DSAIT4020
Elements of Statistical Learning

laboratory course manual

Version 0.9.1, 2024

David M.J. Tax, Jing Sun, Tom Viering

Contents

Week 1

Basics of Machine Learning

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

- should be able to experimentally determine if classifiers are sensitive to the rescaling of one of the features in a dataset,
- have started building an ML pipeline for performing classification experiments (see Midterm project next chapter).

1.1 Prior knowledge

The next topics I assume are known, as they are taught in DSAIT4005 Machine and Deep Learning:

1. Classifiers: QDC, LDA, nearest mean, Parzen classifier, kNN classifier, Naive Bayes, logistic classifier, decision trees, random forest, support vector machines
2. Deep learning methods, CNN, RNN, self-attention,
3. overfitting,
4. curse of dimensionality, bias-variance tradeoff,
5. Cross-validation,
6. Learning curve and feature curves.

If you are unsure, please look back to the material of course DSAIT4005 Machine and Deep Learning.

In the exercises you can use `prtools` as `sklearn`. The setup of `prtools` is exactly the same as you did in the course DSAIT4005 Machine and Deep Learning. The `sklearn` toolbox is generally more used in the world, but not always the most instructive for classroom exercises (because it sometimes contains hacks).

1.2 Generative AI

The use of Generative AI/Large Language Models is not allowed, except for improving the writing quality (e.g. grammar and spelling checks) and readability of a text. If you use it for gammar or spelling checks, mention how you have used Generative AI in your report.

Generative classifiers that make use of Bayes rule, are allowed to solve classification or inference problems.

1.3 The scaling problem

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

Here 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 1.1 **(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. This can be done by:

```
newtrain = a;
newtrain[:,[1]] *= 10.
```

Train the above mentioned classifiers 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 1

Practice project: Digit Classification

You have to predict the digit based on pixel values of images. Your task is to explore and preprocess the dataset. Furthermore, you will train and finetune models for the classification task. This assignment is a preperation for your final assignment, so that you can learn how to set up a proper machine learning experiment. Grading will be done by the instructors and via peer feedback. Participation is optional but recommended, as it is a good practice for the final project which is mandatory.

What will you learn? You will learn...

- To explore the given dataset and perform necessary preprocessing.
- To train and **fairly** evaluate classification models using the accuracy.
- To optimize model performance using hyperparameter tuning via gridsearch.
- To present your findings and what you did **and why** in a brief report.

Overview

You will train 4 classification models: KNN (K-Nearest Neighbour) classifier, logistic regression trained with SGD, the SVM (Support Vector Machine) and the Decision Tree. Performance is measured in terms of accuracy. When we mention 'performance' it will always refer to the accuracy on unseen data (generalization performance). A fair estimate of the performance should thus reflect the performance on new and unseen data.

The data consists of images with a resolution of 28 times 28 pixels. The classification target is the digit. We provide a template Jupyter Notebook to load the data with Python.

You will submit your machine learning models' predictions to the autograder. You also need to provide an estimate of the accuracy you think your model has on new data. Both the predictions and your predicted accuracy will be automatically graded. You are encouraged to submit early and as often as possible, this can help you find mistakes. Good luck, and please feel free to ask if you have any questions!

Important Notes

- **Submission Deadline:** Friday 29th of November at 17:00 (end of week 2.3).
- **Collaboration:** This is an assignment that can be done in groups. You may discuss ideas with your peers, but your code and report must be original and made by your team. All team members should be able to explain the code and the choices made.
- **Academic Integrity:** We perform automated fraud checks. Copying and pasting pieces of code from other teams is explicitly forbidden. Fraud will have serious consequences.

Deliverable 1: Report

Submit a PDF report (2-3 pages) that includes the answers to the questions. Structure your report according to part A, B, C, D (see next page) and clearly indicate the title of each part. Do not repeat the questions in the report. Use running text and use figures and or tables to present your findings, text (also in figures) should be readable when printed to A4.

