

Lecture Notes on Machine Learning

Ulrike von Luxburg

Summer 2013

Department of Computer Science, University of Hamburg

(Version as of April 23, 2013)

Contents will be updated during the lectures. Up-to-date material
to be found on the lecture's webpage.

Table of contents

Introduction to Machine Learning

What is machine learning?	5
Motivating examples and applications	6
Machine learning as inductive inference	39
Different learning scenarios	63

Warmup: k -nearest neighbor classification	74
The kNN algorithm for classification	75

Recap: Maths basics

Recap: Probability theory	101
Discrete Probability Theory	102
Continuous probability theory	125

Table of contents (2)

Recap: Linear algebra	133
Supervised learning	
Formal setup	157
Statistical and Bayesian Decision theory	158
Bayesian decision theory, intuitively	161
Formal setup	172
Particular setup for classification	188
Particular setup for regression	204
Linear Methods for regression	213
Linear least squares regression	214
Least squares with linear combination of basis functions	235
Ridge regression: least squares with L_2 -regularization	240
Lasso: least squares with L_1 -regularization	260

Table of contents (3)

Linear Methods for classification	274
Intuition and feature representation	275
Linear discriminant analysis	290
Logistic regression	307

Literature and abbreviations

Here are some books I used to prepare the lectures:

- ▶ Hastie, Tibshirani, Friedman: The elements of statistical learning.
- ▶ Devroye, Györfi, Lugosi: Probabilistic Theory of Pattern Recognition
- ▶ Györfi, Kohler, Krzyzak, Walk: A distribution-free theory of nonparametric regression.
- ▶ Bishop: ...

Introduction to Machine Learning

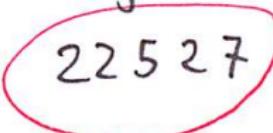
What is machine learning?

Motivating examples and applications

Hand-written digit recognition

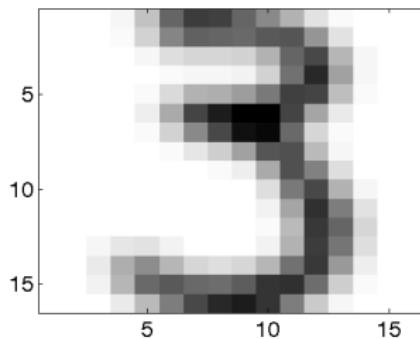
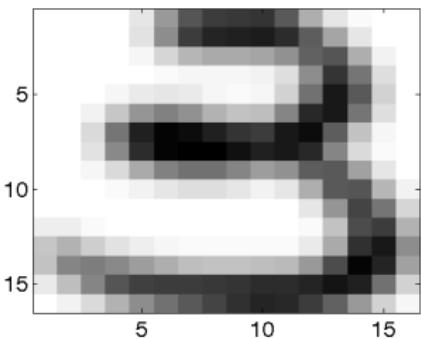
Want to automatically recognise the postal code in the address field of a letter:

Fachbereich Informatik
Vogt-Kölln-Str. 30
22527 Hamburg



Hand-written digit recognition (2)

- ▶ Take a camera picture of the address
- ▶ Segment the image into individual letters and digits
- ▶ Image of a digit: 16×16 greyscale image, corresponds to a vector with 256 entries, each entry between 0 and 1 ($0 =$ white, $1 =$ black)



- ▶ Goal: want to know which digits they are, that is we **want to find a “correct” mapping $f : [0, 1]^{256} \rightarrow \{0, 1, 2, \dots, 9\}$.**

Hand-written digit recognition (3)

- ▶ Problem: it is impossible to hand-design such a rule!

5 5 5 5 5 5 5

9 9 9 9 9 9 9

7 7 7 7 7 7 7

1 1 1 1 1 1 1

Hand-written digit recognition (4)

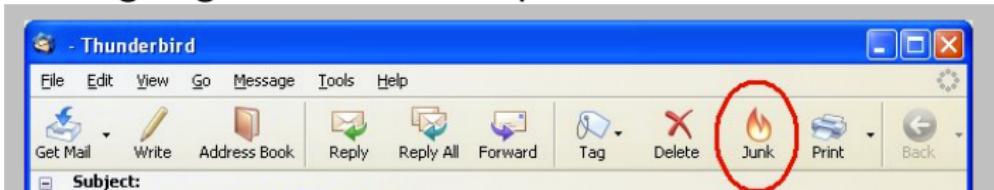
The machine learning approach:

- ▶ Present many “training examples” to the computer:
 $(X_i, Y_i)_{i=1,\dots,n}$ such that X_i = greyscale image,
 $Y_i \in \{0, 1, 2, \dots, 9\}$ the true class label
- ▶ The computer is supposed “to learn” a function f that correctly assigns digits to greyscale images

This problem is one of the “founding problems” of pattern recognition and machine learning.

Spam filtering

- ▶ Want to classify all emails into two classes: spam or not-spam
- ▶ Similar problem as above: hand-designed rules don't work in many cases
- ▶ So we are going to "train" the spam filter:



Spam filtering (2)

- ▶ Internally, the spam filter “updates its rules” based on the training example it gets.

This is a typical “online learning problem”: training arrives in an online stream, rules have to be updated all the time

Recommender Systems

Amazon: “If you like this book, you might also like this other book.”

How such a recommender system built?

- ▶ Collect shopping data of all users
- ▶ Based on the shopping data, learn a “rule” that predicts what item a certain customer might want to buy.
- ▶ This rule is based both on the joint shopping data of all users (this is our training input we use to “build a model”), and on the past shopping behavior of the particular customer (this is input X_i)

Robotics, first scenario

The DARPA grand challenge in 2005:

- ▶ Build an autonomous car that can find its way 100 km through the desert.



- ▶ The winning team was the one of Sebastian Thrun (Stanford), one of the leading machine learning researchers.

Robotics, first scenario (2)

... VIDEOS ...

Videos in teaching/videos_for_teaching

Start of the race: GrandChallenge1.flv, start at min. 1:50

Crashes: GrandChallenge1.flv, start at min 0:37

Beerbottle pass and final: GrandChallenge2.flv, start at min 4:16

Robotics, first scenario (3)

How does such a thing “work”?

- ▶ lots of sensors, cameras, etc
- ▶ all of them generate signals
- ▶ Need to “extract information” from each sensor like
 - ▶ “here is a wall”
 - ▶ “here the slope is very steep”

Each such extraction step is done by machine learning.

- ▶ Now you get lots of second order information of all these detection systems. This information is very noisy, unreliable, might be contradicting. So you need to find a way to combine all this. This is done by machine learning as well.

All these things sound not so difficult, but the DARPA success is really the end of a very long way of trial and error.

Machine learning techniques were the crucial ingredient.

Robotics, first scenario

- ▶ Robot is supposed to do very complicated movements.
- ▶ Impossible to “hand-endcode” all the rules how to do this.

Example: playing table tennis

... VIDEO ... VIDEO ...

[teaching/videos_for_teaching/JanPeters_RobotLearning](#)

Lots of reinforcement learning is involved!

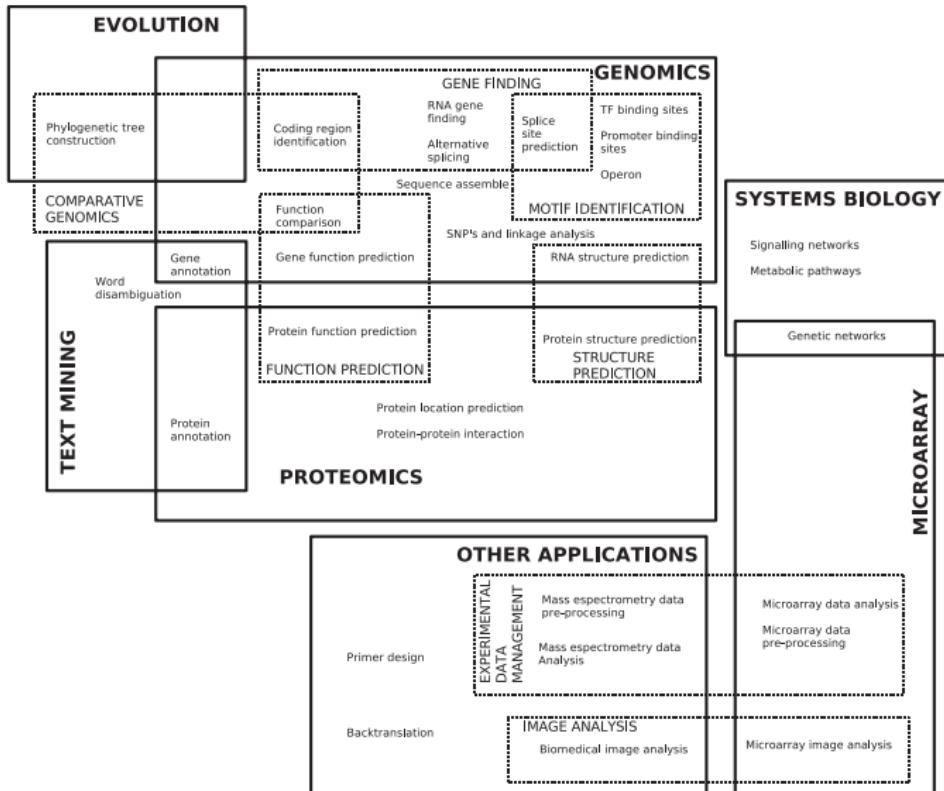
Bioinformatics

Bioinformatics is full of machine learning!

