: `adaboostc(a,stumpc,number_of_base_classifiers,[],0)`. The final `0` silences the verbose.

(b) When combining 1000 base classifiers, AdaBoost seem to suffer from what is called overtraining. How can you maybe see this in the scatter plot? How could you probably see this from the error rates on an independent test set?

(c) Repeat the experiment but replace the base classifier with `nmc` and with `fisherc`. Are there noteworthy differences between the various solutions obtained? Also compare results between different base classifiers.

(d) What can we expect AdaBoost to do when we employ it in combination with, say, a 1NN or an SVM with a radial basis function? Could one expect any improvements from such scheme?