Assume the audience is another student in the class. You do not have to explain in detail what the dataset is and what the goal of the assignment is. The report **should** contain sufficient information such that the experiments are **reproducible**. This means that another student with your report should be able to reproduce your result.

Deliverable 2: Your code

- This deliverable contains all your code. In case of multiple files please ZIP them.
- Ensure that all your code runs. Explain the required packages needed and their versions, or any other dependencies.
- Make sure the code is documented so we can find what is where via a readme.

Assessment criteria:

- **Runnable code**: The code should run without errors and should produce all results contained in the report.
- **Reproducible report**: The report has sufficient amount of detail so another student could reproduce your work.
- **Autograder**: The autograder grades the performance of your model's predictions, and it also grades the accuracy you have provided.
- **Explanations and Arguments**: You explain concisely what you did and why.
- **Technical Correctness**: You follow the standards in machine learning to ensure your to ensure you have a reliable experiment and findings.
- **Presentation of Results**: Clear and concise presentation of your results, including tables and graphs (if necessary), and a discussion of findings. Graphs have labeled axes that are readable, and tables are readable and columns and rows are labeled.

Assignments

Stuck? Maybe consult the hints on the next page.

Part A - Data Exploration and Preprocessing

1. Load and inspect the dataset: discuss the meaning of the data and make sure that they are properly loaded. For example, are all data values sensible? Please include a figure or table to illustrate that the data is properly loaded.
2. Perform the necessary steps to put the data in the right format for the machine learning algorithms. Explain the steps you take and why you take them. You are expected to use all features and images — there is no need for outlier pruning or other fancy preprocessing techniques.

Part B - Regression with Default Hyperparameters

3. What is the simplest baseline model we should aim to beat? Or in other words; if you would have to make a guess for the label without knowing anything about the image, what would you guess? What is the accuracy of such a guess?
4. Train the 4 models with default hyperparameters and fairly estimate their performance. Explain why the performance estimate is fair and how you estimated the performance.
5. Submit your your work to the autograder to check your work so far.

Part C - Tuning with GridSearch

6. For all 4 models, use gridsearch (or any other strategy) to identify the best hyperparameters. Use a systematic way to tune hyperparameters that is reproducible. Explain your choice of hyperparameter search ranges and settings. Include sufficient details in the report so that another student can reproduce your experiment.
7. Include a training curve (accuracy versus epochs) to illustrate how SGD converges.
8. Explain why the performance estimate is fair for the tuned models and how you estimated the performance.
9. Compare the performance of all models before and after hyperparameter tuning.

Part D - Conclusion

10. Reflect on the impact of hyperparameter tuning on the performance of the models. Do you have any ideas how you could further improve the performance of this system?
11. Select your best final model and use it in the autograder.

Hints:

These documentation pages of scikit may provide further hints:

- Dummy estimators
- Pre-processing continuous features
- Hyper-parameters tuning
- Cross-validation
- Pipelines (recommended to combine preprocessing and training in one go)

Frequently Occuring Problems:

- I score extremely low in the autograder. What can be wrong? Probably the preprocessing went wrong. Please check this in detail.
- The hyperparameter tuning did not improve the performance of the models? This probably means you did not perform the hyperparameter search correctly or you did not choose the ranges carefully. Ensure that the defaults are part of the range (why?).
- For some hyper-parameters we know that they are always positive numbers and they can go to fairly large ($1e16$) or small ($1e-16$) values. In such cases you should use (start with) a logarithmic scale for tuning: 0.1, 1.0, 10, 100, ...
- The training curve for SGD looks very strange or not converged? This probably means that you did not consider enough hyperparameters for tuning.
- Hyperparameter tuning takes very long. In this case, try to understand why that is the case. Perhaps you can try to select hyperparameter ranges to avoid this? Explain what you do and why to avoid overly long computation times.

Week 2

Computational Learning and AdaBoost

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 learning tasks in Machine Learning,
- be able to follow the derivation of a PAC bound,
- be able to follow the derivation of the AdaBoost classifier,
- implement the AdaBoost classifier,
- analyse the behaviour of Adaboost.