Examples:

- ▶ Learn to predict the location of genes in the genome, identify regulatory elements, non-coding RNA genes
- ▶ Predict the 3d structure of a protein
- ▶ Classify different types of diseases based on microarray data

Bioinformatics (2)



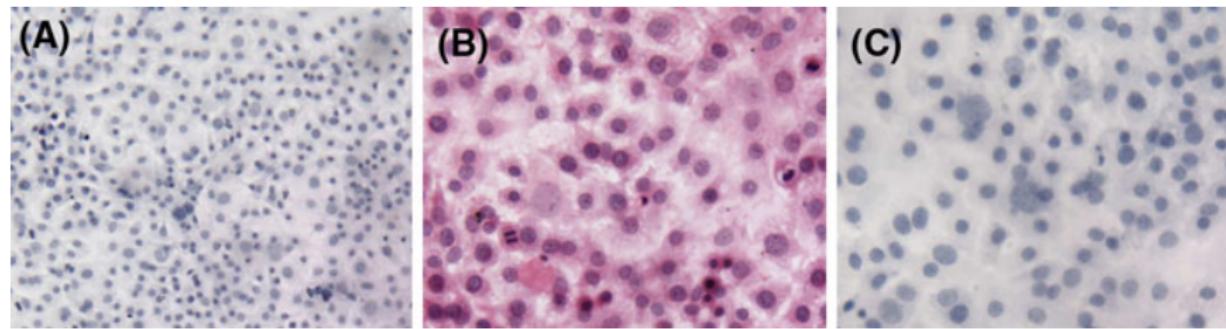
Bioinformatics (3)

Figure from Larranaga et al., 2005

Medical image analysis

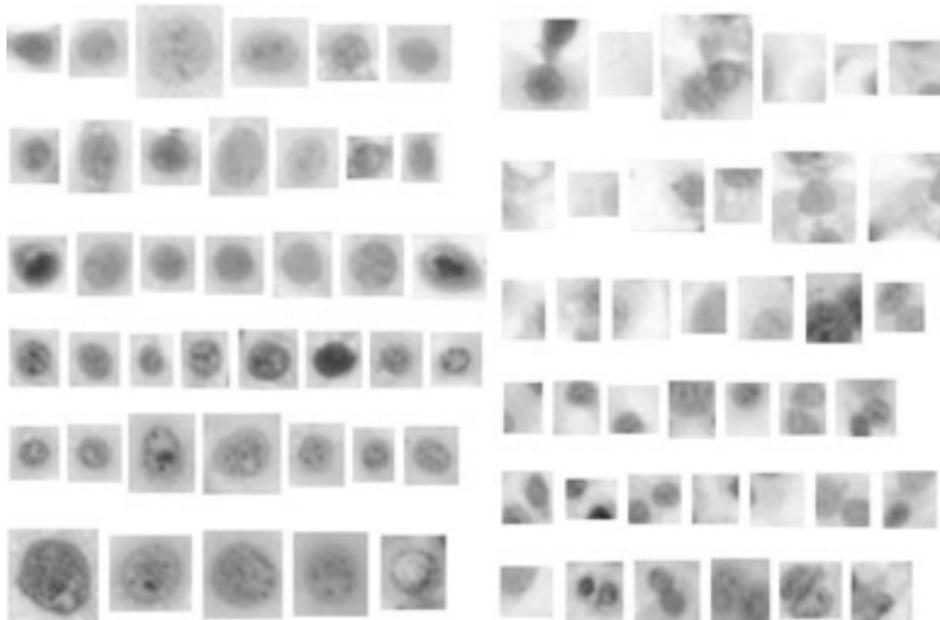
Example from cancer research: automatic detection and classification of cell nuclei in microscopy images:

Images look like this:



Medical image analysis (2)

The training data:

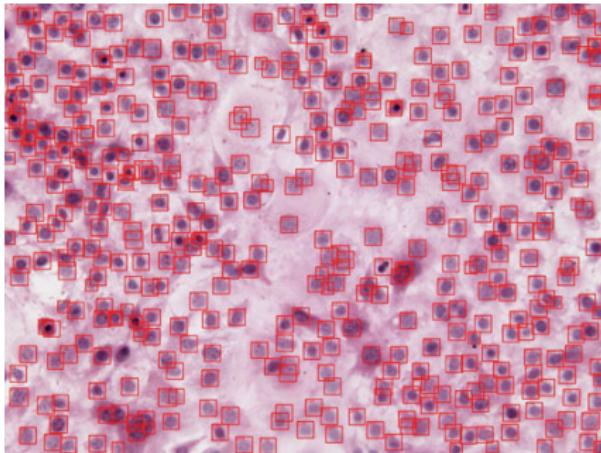


Positive training samples

Negative training samples

Medical image analysis (3)

The results:



Reference: The Application of Support Vector Machine Classification to Detect Cell Nuclei for Automated Microscopy (J.W. Han, T.P. Breckon, D.A. Randell, G. Landini), In Machine Vision and Applications, Springer, Volume 23, No. 1, pp. 15-24, 2012.

Language processing

2011: Computer “Watson” wins the american quiz show “jeopardy”. It is a bit like “Wer wird Millionär”, but not so much about facts and more about word games.



Language processing (2)

Example questions:

- ▶ One definition of this is entering a private place with the intent of listening secretly to private conversations

Language processing (2)

Example questions:

- ▶ One definition of this is entering a private place with the intent of listening secretly to private conversations
 - ~~~ eavesdropping

Language processing (2)

Example questions:

- ▶ One definition of this is entering a private place with the intent of listening secretly to private conversations
 ~ eavesdropping
- ▶ It's a poor workman who blames these

Language processing (2)

Example questions:

- ▶ One definition of this is entering a private place with the intent of listening secretly to private conversations
 ~ eavesdropping
- ▶ It's a poor workman who blames these ~ tools

Language processing (2)

Example questions:

- ▶ One definition of this is entering a private place with the intent of listening secretly to private conversations
 ~ eavesdropping
- ▶ It's a poor workman who blames these ~ tools
- ▶ The USPS cost for mailing this, a minimum of $3\frac{1}{2} \times 5$ inches, is 28 cents; wish you were here!

Language processing (2)

Example questions:

- ▶ One definition of this is entering a private place with the intent of listening secretly to private conversations
 ~> eavesdropping
- ▶ It's a poor workman who blames these ~> tools
- ▶ The USPS cost for mailing this, a minimum of $3\frac{1}{2} \times 5$ inches, is 28 cents; wish you were here! ~> postcard

Language processing (2)

Example questions:

- ▶ One definition of this is entering a private place with the intent of listening secretly to private conversations
 ~> **eavesdropping**
- ▶ It's a poor workman who blames these ~> **tools**
- ▶ The USPS cost for mailing this, a minimum of $3\frac{1}{2} \times 5$ inches, is 28 cents; wish you were here! ~> **postcard**
- ▶ There are about 50 species of the hedgehog type of this plant, so named for its spiny fruit

Language processing (2)

Example questions:

- ▶ One definition of this is entering a private place with the intent of listening secretly to private conversations
 ~> **eavesdropping**
- ▶ It's a poor workman who blames these ~> **tools**
- ▶ The USPS cost for mailing this, a minimum of $3\frac{1}{2} \times 5$ inches, is 28 cents; wish you were here! ~> **postcard**
- ▶ There are about 50 species of the hedgehog type of this plant, so named for its spiny fruit ~> **cactus**

Language processing (2)

Example questions:

- ▶ One definition of this is entering a private place with the intent of listening secretly to private conversations
 ~> **eavesdropping**
- ▶ It's a poor workman who blames these ~> **tools**
- ▶ The USPS cost for mailing this, a minimum of $3\frac{1}{2} \times 5$ inches, is 28 cents; wish you were here! ~> **postcard**
- ▶ There are about 50 species of the hedgehog type of this plant, so named for its spiny fruit ~> **cactus**
- ▶ Milorad Cavic almost upset this man's perfect 2008 Olympics, losing to him by one hundredth of a second

Language processing (2)

Example questions:

- ▶ One definition of this is entering a private place with the intent of listening secretly to private conversations
 ~> **eavesdropping**
- ▶ It's a poor workman who blames these ~> **tools**
- ▶ The USPS cost for mailing this, a minimum of $3\frac{1}{2} \times 5$ inches, is 28 cents; wish you were here! ~> **postcard**
- ▶ There are about 50 species of the hedgehog type of this plant, so named for its spiny fruit ~> **cactus**
- ▶ Milorad Cavic almost upset this man's perfect 2008 Olympics, losing to him by one hundredth of a second ~> **Michael Phelps**

Language processing (2)

Example questions:

- ▶ One definition of this is entering a private place with the intent of listening secretly to private conversations
 ~*eavesdropping*
- ▶ It's a poor workman who blames these ~*tools*
- ▶ The USPS cost for mailing this, a minimum of $3\frac{1}{2} \times 5$ inches, is 28 cents; wish you were here! ~*postcard*
- ▶ There are about 50 species of the hedgehog type of this plant, so named for its spiny fruit ~*cactus*
- ▶ Milorad Cavic almost upset this man's perfect 2008 Olympics, losing to him by one hundredth of a second ~*Michael Phelps*
- ▶ A piece of wood from a tree, or to puncture with something pointed

Language processing (2)

Example questions:

- ▶ One definition of this is entering a private place with the intent of listening secretly to private conversations
 ~*eavesdropping*
- ▶ It's a poor workman who blames these ~*tools*
- ▶ The USPS cost for mailing this, a minimum of $3\frac{1}{2} \times 5$ inches, is 28 cents; wish you were here! ~*postcard*
- ▶ There are about 50 species of the hedgehog type of this plant, so named for its spiny fruit ~*cactus*
- ▶ Milorad Cavic almost upset this man's perfect 2008 Olympics, losing to him by one hundredth of a second ~*Michael Phelps*
- ▶ A piece of wood from a tree, or to puncture with something pointed ~*stick*

Language processing (2)

Example questions:

- ▶ One definition of this is entering a private place with the intent of listening secretly to private conversations
 ~*eavesdropping*
- ▶ It's a poor workman who blames these ~*tools*
- ▶ The USPS cost for mailing this, a minimum of $3\frac{1}{2} \times 5$ inches, is 28 cents; wish you were here! ~*postcard*
- ▶ There are about 50 species of the hedgehog type of this plant, so named for its spiny fruit ~*cactus*
- ▶ Milorad Cavic almost upset this man's perfect 2008 Olympics, losing to him by one hundredth of a second ~*Michael Phelps*
- ▶ A piece of wood from a tree, or to puncture with something pointed ~*stick*

Language processing (3)

VIDEO ... VIDEO ... VIDEO

watson_jeopardy.flv, start at min 17:36

Amazing, isn't it??? How can this work???

Internally, Watson is full of machine learning!

Machine learning as inductive inference

What is machine learning?

First explanation:

- ▶ Development of algorithms which allow a computer to “learn” specific tasks from training examples.
- ▶ Learning means that the computer can not only memorize the seen examples, but can generalize to previously unseen instances
- ▶ Ideally, the computer should use the examples to extract a general “rule” how the specific task has to be performed correctly.

Deduction vs. Induction

WHO KNOWS WHAT INDUCTION AND DEDUCTION MEAN?

Deduction vs. Induction (2)

Deductive inference is the process of reasoning from one or more general statements (premises) to reach a logically certain conclusion.

Example:

- ▶ Premise 1: every person in this room is a student.
- ▶ Premise 2: every student is older than 10 years.
- ▶ Conclusions: every person in this room is older than 10 years.

If the premises are correct, then all conclusions are correct as well.

Nice in theory. For example, mathematics is based on this principle. But no natural way to deal with uncertainty regarding the premises.

Deduction vs. Induction (3)

Inductive inference: reasoning that constructs or evaluates general propositions that are derived from specific examples.

Example:

- ▶ We throw lots of things, very often.
- ▶ In all our experiments, the things fell down and not up.
- ▶ So we conclude that likely, things always fall down.

Very important: we can never be sure, our conclusion can be wrong!

Deduction vs. Induction (4)

Humans do inductive reasoning all the time: draw uncertain conclusions from our relatively limited experiences.

Example:

- ▶ You come 10 minutes late to every lecture I give.
- ▶ The first 7 times I don't complain.
- ▶ You conclude that I don't care and it won't have any consequences.
- ▶ BUT you cannot be sure ...

Machine learning as inductive inference

Here comes now our second, more abstract description of what machine learning is:

Machine learning tries to automate the process of inductive inference.

Machine learning as inductive inference (2)

In most applications of machine learning: focus is not so much on building models (“understanding the underlying process”) but more on predicting.

Example with the things that always fall down:

- ▶ We are not so much concerned *why* the stones always fall down and not up
- ▶ We just want to *predict* that they fall down

Machine learning as inductive inference (3)

Very important observation:

- ▶ Being able to predict is often easier than understanding the underlying mechanism.
- ▶ In particular, it is often not necessary to understand the underlying mechanism for being able to make good predictions!

Example:

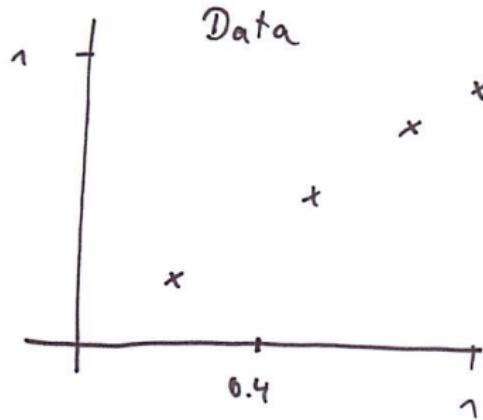
- ▶ We observe that the sun rises every day.
- ▶ So we predict that the sun is also going to rise the next day.
- ▶ To make this prediction, we don't need to understand why this is the case.

Why should machine learning work at all?

Consider the following simple classification examples:

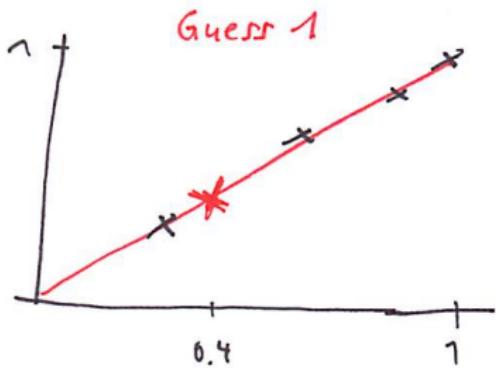
- ▶ Given: input-output pairs (X_i, Y_i) .
- ▶ Goal: learn to predict the Y -values from the X -values, that is we want to “learn” a suitable function $f : \mathcal{X} \rightarrow \mathcal{Y}$.

EXAMPLE 1: WHAT DO YOU BELIEVE IS THE VALUE $f(0.4)$?



Why should machine learning work at all? (2)

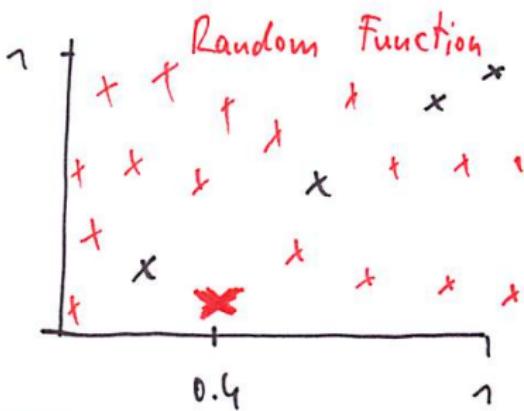
Here are two guesses:



WHICH ONE IS BETTER?

Why should machine learning work at all? (3)

Now I tell you that in fact, the function values Y_i have been generated by a uniform random number generator.



WHAT DO YOU PREDICT NOW?

Why should machine learning work at all? (4)

Consequence 1: we will only be able to learn if “there is something we can learn”.

- ▶ Output Y “has something to do” with input X
- ▶ “Similar inputs” lead to “similar outputs”
- ▶ There is a “simple relationship” or “simple rule” to generate the output for a given input
- ▶ The function f is “simple” (say, continuous and has a flat slope).

These assumptions are rarely made explicit, but something along this line has to be satisfied, otherwise ML is doomed.

Why should machine learning work at all? (5)

Consequence 2: We need to have an idea what we are looking for. This is called the “inductive bias”. Learning is impossible without such a bias.

Let's try to get some intuition for what this means.

Inductive bias: very simple example

Discrete input space $\mathcal{X} = \{0.01, 0.02, \dots, 1\}$.

Output space: $\mathcal{Y} = \{0, 1\}$

Given: training examples $(X_i, Y_i)_{i=1,\dots,n} \subset \mathcal{X} \times \mathcal{Y}$, assume there is no label noise (all training labels are correct).

Goal: Learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ based on the examples

Case 1: no inductive bias, every function $f : \mathcal{X} \rightarrow \mathcal{Y}$ can be the correct one.

Formally:

- ▶ we want to find a function out of $\mathcal{F} := \mathcal{Y}^{\mathcal{X}}$ (the space of all functions). This space contains 2^{100} functions.

Inductive bias: very simple example (2)

- ▶ Now assume we have already 5 training points and their labels.
- ▶ This means that we can rule out all functions from \mathcal{F} which do not satisfy $f(X_i) = Y_i$.
So we are left with 2^{95} possible functions.
- ▶ Now we want to predict the value at a previously unseen point X' .
- ▶ There are 2^{94} remaining functions with $f(X) = 0$ and the same number of functions with $f(X) = 1$.
And there is no way we can decide which one is going to be the best prediction.
- ▶ In fact, no matter how many data points we get, our prediction on unseen points will be as bad as random guessing.

Without any further restrictions or ingredients, the problem of machine learning would be ill posed!

Inductive bias: very simple example (3)

Case 2: model with an inductive bias.

- ▶ Assume that the true function is one out of two functions: either the constant one function **1** or the constant zero function **0**.
Our hypothesis space is $\mathcal{F} = \{\mathbf{0}, \mathbf{1}\}$.
- ▶ Then, after we observed one training example, we know exactly which function is the correct one and can predict without error.

If we have a (strong enough) inductive bias, we can predict based on few training examples.

Inductive bias: very simple example (4)

A bit too simplistic?

- ▶ Yes, the hypothesis \mathcal{F} seems too restricted to be useful for practice. The problem of selecting a good hypothesis class is called **model selection**.
- ▶ And yes, we did not take noise into account (yet).
- ▶ And yes, we did not talk about what happens if the true function is in fact not contained in \mathcal{F} after all.

In fact, the details of all this are quite tricky. **It is the big success story of machine learning theory to work out how exactly all these things come together.**

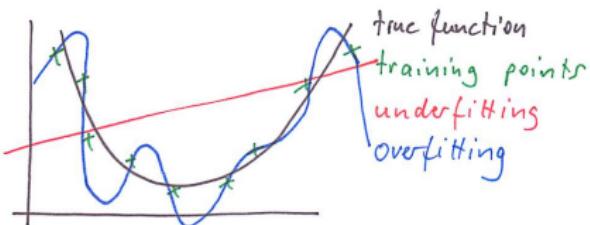
At the end of the course you will know all this, at least roughly ☺
Here is some more intuition:

Overfitting, underfitting

Choosing a “reasonable function class \mathcal{F} ” is crucial.

Consider the following example:

- ▶ True function f : quadratic function
- ▶ Training points: $Y_i = f(X_i) + \text{noise}$
- ▶ Red curve: $\mathcal{F} = \text{all linear functions}$
- ▶ Blue curve: $\mathcal{F} = \text{all polynomial functions}$



Overfitting, underfitting (2)

Overfitting:

- ▶ We can always find a function that explains all training points very well or even exactly
- ▶ But such a function tends to be very complicated and models the noise as well
- ▶ Predictions for unseen data points are poor (“large test error”)
- ▶ Low approximation error, high estimation error (\rightsquigarrow see later for definitions)

Underfitting:

- ▶ Model is too simplistic
- ▶ But estimated functions are stable with respect to noise
- ▶ Large approximation error, low estimation error (\rightsquigarrow see later for definitions)

Excursion: Inductive bias in animal learning

Any “system” that learns has an inductive bias. Consider learning in animals:

Rats get two choices of water. One choice makes them sick, the other one doesn't.

Experiment 1:

- ▶ Two types of water taste differently (neutral and sugar).
- ▶ Rats learn very fast not to drink the water that makes them sick.

Excursion: Inductive bias in animal learning (2)

Experiment 2:

- ▶ Same water, but one type of water is presented together with “audio-visual stimuli” (certain sounds and light conditions), while the other type of water is presented without these accompanying audio-visual stimuli.