2.1 Proof

Exercise 2.1 **(a)** Prove that $e^{-x} \geq (1 - x)$.

2.2 AdaBoost

Before you answer the following questions, you need to read the paper 'A decision-theoretic generalization of on-line learning and an application to boosting' by Y. Freund and R.E. Schapire, 1995 (also available from Brightspace, under Course Documents, Reading Material, Computation Learning Theory).

Exercise 2.2 **(a)** Show the equivalence between the formulation of Adaboost as it is given in the slides of the course, and the formulation as it is given in the paper.

Exercise 2.3 **(a)** Implement a weak learner **WeakLearn**: the decision stump, that is minimising the apparent classification error.

To find the optimal feature f and threshold θ, you have to do an *exhaustive search* for a training set. Convince yourself that it indeed optimizes what it should optimize.

Extend the implementation of the weak learner such that it accepts a weight per object.

Exercise 2.4 (a) Test the implementation of the decision stump on a simple dataset: for instance, generate two classes from two Gaussian distributions, where the means are $\mu_1 = [0, 0]^T$ and $\mu_2 = [2, 0]^T$, and the covariance matrices are identity matrices. (If you are using Prtools, you can use the function `gendats`.) Do the parameter values that you obtain from the decision stump make sense?

Does the decision stump change if you rescale one of the features (for instance, when you multiply feature 2 by a factor of 10)?

Exercise 2.5 (a) Test the implementation on (an adapted version of) the Fashion NIST dataset. On Brightspace you will find a training and a test set, called `fashion57_train.txt` and `fashion57_test.txt`. They consist of the pixel values of small 28×28 images. In the training set, the first 32 rows are class 1, and the next 28 are class 2. The test set is larger: the first 195 rows contain samples from class 1, and the next 205 class 2. In Matlab you can read in the data using:

```
>> a = dlmread('fashion57_train.txt');
```

Train the decision stump on the training set and evaluate on the test set. What are the apparent error and the test error? Is AdaBoost overfitting?

Exercise 2.6 Implement the AdaBoost algorithm and implement the code to classify new and unseen data. Show the code.

Exercise 2.7 (a) Test your implementation on a simple dataset (for instance, the dataset from question (d.)). Which objects obtain a large weight? Are these indeed the more difficult objects?

Exercise 2.8 (a) Test the implementation on the Fashion dataset. What is the classification error on the test objects? Does it improve over the decision stump?

Exercise 2.9 (a) How does the error depend on the number of iterations T? At what value of T are you satisfied. At the optimal T, which training objects get a high weight?

Exercise 2.10 (a) Looking at the learning curve for `n=[2,4,6,10,15,20, ...]` training objects per class, how many training objects do you need to not overfit?

Week 3

Regression and Some Related Stuff

It should be noted that 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. Besides this, there are some exercises deemed optional, which are just, well, optional. So, judge for yourself which exercises you need to, want to, or should practice. There are probably too many to go through in the 2 hours of exercise lab that we have. On another note, at this point, the lecturer (ML) cannot claim that all exercises are thoroughly checked, though he really did his best.[1] If you think something is wrong, unclear, etc., let us (i.e., me, 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 formulate the basic least squares regression models (regularized, probabilistic, etc.) and derive its optimal estimators,
- comprehend the idea and use of polynomial, transformed, and kernelized regression,
- 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 Fisher's linear discriminant is formulated in terms of standard linear regression,
- appreciate the general formulation of a learning problem in terms of hypotheses, loss, and regularizer,
- have a basic understanding of kernel regression and how it differs from kernelized regression.

[1] Sure he did...

3.1 Linear Least Squares without and with Intercept

Exercise 3.1 Consider standard linear regression with the squared loss as the performance measure:

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

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 3.2 **(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 3.3 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 $X = \begin{pmatrix} 1 & 1 \\ -2 & 1 \end{pmatrix}$ and $Y = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$

(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?

Exercise 3.4 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 3.6), 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?