- ▶ In this setting, rats did NOT learn to avoid the water that makes them sick.
- ▶ Apparently, they cannot make a connection between “sound of the food” and “sickness”.

In psychology, this effect is called the “Garcia effect” (published in a line of papers by John Garcia and coworkers in the 1960ies).

Excursion: Inductive bias in animal learning (3)

Explanation:

- ▶ From the point of view of evolution, it makes a lot of sense that the taste of food is related to whether it makes sick or not, whereas this does not seem so useful for sounds coming with food.

In our words: the rat has an inductive bias!

Reference: Garcia, John and Brett, Linda Phillips and Rusiniak, Kenneth W. Limits of Darwinian conditioning. In S.B. Klein and R.R. Mowrer, editors, Contemporary learning theories: instrumental conditioning theory and the impact of biological constraints, pages 181-204, 1989.

Inductive bias, bottom line for now

Any successful learning algorithm has an inherent inductive bias.

- ▶ we prefer to select a hypothesis from some “restricted” or “small” function space \mathcal{F} .
- ▶ Whether this function is “close to the truth” depends on whether the model class \mathcal{F} is “selected well” for the problem at hand.
- ▶ Note: for some algorithms it is obvious what the inductive bias is. For some algorithms it is hard to understand what exactly the bias is. **But if the algorithm works, there HAS TO BE a bias. This is very important to keep in mind.**

All these things will be made precise during the course of this lecture.

Different learning scenarios

Supervised, semi-supervised, unsupervised learning

Supervised learning:

- ▶ We are given input-output pairs (X_i, Y_i) as training data.
- ▶ The goal is to “learn” the function $f : X_i \mapsto Y_i$.
- ▶ “Supervised”: the teacher tells us what the true outcome should be on the given samples

The most important supervised learning tasks:

- ▶ **Classification:** $Y_i \in \{0, 1\}$, or $Y_i \in \{1, 2, \dots, K\}$.
Applications: spam classification, digit recognition, ...
- ▶ **Regression:** $Y_i \in \mathbb{R}$.
Example: predict how much a person is willing to pay for a laptop, based on his past shopping behavior

Supervised, semi-supervised, unsupervised learning (2)

Unsupervised learning:

- ▶ We are just given input values X_i , but no output values
- ▶ The learner is supposed to learn the “structure” of these inputs.

Examples:

- ▶ **Clustering:** partition the data into “meaningful groups”.
Applications:
 - ▶ Different types of cancer based on gene-expression profiles
 - ▶ Find communities in a social network
 - ▶ Build a taxonomy of a document collection, based on the topics of the documents
- ▶ **Outlier detection:** find data points that are “outliers”.
Application: intrusion detection, data preprocessing.

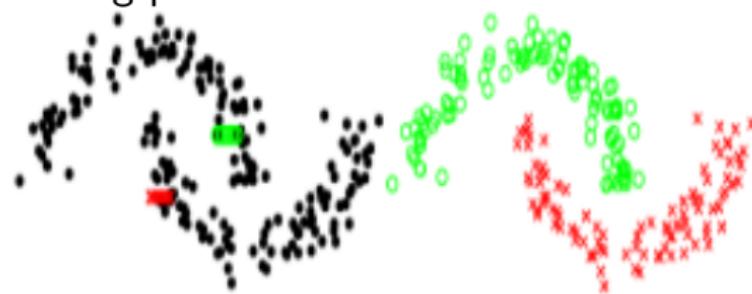
Supervised, semi-supervised, unsupervised learning (3)

Semi-supervised learning:

- ▶ We are given very many input values X_1, \dots, X_u (unlabeled points), and some input-output pairs (X_i, Y_i) ("labeled points").
- ▶ Goal is, as in supervised learning, to predict the function $f : X_i \mapsto Y_i$.

Supervised, semi-supervised, unsupervised learning (4)

- ▶ But we want to exploit the extra knowledge we gain from the unlabeled training points.



- ▶ Applications:
 - ▶ Predict the function of certain proteins: lots of proteins are known (unlabeled points), but it is very expensive to run lab experiments to find out whether they have a certain functionality or not (labeled points)

Supervised, semi-supervised, unsupervised learning (5)

- ▶ Computer tomography: lots of images of patients are available, but only few of them are labeled by a medical doctor as to contain a tumor or not.

Batch vs. Online vs. Active Learning

Batch learning:

- ▶ We get a bunch of training data before we start the learning process.
- ▶ Then we learn on this whole training set.

Online learning:

- ▶ Examples arrive online, one at a time.
- ▶ Each time a new training example arrives, we update our internal model.
- ▶ Prediction accuracy is measured in an online fashion as well (we have to predict constantly).
- ▶ Example: spam detection

Batch vs. Online vs. Active Learning (2)

Active learning:

- ▶ Instead of examples being given to us, we can actively choose which example should be the next one we want to learn (we can “actively ask” a teacher)
- ▶ Of course it is crucial to select a question that gives you as much information about the true underlying function as possible.
- ▶ Applications: domains where obtaining labels is expensive. Active learning is particularly interesting in combination with semi-supervised learning.

Discriminative vs. generative approach

We distinguish two different machine learning scenarios:

- ▶ **Discriminative approach.** We are just interested in predicting the labels Y .
- ▶ **Generative approach.** For some cases, we not only want to predict the labels, but we want to “understand” the underlying process, that is we want to model the joint distribution $P(X, Y)$, in particular the class conditional distributions $P(X|Y = 1)$ and $P(X|Y = 0)$.

In general, solving discriminative problems is easier than generative problems.

Reinforcement learning

Agent is supposed to learn how to interact and “behave optimally” in a complex environment with many possible actions and consequences. At the same time, he must get to know the environment and learn how to act in a useful way.

Examples:

- ▶ Rescue robots, say in a mining accident. Robot is supposed to find its way through the mine and accomplish certain tasks
- ▶ Control problems: We've seen the video of the robot arm that learns to throw the ball in the cup. At each point in time, the robot arm has “to decide” how to move and direct its forces in order to accomplish the task.
- ▶ Backgammon.

Reinforcement learning (2)

General model:

- ▶ Model the interaction of the agent with the environment
- ▶ At each point in time, agent is in a certain state, gets a certain amount of input (“observations”) and can perform actions that bring him to another state.
- ▶ Each action gives him a certain amount of reward.
- ▶ Goal is to learn a strategy such that he obtains as much accumulated reward as possible

Key problem: exploration — exploitation tradeoff

Reinforcement learning will be treated in the last part of the course by Dr. Norman Hendrich.

Warmup: k -nearest neighbor classification

The kNN algorithm for classification

Literature:

Hastie, Tibshirani, Friedman Section 2.3.2

Duda, Hart Section 4.5

Setting up a simple machine learning experiment

Data:

- ▶ Take a set of **training points and labels** $(X_i, Y_i)_{i=1,\dots,n}$. The machine learning algorithm has access to this training input and can use it to generate a classification rule $f : \mathcal{X} \rightarrow \{0, 1\}$.
- ▶ Take a set of **test points** $(X_j, Y_j)_{j=1,\dots,m}$. This set is independent from the training set ("previously unseen points") and will be used to evaluate the success of the training algorithm.

Setting up a simple machine learning experiment (2)

Assume our machine learning algorithm has used the training data to construct a rule f_{alg} for predicting labels.

Training error:

- ▶ Predict the labels of all training points: $\hat{Y}_i := f_{alg}(X_i)$.
- ▶ Compute the error of the classifier on this particular point:

$$err(f_{alg}, X_i, Y_i) := \begin{cases} 0 & \text{if } \hat{Y}_i = Y_i \\ 1 & \text{otherwise} \end{cases}$$

This is called the “pointwise 0-1-loss”.

Setting up a simple machine learning experiment (3)

- ▶ Define the training error of the classifier as the average error, over all training points:

$$\text{err}_{\text{train}}(f_{\text{alg}}) = \frac{1}{n} \sum_{i=1}^n \text{err}(f_{\text{alg}}, X_i, Y_i)$$

Later we will call this the “empirical risk” of the classifier (with respect to the 0-1-loss).

Setting up a simple machine learning experiment (4)

Test error:

- ▶ Predict the labels of all test points: $\hat{Y}_j := f_{alg}(X_j)$.
- ▶ Compute the error of the classifier on this particular point:

$$err(f_{alg}, X_j, Y_j) := \begin{cases} 0 & \text{if } \hat{Y}_j = Y_j \\ 1 & \text{otherwise} \end{cases}$$

- ▶ Define the test error of the classifier as the average error, over all test points:

$$err_{test}(f_{alg}) = \frac{1}{m} \sum_{j=1}^m err(f_{alg}, X_j, Y_j)$$

Setting up a simple machine learning experiment (5)

Remarks:

- ▶ Obviously, it is not so much of a challenge for an algorithm to correctly predict the training labels (after all, the algorithm gets to know these labels).
- ▶ Still, machine learning algorithms usually make training errors, that is they construct a rule f that does not perfectly fit the training data.
- ▶ **But the crucial measure of success is the performance of the classifier on an independent test set.**
- ▶ In particular, it is not the case that a low training error automatically indicates a low test error or vice versa.
- ▶ We will see in the next lecture how typically training and test errors behave.

The kNN classifier

Given: Training points $(X_i, Y_i)_{i=1,\dots,n} \subset \mathcal{X} \times \{0, 1\}$ and a distance function $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$.

Goal: Construct a classifier f that predicts the labels from the inputs.

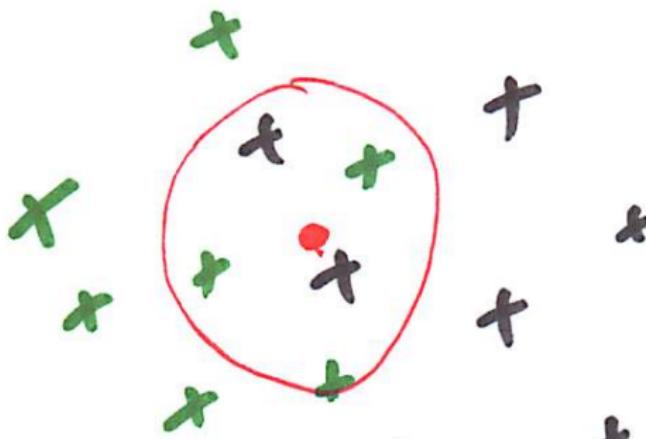
- ▶ Compute all distances $d(X', X_i)$ and sort them in ascending order.
- ▶ Let X_{i_1}, \dots, X_{i_k} be the first k points in this order (the k nearest neighbors of X'). We denote the set of these points by $kNN(X')$.
- ▶ Assign to Y' the majority label among the corresponding labels Y_{i_1}, \dots, Y_{i_k} , that is define

$$Y' = \begin{cases} 0 & \text{if } \sum_{j=1}^k Y_{i_j} \leq k/2 \\ 1 & \text{otherwise} \end{cases}$$

Example

Example (2)

Class 0 Class 1



Test point and its
5 nearest neighbors

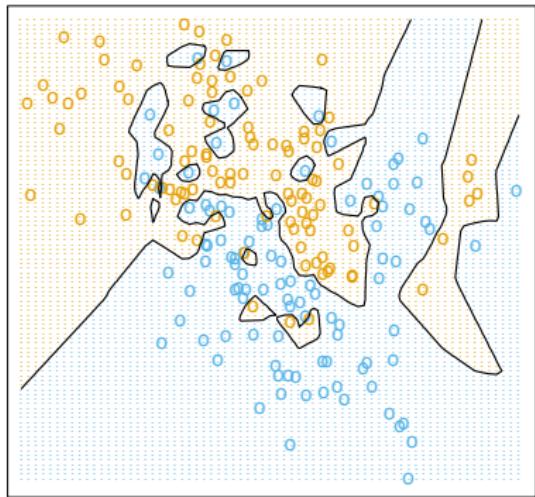
Influence of the parameter k

The classification result will depend on the parameter k .

WHAT DO YOU THINK, IS IT BETTER TO HAVE k SMALL OR LARGE?

Influence of the parameter k (2)

1-Nearest Neighbor Classifier



15-Nearest Neighbor Classifier

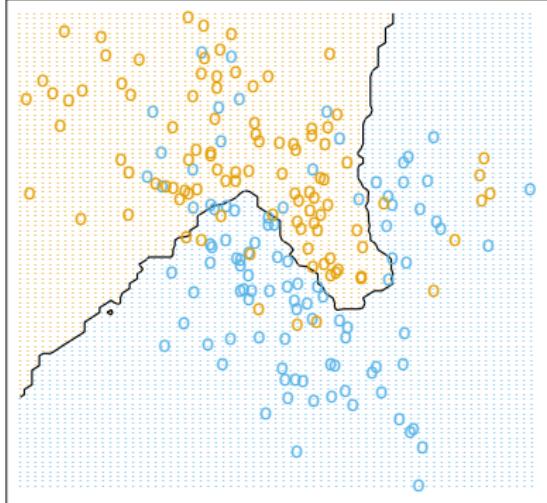


Figure from Hastie/Tibshirani/Friedman

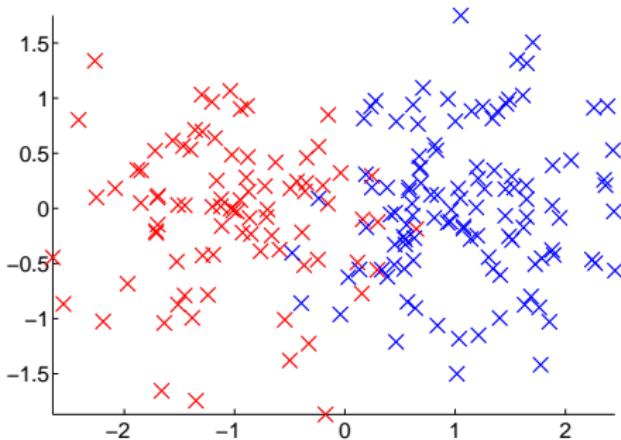
Influence of the parameter k (3)

Generally:

- ▶ k too small \rightsquigarrow overfitting
(Extreme case: $k = 1$, very wiggly and prone to noise, zero training error!)
- ▶ k too large \rightsquigarrow underfitting
(Extreme case: $k = n$, then every point gets the same label, namely the overall majority label)
- ▶ Based on theoretical considerations: k should be about order $\log n$ as $n \rightarrow \infty$ (see later in the course)

Application: simple mixture of Gaussians

We draw 100 points randomly from a mixture of two Gaussian distributions. The figure shows a typical training set:



Application: simple mixture of Gaussians (2)

Conduct the following experiment:

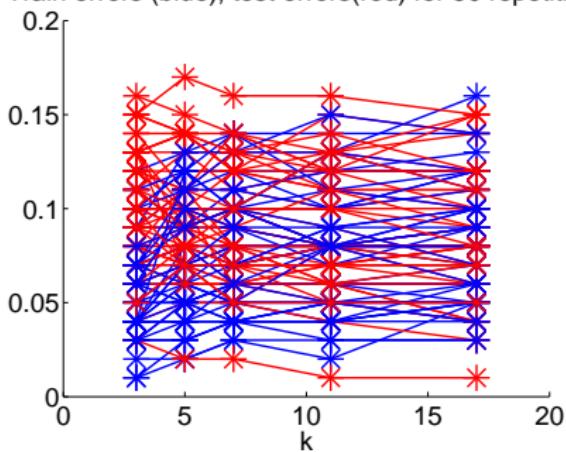
- 1 **for** $rep = 1, \dots, 10$
- 2 Draw n training points $(X_i, Y_i)_{i=1, \dots, n}$
- 3 Draw m test points $(X'_i, Y'_i)_{i=1, \dots, m}$
- 4 **for** $k = k_1, \dots, k_s$
- 5 Predict the labels of all training points, using the k nearest training points
- 6 $\text{ErrTrain}(k, rep) =$ the training error, averaged over all training points
- 7 Predict the labels of all test points, using the k nearest training (!) points
- 8 $\text{ErrTest}(k, rep) =$ the test error, averaged over all test points
- 9 **return** For each k , return the average train and test error (where the average is taken over the repetitions)

Application: simple mixture of Gaussians (3)

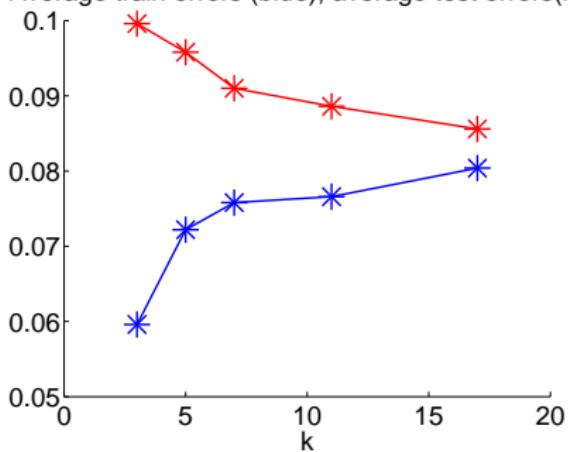
The following figure shows these train and test errors:

- ▶ Left figures: errors in each individual repetition
- ▶ Right figure: errors averaged over all repetitions

Train errors (blue), test errors(red) for 50 repetitions

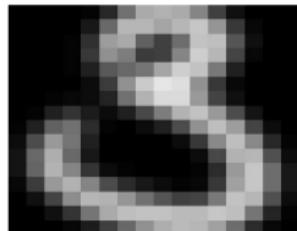
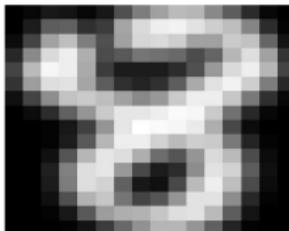
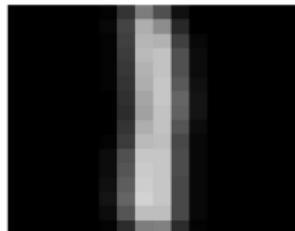


Average train errors (blue), average test errors(red)



Application: hand written digits

Data:



Represented as 16×16 greyscale image. That is, each digit corresponds to a vector of length 256 with entries in $[0, 1]$.

Task 1: Learn to distinguish between 1 and 8

Task 2: Learn to distinguish between 3 and 8

Application: hand written digits (2)

Setup:

- ▶ To apply the kNN rule we need to define a distance function between digits.

- ▶ For simplicity, we use the Euclidean distance between the vectors:

for $X = (X_1, \dots, X_{256})^t$ and $X' = (X'_1, \dots, X'_{256})^t$ we set

$$d(X, X') = \left(\sum_{s=1}^{256} (X_s - X'_s)^2 \right)^{1/2}$$

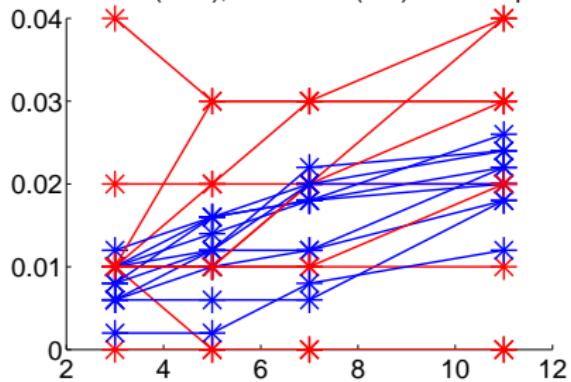
Application: hand written digits (3)

The following plots show the (average) train and test errors based on $n = 500$ training points:

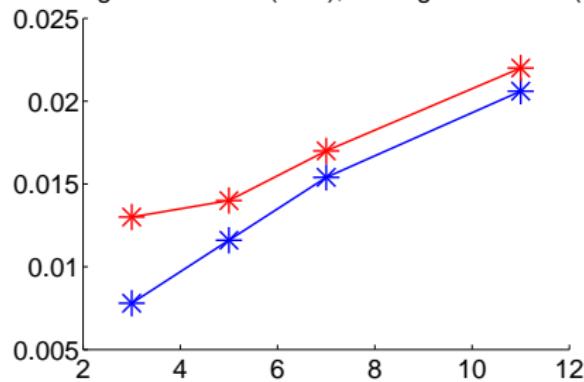
Task 1:

Digits 1 vs. 8

Train errors (blue), test errors(red) for 10 repetitions



Average train errors (blue), average test errors(red)



(x -Axis: parameter k , y -axis: error)

Application: hand written digits (4)

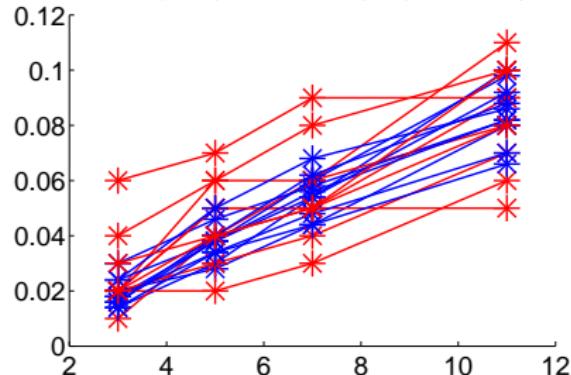
WHAT DO YOU THINK, IS THIS GOOD OR BAD?

Application: hand written digits (5)

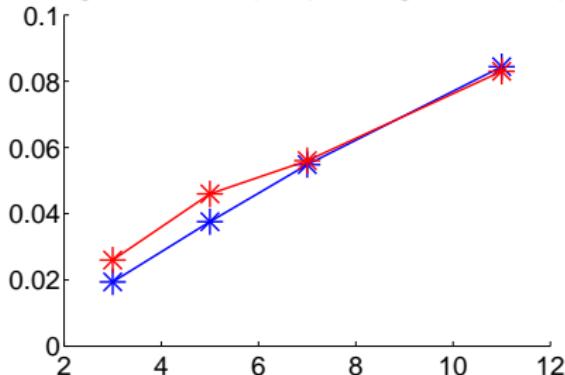
Task 2:

Digits 3 vs. 8

Train errors (blue), test errors(red) for 10 repetitions



Average train errors (blue), average test errors(red)



These results are surprisingly good!!!

Influence of the similarity function

The choice of the similarity function is crucial as well:

- ▶ The performance of kNN rules can only be good if the distance / similarity function encodes the “relevant information”.
Example: you want to classify mushrooms as “edible” or “not edible” and as distance function between mushrooms you use the difference in weight ...
- ▶ in many applications it is not so obvious how to define a good distance / similarity function