3.2 Regularization

Exercise 3.5 Consider standard linear regression with the squared Euclidean norm as the so-called regularizer (also referred to as L_2 regularization):

$$\sum_{i=1}^{N}(x_i^T w - y_i)^2 + \lambda\|w\|^2 = \|Xw - Y\|^2 + \lambda\|w\|^2. \tag{3.2}$$

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`).

In the next exercise, we investigate the effect of this regularization term a bit. First, however, some more math.

(a) 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 3.6 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.

Exercise 3.7 Set up an experiment demonstrating that `ridger` does, in fact, not regularize the intercept.

Exercise 3.8 Given a regression data set of fixed size. How would you tune the regularization parameter?

3.3 Bias and Variance in Classification

Exercise 3.9 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?

3.4 A Pseudoinverse

Exercise 3.10 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.

3.5 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 3.11 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 form 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 3.11(a), take the same number of samples form 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?

3.6 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 3.12 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 3.13 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?

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.

Exercise 3.14 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.)

3.7 Let's Kernelize

Exercise 3.15 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 $y = x_1 x_2$. 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.

Exercise 3.16 **(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?

Exercise 3.17 **(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 3.17(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 3.15(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)?

3.8 Nadaraya-Watson Kernel Regression

The function `kernelr` implements Nadaraya-Watson kernel regression with a Gaussian kernel. Let us compare `kernelr` and the earlier constructed kernelized regressor (using the same kernel) a bit.

Exercise 3.18 **(a)** What is the limiting behavior of both regressors when the kernel width goes to infinity. You can have a look at what happens experimentally, but derive your end result formally (i.e., by means of math). To which solution does the kernel regressor converge? Does kernelized regression take on the same values? What happens for different values of λ in the regularization?

(b) Using `getdatr`, generate a small data set, say, with 5 random 1D inputs and 5 corresponding random 1D outputs. Plot Nadaraya-Watson kernel regression and your kernelized regressor using medium to small kernel widths and very small λ. Why does the kernelized regression line have a (maybe weird) bias towards zero? Why does Nadaraya-Watson not have that? What seems to be the limiting solution of Nadaraya-Watson when the width becomes smaller and smaller?

(c) Does kernel regression typically go through all training points? How about kernelized regression? Can one enforce such behavior?

3.9 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 3.19 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.

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.

Week 4

Feature Reduction

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, PCA, ICA, LLE, etc.,
- know how to kernelize PCA.

4.1 Some Combinatorics

Exercise 4.1 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?

4.2 Stuff on Scatter Matrices

Exercise 4.2 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.

Exercise 4.3 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 4.4 **(a)** Variation to Exercise 5.18 from the book (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 4.5 Make Exercise 5.20 from the book. (Note that S_{xw} actually equals S_w?)

Exercise 4.6 **(a)** (Another variation to Exercise 5.18 from the book.) 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.

4.3 Supervised Feature Extraction: the Fisher Mapping

Exercise 4.7 **(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?

4.4 PCA and Fisher

Exercise 4.8 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 $v^T Cv$

Exercise 4.9 Assume that the overall mean of a data set is zero and the matrix can be written as

$$X = \begin{bmatrix} x_{11} & x_{21} & \dots & x_{n1} \\ x_{12} & x_{22} & \dots & x_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1p} & x_{2p} & \dots & x_{np} \end{bmatrix}$$

where p represents the number of features, n represents the number of objects.

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

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

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

(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 $X^T X$ and $x^T X$ contain.)

Exercise 4.10 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 4.11 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 4.12 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 $\left(\begin{smallmatrix}1&0\\0&2\end{smallmatrix}\right)$?

Exercise 4.13 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?

4.5 Laplacian Eigenmap, ISOMAP, etc.

Exercise 4.14 **(a)** What mapping does the Laplacian eigenmap become when we the underlying graph is fully connected and all weights equal 1? (I didn't derive the answer yet, but it seems like an interesting thing to wonder. . .)

(b) In a similar fit of curiosity, one can wonder whether one can pick a graph and a similarity measure such that the Laplacian eigenmap reduces to PCA. So, is that possible?