Example: you want to classify the genre of songs. How do you compute a similarity between different songs???

Inductive bias

WHAT DO YOU THINK IS THE INDUCTIVE BIAS IN THIS ALGORITHM? WHAT KIND OF FUNCTIONS ARE “PREFERRED” OR “LEARNED”?

Inductive bias

WHAT DO YOU THINK IS THE INDUCTIVE BIAS IN THIS ALGORITHM? WHAT KIND OF FUNCTIONS ARE “PREFERRED” OR “LEARNED”?

Input points that are close to each other should have the same label

Extensions

- ▶ The kNN rule can easily be used for regression as well: As output value take the average over the labels in the neighborhood.
- ▶ kNN -based algorithms can also be used for many other tasks such as density estimation, outlier detection, clustering, etc.

Summary

- ▶ The kNN classifier is about the simplest classifier that exists.
- ▶ But often it performs quite reasonably.
- ▶ For any type of data, always consider the kNN classifier as a baseline.
- ▶ One can prove that in the limit of infinitely many data points, the kNN classifier is “consistent”, that is it learns the best possible function (see next lecture).

Recap: Maths basics

Recap: Probability theory

Literature:

- ▶ In general, any book on probability theory
- ▶ On the homepage you can also find the link to a probability recap writeup for a CS course at Stanford University (written by Arian Maleki and Tom Do).

Discrete Probability Theory

Discrete probability measure

- ▶ $\Omega = \text{space of "elementary events", "sample space"}$.
This space is called “discrete” if it has finitely many elements.
- ▶ “Space of events”: In the discrete case this is simply the power set $\mathcal{P}(\Omega)$ of Ω , that is all possible subsets of Ω .
(In general it is more complicated, the space of events has to be a “ σ -algebra”).
- ▶ Probability measure: $P : \mathcal{P}(\Omega) \rightarrow [0, 1]$ such that the following three rules are satisfied (“Axioms of Kolmogorov”)
 - ▶ $P(A) \geq 0$ for all events $A \subset \mathcal{P}(\Omega)$
 - ▶ $P(\Omega) = 1$
 - ▶ “sigma-additivity”: Let $S_1, S_2, \dots \subset \Omega$ be at most countably many disjoint sets. Then $P(S_1 \cup S_2 \cup \dots) = \sum_i P(S_i)$

Note: in the discrete case, the probability measure is uniquely defined on all of $\mathcal{P}(\Omega)$ if we know $P(\omega)$ for all elementary events $\omega \in \Omega$.

Discrete probability measure (2)

Example: throwing a dice

- ▶ Elementary events: $\{1, 2, \dots, 6\}$
- ▶ Probability of the elementary events:
 $P(1) = P(2) = \dots = P(6) = 1/6.$
- ▶ Probabilities of all other subsets of Ω can be computed based on the elementary events due to the sigma-additivity.

Example: $P(1, 2, 5) = P(1) + P(2) + P(5) = 3 \cdot 1/6 = 1/2.$

Conditional probabilities

Define the probability of event A under the condition that event B has taken place:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Example with a dice: compute the probability $P(\{3\} \mid \text{"uneven"})$.

Solution:

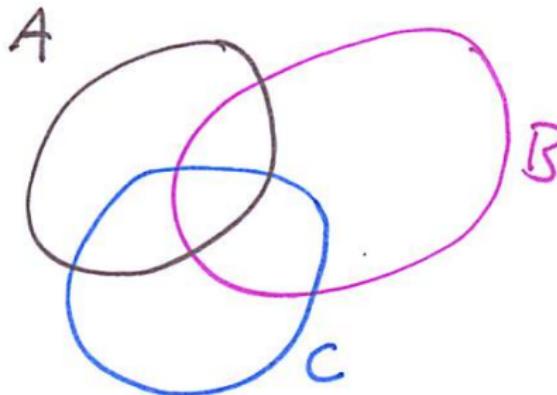
$A = \{3\}$, $B = \{1, 3, 5\}$, $P(A \cap B) = P(\{3\}) = 1/6$, $P(B) = 1/2$,
this implies $P(\{3\} \mid \text{"uneven"}) = (1/6)/(1/2) = 1/3$.

Important formulas

- **Union bound.** Let A_1, \dots, A_k be any events. Then

$$P(A_1 \cup A_2 \cup \dots \cup A_k) \leq \sum_{i=1}^k P(A_i)$$

Intuitive reason:



Important formulas (2)

- ▶ **Formula of total probability.** Let B_1, \dots, B_k be a disjoint decomposition of the probability space, that is all B_i are disjoint and $B_1 \cup \dots \cup B_k = \Omega$. Then:

$$P(A) = \sum_{i=1}^k P(A \mid B_i)P(B_i)$$

Important formulas (3)

- ▶ Bayes' formula:

$$P(B \mid A) = \frac{P(B \cap A)}{P(A)} = \frac{P(A \mid B) \cdot P(B)}{P(A)}$$

Example:

The probability that a woman has breast cancer is 1%. The probability that the disease is detected by a mammography is 80 % (true positive rate). The probability that the test detects the disease although the patient does not have it is 9.6% (false positive rate). If a woman at age 40 is tested as positive, what is the probability that she indeed has breast cancer?

Use Bayes' theorem to obtain 7.8 % only!

Random variables

A **random variable** is a function $X : \Omega \rightarrow \mathbb{R}$.

Example:

- ▶ We have 5 red and 5 black balls in an urn
- ▶ We draw 3 balls randomly without replacement
- ▶ Random variable $X = \text{number of red balls we got}$

A **random variable is called discrete** if its image is discrete (it can take at most finitely many values).

Random variables (2)

A random variable $X : \Omega \rightarrow \mathbb{R}$ induces a probability distribution P_X on its image: for any (measurable) set $A \subset \mathbb{R}$ we define

$$P_X(A) = P(X \in A)$$

The measure P_X is called the **distribution of the random variable**.

Important discrete probability distribution

- ▶ **Bernoulli distribution:** we throw a biased coin once. It takes value 1 with probability p and value 0 with probability $(1 - p)$.
- ▶ **Binomial distribution $B(n, p)$:** We throw a biased coin n times independently from each other. The binomial random variable counts how often we got 1. It is defined as

$$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$$

It has expected value np and variance $np(1 - p)$.

Important discrete probability distribution (2)

- ▶ Poisson distribution $Pois(\lambda)$.

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

The Poisson distribution counts the occurrence of “rare events” in a fixed time interval (like radioactive decay), λ is the intensity parameter.

It has expected value λ and variance λ .

Independence

Two events A, B are called independent if $P(A \cap B) = P(A) \cdot P(B)$.

Note that this implies that $P(A \mid B) = P(A)$.

Two random variables $X, Y : \Omega \rightarrow \mathbb{R}$ are called independent if for all events A, B we have that

$$P(X \in A, Y \in B) = P(X \in A) \cdot P(Y \in B).$$

Example:

- ▶ Throw a coin twice. X = result of the first toss, Y = result of the second toss. These two random variables are independent.
- ▶ Throw a coin twice. X = result of the first toss, Y = sum of the two results. These two random variables are not independent.

Expectation

For a discrete random variable $X : \Omega \rightarrow \{r_1, \dots, r_k\}$ its expectation (mean value) is defined as

$$E(X) := \sum_{i=1}^k r_i \cdot P(\{X = r_i\})$$

Intuition: the expectation is the “average result”, where the results are weighted according to their probabilities.

Examples:

- ▶ We throw a dice, X is the result. Then
$$E(X) = \sum_{i=1}^6 i \cdot \frac{1}{6} = 3.5.$$
- ▶ We throw a biased coin, head comes with probability p , tail comes with probability $1 - p$. We assign the random variable $X = 1$ for head and $X = 0$ for tail. Then
$$E(X) = 0 \cdot (1 - p) + 1 \cdot p = p.$$

Expectation (2)

Important formulas and properties:

- ▶ The expectation is linear: $E(\sum_{i=1}^n a_i X_i) = \sum_{i=1}^n a_i E(X_i)$
- ▶ Expectation and independence: If X, Y are independent, then $E(X \cdot Y) = E(X) \cdot E(Y)$.

Variance

For a discrete random variable, its variance is defined as