(c) Under what choice of graph and distance measure does ISOMAP become equivalent to PCA?

4.6 Feature Selection

Exercise 4.15 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.

Exercise 4.16 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 4.17 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 4.18 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.]

4.7 Extraction and Selection

Exercise 4.19 Consider the three equally probable classes X, Y, and Z in a 2-dimensional feature space. See Figure 4.7. All have a uniform and circular distributions of diameter 2 and the three class means are located in $(0, 0)$, $(2, 0)$, and $(5, 3)$, respectively. In

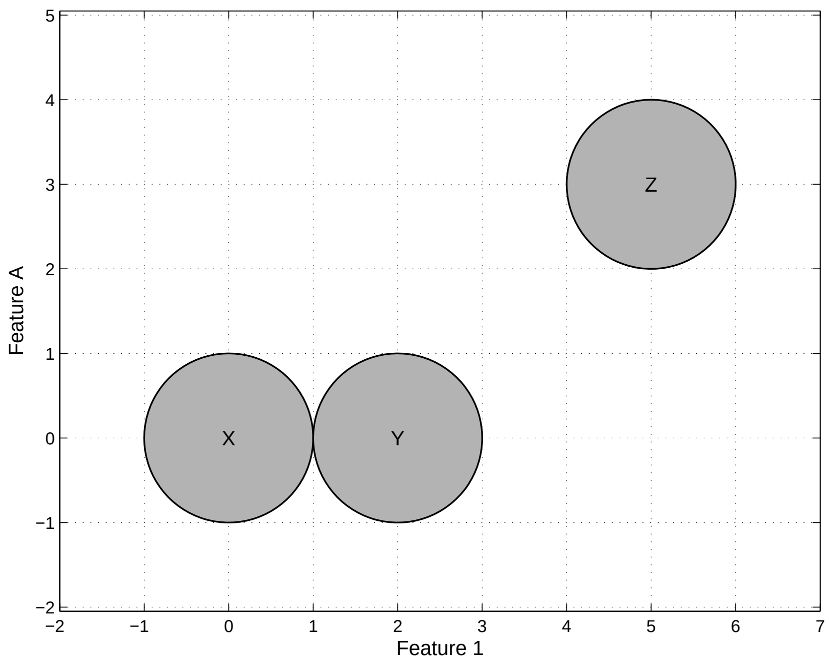

Figure 4.1: Three equally probable classes X, Y, and Z in a 2-dimensional feature space.

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?

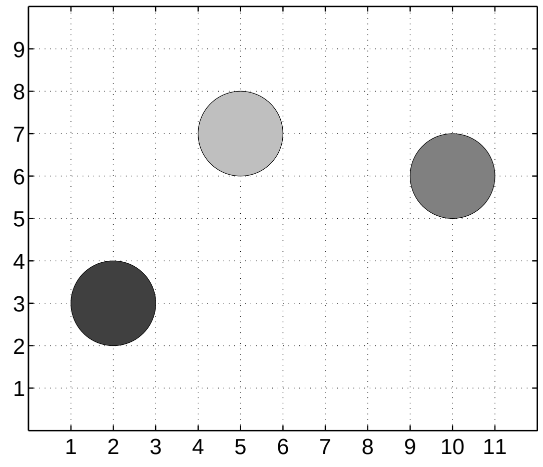

Figure 4.2: Example configuration of three classes.

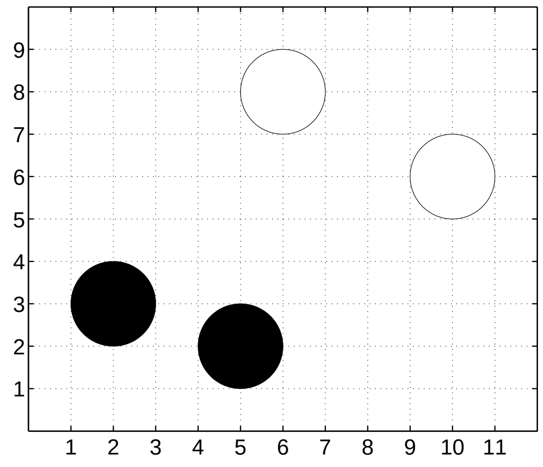

Figure 4.3: Example configuration of two classes (both of which in turn consist of two clusters).

Exercise 4.20 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 4.2 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 4.2, 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 4.3.

(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 4.21 Figure 4.4 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?

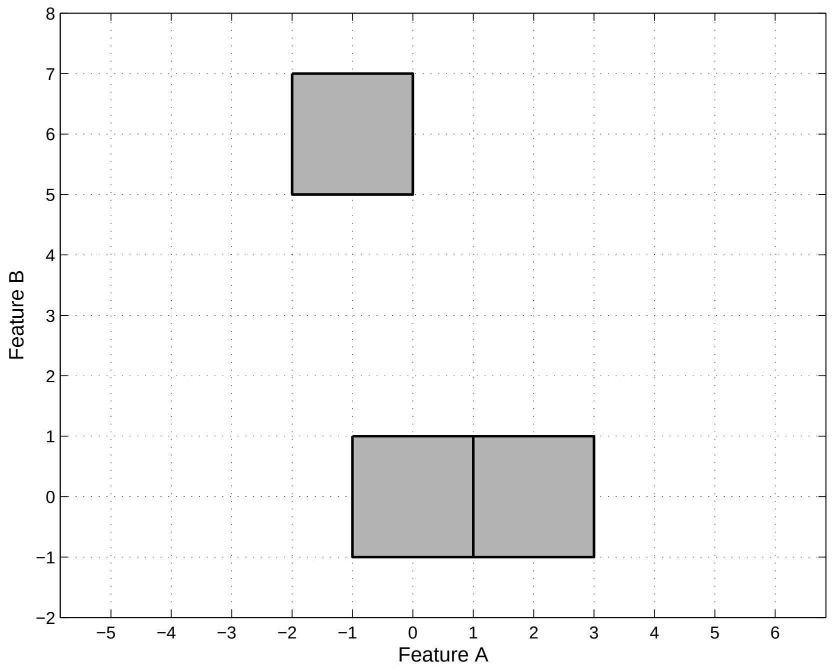

Figure 4.4:

(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.

Week 5

Clustering

In this chapter, 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.

Objectives for this week:

- 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.
- 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.

In this week, we will explore ways to guide our choice when we judge clustering results and determine the number of clusters.

5.1 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.

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 5.1 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 5.2 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 a distance matrix `D` using complete linkage, to get 3 clusters, by doing:

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

This distance matrix can be obtained from the original dataset `a` by computing some dissimilarity between the objects:

```
D = a*proxm(a,('eucl'))  # Euclidean distance
```

Of course, you're free to choose other dissimilarity definitions as well:

```
D = a*proxm(a,('city'))  # or, city-block distance
```

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 5.3 **(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?

5.2 K-means clustering

Exercise 5.4 The operation of this algorithm was explained in the lecture. In this practical session, we will familiarise ourselves with the K-means algorithm by applying it in various settings. The function `kmeans` is provided to perform K-means clustering. Take a moment to familiarise yourself with the input, output and operation of this function. For reasons of simplicity we recommend initialising the clustering with the `random` option. Apart from the aforementioned, the datasets `triclust`, `messy` and `cigars` are also available.

Exercise 5.5 This process will be performed in several simple steps, such that by the end of these steps you will have an algorithm that works (at least most of the time).

STEP 1: Function interface Call your function `km`. Think about the inputs it will need as well as the outputs it will have to generate.

```
function [P,lab] = km(a,k)
% KM Performs k-means clustering
%
% INPUT:   a     - an (n x p) dataset
%          k     - number of desired clusters
%
% OUTPUT:  P     - (k x p) matrix, the prototypes
%          lab   - (n x 1) vector, the cluster labels
```

STEP 2: Choosing initial prototypes. This can be done in several ways. Two simple ways are

1. placing the prototypes uniformly distributed in the domain (approximated by the bounding box) of the data and
2. selecting a number of the samples at random as prototypes.

For reasons of simplicity we recommend the second option.

```
[n,p] = size(a);
r = randperm(n);
P = a(r(1:k),:);
```

OPTIONAL

The initialisation of the K-means algorithm (positioning of the prototypes) can be done in several ways. Two simple ways are:

1. placing the prototypes uniformly distributed in the domain (approximated by the bounding box) of the data; or
2. selecting a number of the samples at random as prototypes.

For reasons of simplicity we recommend the second option.

(a) Can you think of possible advantages and disadvantages to these two initialisation approaches?

END OPTIONAL

STEP 3: Assigning samples to prototypes. Compute the distance of each sample to all prototypes. Assign each sample to the closest prototype.

```
dprot = distmat(a,P);
[min_dprot,lab] = min(+dprot,[],2);
```

STEP 4: Compute new prototypes. Compute a new set of prototypes by averaging the samples associated with a specific prototype. Save these new prototypes under a different name as the original prototypes.

```
Pnew = [];
for i = 1:k,
  samples_in_clust = a(find(lab == i),:);
  Pnew(i,:) = mean(samples_in_clust,1);
end
```

STEP 5: Convergence check. Compare the new set of prototypes with the old set. Convergence occurs if the largest change between the previous and the new position of a prototype dips below a preset threshold.

```
Pchange = mean((P - Pnew).^2,2);
maxPchange = max(Pchange);
if (maxPchange < thresh),
 exit_flag = 1;
end
```

STEP 6: Wrap it all in a loop. Insert the code written in **STEPS 3 - 5** in a `while` loop from which the function exists when the algorithm has converged.

```
exit_flag = 0;
while (~exit_flag),
  % CODE FROM STEPS 3 - 5 HERE
  P = Pnew;
end
```

OPTIONAL

The progress of the algorithm can be followed by plotting the prototypes as a scatterplot, and then plotting the samples, colour-coded according to the prototype they are assigned to. The intermediate steps can be viewed with the aid of the `pause` function.

END OPTIONAL

Exercise 5.6 (a) Apply `kmclust` with `plotflag` and `stepflag` set on the `cigars` and `messy` datasets for different numbers of prototypes. Is K-means better suited to one of these datasets, and if so, why?

5.2.1 The local minimum problem

You might recall that the K-means algorithm is a simple iterative way of attempting to find those prototype positions that correspond to the global minimum of the total within-scatter.

However, there are two factors that could cause this minimisation procedure to get stuck without having reached the global minimum. First, K-means is a procedure that always updates the positions of the prototypes such that the within-scatter *decreases*. Secondly, the within-scatter is not a monotonically decreasing function of prototype positions. This implies that from a specific set of non-optimal prototype positions the road to the optimum (global minimum) contains an initial *increase* in within-scatter before reaching the optimum. If K-means ends up in such a position (local minimum), it gets stuck. The following exercise illustrates this.

Exercise 5.7 (a) Load dataset `triclust`. Inspect it with the aid of a scatterplot. There are clearly three equally spaced, spherical clusters.

(b) Run your K-means algorithm several times, for $g = 3$ clusters, until a solution is found where there are two prototypes in a single cluster, and the third is positioned between the remaining two clusters. Why is this a local minimum?

(c) Does hierarchical clustering also suffer from this problem?

(d) What can we do to overcome the local minimum problem?

5.3 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) 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 5.8 (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?

5.4 Cluster validation

This part of the session is intended to familiarise you with different cluster validity measures. 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.

5.4.1 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 5.9 (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 5.10 (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 5.11 (a) Load the `hall` dataset. Perform single linkage hierarchical clustering and display the fusion graph. What do you observe in the fusion graph?

Exercise 5.12 (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?

5.4.2 The Davies-Bouldin index

D.L. Davies and D.W. Bouldin[1] 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:

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

Exercise 5.13 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?

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

(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

Final Project

6.1 Case

In the final project you are asked to solve a classification, regression or a clustering problem. What exactly, is up to you. You can decide what problem you want to tackle.

But be a bit creative. Taking a problem from Kaggle and apply another method to it, is not very creative. This is what you basically did in your practice project. In this project you are challanged to come up with a more serious research question.

What problems can you come up with. For instance:

1. Define a new classification/regression problem, collect data for that, extract useful features and train and evaluate a bunch of standard classifiers. Very recommended, although collecting data can be time consuming. An example proposed this year was to predict the train delays at Station Delft.

2. Complicate a standard/Kaggle dataset by introducing issues that can occur in real life. For instance, make the classes in a classification problem very unbalanced or introduce missing values in the features. You can also look for datasets/problems that contain sensitive attributes (age, sex, ethnic background) and see how unfairly biased classifiers become. Or what happens if you allow yourself a reject option?

3. Rephrase a standard/Kaggle dataset into another ML problem. Sometimes a regression problem can be solved easier as a classification problem, and sometimes the classes in a classification problem are ordinal (and can better be solved as a regression problem). Can some problems even be solved by unsupervised clustering?

4. Analyse the resulting classifiers more thoroughly: what features do the classifiers find most informative? Which objects are consistently misclassified? Which classes are most confused? If you look at a learning curve, can you give a hint on how many additional training data may help your method? Can you explain what a classifier is doing, as a feedback to the user (Explainable AI)?

5. ...

You are asked to be creative! But check with the teachers if your problem is interesting/challenging enough, and feasible in the given amount of time.

Note that we are typically *not* interested in superior performance of your method over other methods. We want to see that you analysed the results you have, and did an honest evaluation between different approaches. Do honest crossvalidation, report the stability of the results, do a test to check if performances between models are significant. Try to find out if there are bottlenecks in your system and where you could improve. Can you get some insight into your problem?

6.2 The deliverables

You are asked to write a report of maximally 8 pages. In this report you:

1. Explain the problem you are tackling,
2. The research question(s) attached to this problem,
3. Explain and justify the methods you use for preprocessing, predicting and evaluation,
4. Compare models using crossvalidation, perform tests to see if some methods significantly outperform other methods,
5. Clearly present your results, conclusions and discussion points, supported by relevant tables and figures.
6. Finally, indicate which group member contributed to which part of the project.

The report is graded on the complexity/interestingness of the problem, the choices of the applied methods and their motivation, the presentation of the results, and the conclusions and discussion.

Please submit both your report and your code. The code is only for reference, and the code quality does not affect your grade. The report should be complete, and understandable without the code.

We expect you to at least satisfy the requirements of the practice project, but in order to get a passing grade, you should add some interesting/non-trivial research questions and some solid evaluation of the resulting models.

6.3 Backup project: wage prediction

In case you have no idea/no inspiration to find your own dataset we provide a backup dataset. (Of course, bringing your own dataset is preferred by us) On Brightspace a dataset with data from 19,178 football players is provided. Each player is characterized by 28 features, given in the table below:

age	height cm	weight kg
nationality name	overall	potential
attacking crossing	attacking finishing	attacking heading accuracy
attacking short passing	attacking volleys	skill dribbling
skill curve	skill fk accuracy	skill long passing
skill ball control	movement acceleration	movement sprint speed
movement agility	movement reactions	movement balance
defending standing tackle	defending sliding tackle	goalkeeping diving
goalkeeping handling	goalkeeping kicking	goalkeeping positioning
goalkeepingreflexes		

The original goal is to predict the salary of the player, but next to that numerous other questions can be asked:

1. What features are most important to predict the wage (as a regression problem)?
2. What features are important to get to the top 10% wage? Or bottom 10%?
3. Are there unfair biases in terms of age, nationality, height of the person...?
4. How redundant is the dataset; or in other words, with how many players in your training set can you get a good performance? (Hint; make a learning curve)
5. Are there surprising players, that earn surprisingly little, or much?
6. ...

So, if you don't have an alternative, analyse this dataset, and feel free to come up with more interesting questions as well!