$$\begin{aligned}\text{Var}(X) &= \sum_i (r_i - E(X))^2 \cdot P(r_i) \\ &= E((X - E(X))^2)\end{aligned}$$

The variance measures how much the random variable “varies” about its mean.

Example:

- ▶ We throw a biased coin, head comes with probability p , tail comes with probability $1 - p$. We assign the random variable $X = 1$ for head and $X = 0$ for tail.
- ▶ We have already seen: $E(X) = p$.

Variance (2)

- Now let's compute the variance:

$$\text{Var}(X) = (1 - p)^2 p + (0 - p)^2 (1 - p) = (1 - p)p$$

Important properties of the variance:

- If X, Y are independent random variables, then

$$\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y).$$

Important inequalities

- ▶ **Markov's inequality:** Let X be a non-negative random variable and $t > 0$. Then

$$P(X \geq t) \leq \frac{E(X)}{t}$$

- ▶ **Chebyshev's inequality:**

$$P(|X - E(X)| \geq t) \leq \frac{\text{Var}(X)}{t^2}$$

Joint, marginal and product distribution

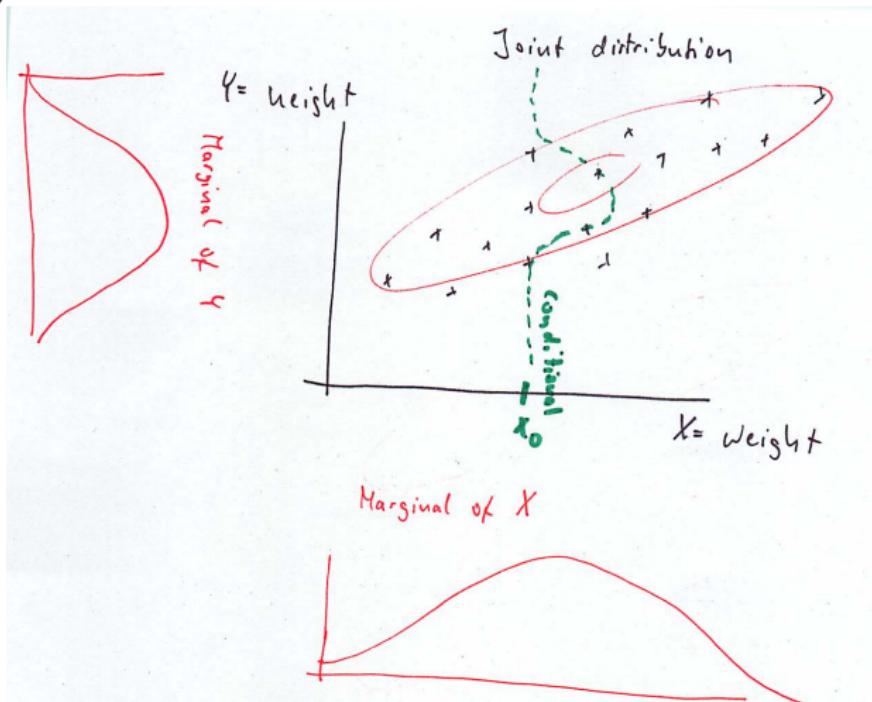
We want to look at the “joint distribution” of two random variables.

Example:

- ▶ We “sample” people: $\Omega = \text{set of all people}$
- ▶ $X = \text{their weight (in kg)}$, $Y = \text{their height (in cm)}$.
- ▶ The **joint distribution** measures how the pair of random variables $(X, Y) : \Omega \rightarrow \mathbb{R}^2$ is distributed.

Joint, marginal and product distribution (2)

- The distribution of X is called the **marginal distribution** of X , similarly for Y .



Joint, marginal and product distribution (3)

- ▶ Note that for given marginal distributions, there exist many joint distributions that respect the marginals!

Joint, marginal and product distribution (4)

A particular joint distribution is the **product distribution**: it gives the joint distribution of X and Y if they are independent of each other:

- ▶ Consider two discrete random variables $X, Y : \Omega \rightarrow \mathbb{R}$.
- ▶ Define the product distribution
$$P((X, Y) = (x, y)) = P(X = x) \cdot P(Y = y).$$

The construction works analogously for a product of finitely many spaces.

Conditional independence

Consider three discrete random variables $X, Y, Z : \Omega \rightarrow \mathbb{R}$. We say that X and Y are conditionally independent given Z if

$$P(X \in A, Y \in B \mid Z \in C) = P(X \in A \mid Z \in C) \cdot P(Y \in B \mid Z \in C)$$

for all sets $A, B, C \subset \Omega$ with $P(Z \in C) > 0$.

Conditional expectation

Example:

- ▶ X, Y two independent throws of a dice, $Z = X + Y$.
- ▶ Want to compute the expectation of Z under the condition that X was 3.
- ▶ We write $E(Z \mid X = 3)$

If we don't fix the outcome value of X , then we write $E(Z \mid X)$, this is a random variable (because we don't know the random outcome of X).

Formally, this is a pretty complicated mathematical object. For those who have not seen it before, we just treat it in an intuitive manner.

Continuous probability theory

Probability theory gets more complicated once we go beyond the discrete regime. In this class, we try to keep it on a somewhat intuitive level.

Density and cumulative distribution

Consider a random variable $X : \Omega \rightarrow \mathbb{R}$. We say that X has **density function** $p : \mathbb{R} \rightarrow \mathbb{R}$ if for all (measurable) subsets $A \subset \mathbb{R}$ we have

$$P(X \in A) = \int_A p(x) dx$$



Density and cumulative distribution (2)

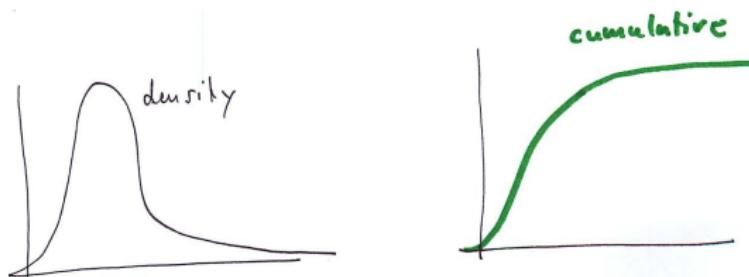
- ▶ Intuitively, a density is something like a “continuous histogram”.
- ▶ Sometimes the density is abbreviated as “pdf” (“probability density function”) in the literature.
- ▶ Density functions are always non-negative and integrate to 1. They don't have to be continuous.
- ▶ Not every random variable can be described by a density, but in this course we won't discuss this.

Cumulative distribution function

A real-valued random variable can always be described by its **cumulative distribution function** (sometimes abbreviated as “cdf” in the literature).

For a random variable $X : \Omega \rightarrow \mathbb{R}$ it is defined as

$$g : \mathbb{R} \rightarrow \mathbb{R}, \quad g(x) = P(X \leq x)$$

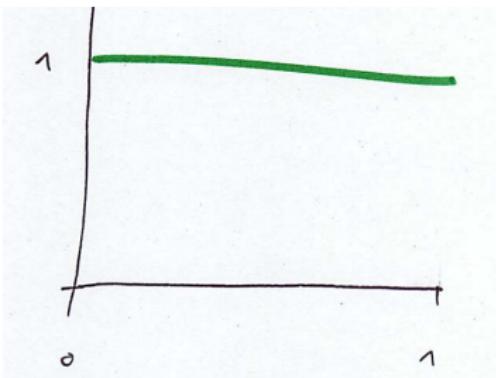


Uniform distribution

The uniform distribution on $[0, 1]$: for $0 \leq a < b \leq 1$ we define

$$P(X \in [a, b]) = b - a$$

Its density is constant:



Normal distribution

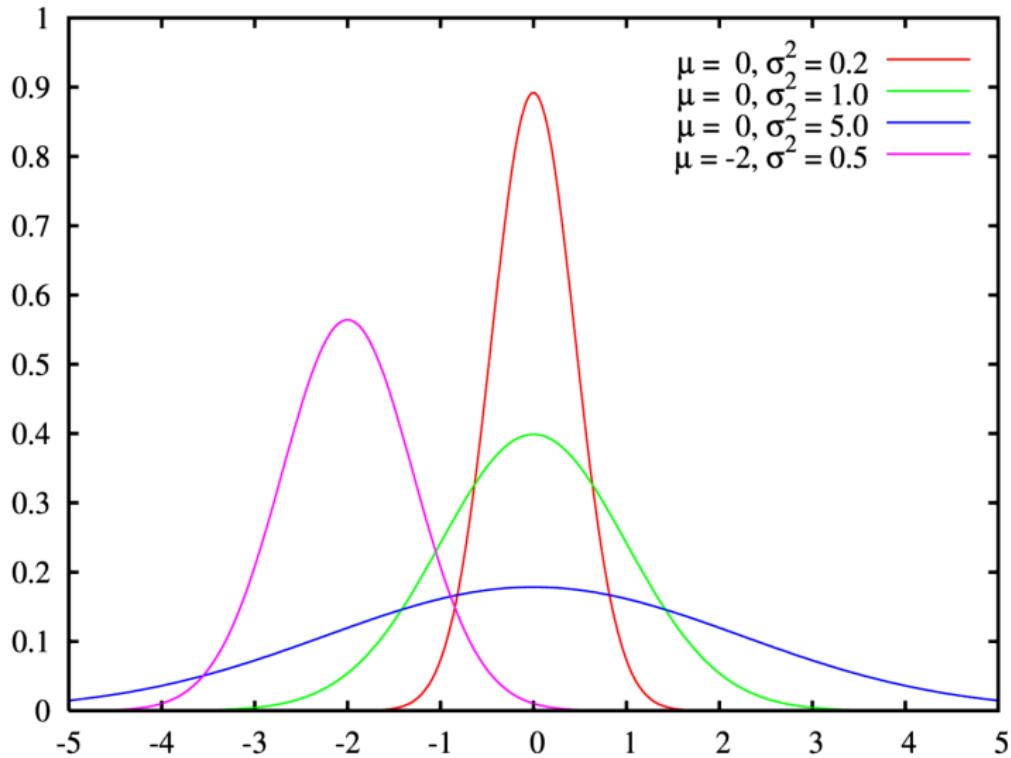
The most important continuous distribution is the normal distribution, abbreviated $N(\mu, \sigma^2)$.

- ▶ It has two parameters: its expectation μ and its variance σ^2 .
- ▶ The density function of $N(\mu, \sigma^2)$ is given as

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- ▶ The special case of mean 0 and variance 1 is called the “standard normal distribution”.

Normal distribution (2)



Expectation

In the continuous domain, sums are going to be replaced by integrals. For example, the expectation of a random variable X with density function $p(x)$ is defined as

$$E(X) = \int_{\mathbb{R}} x \cdot p(x) dx$$

Recap: Linear algebra

Literature:

- ▶ In general, any introductory book on linear algebra
- ▶ On the homepage you can also find the link to a short linear algebra recap writeup (by Zico Kolter and Chuong Do).

Vector space

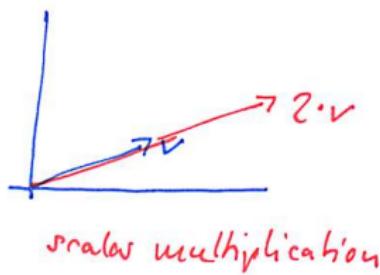
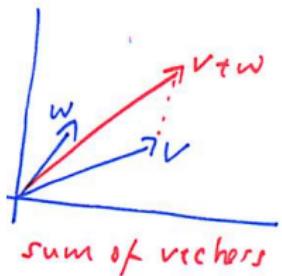
A **vector space** V is a set of “vectors” that supports the following operations:

- ▶ We can add and subtract vectors: For $v, w \in V$ we can build $v + w, v - w$
- ▶ We can multiply vectors with scalars: For $v \in V, a \in \mathbb{R}$ we can build av .
- ▶ These operations satisfy all kinds of formal requirements (associativity, commutativity, identity element, inverse element, and so on).

Vector space (2)

Most prominent example: $V = \mathbb{R}^d$.

- ▶ $(v_1, \dots, v_d) + (w_1, \dots, w_d) := (v_1 + w_1, \dots, v_d + w_d)$
- ▶ $a(v_1, \dots, v_d) := (av_1, \dots, av_d)$.



Basis

A **basis** of a vector space is a set of vectors $b_1, \dots, b_d \in V$ that satisfies two properties:

- ▶ Any vector in V can be written as a linear combination of basis vectors:

For any $v \in V$ there exist $a_1, \dots, a_d \in \mathbb{R}$ such that

$$v = \sum_{i=1}^d a_i b_i$$

- ▶ The vectors in the basis cannot be expressed in terms of each other, they are linearly independent:

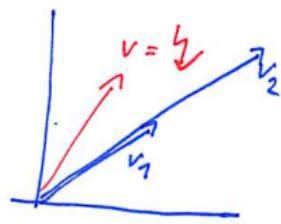
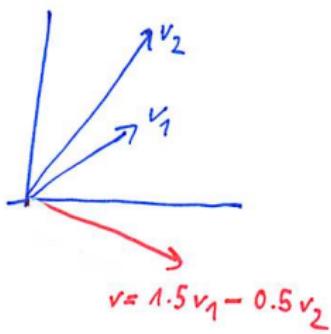
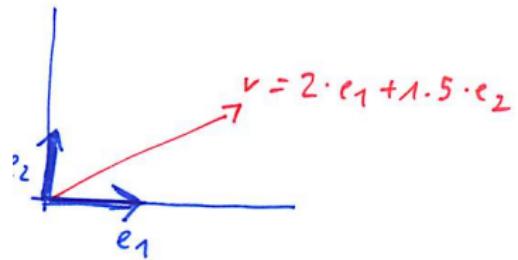
$$\sum_{i=1}^d a_i b_i = 0 \implies a_i = 0 \text{ for all } i = 1, \dots, d.$$

The number of vectors in a basis is called the **dimension** of the vector space.

Basis (2)

Example:

- $e_1 := (1, 0)$ and $e_2 := (0, 1)$ form a basis of \mathbb{R}^2
- $v_1 := (1, 1)$ and $v_2 := (1, 2)$ form a basis of \mathbb{R}^2
- $v_1 := (1, 1)$ and $v_2 := (2, 2)$ do not form a basis of \mathbb{R}^2 .



Linear mappings

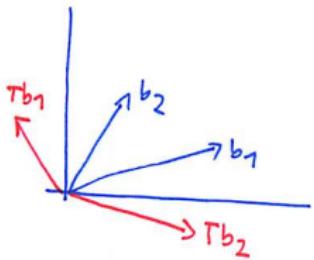
A **linear mapping** $T : V \rightarrow V$ satisfies

$$T(av_1 + bv_2) = aT(v_1) + bT(v_2) \text{ for all } a, b \in \mathbb{R}, v_1, v_2 \in V.$$

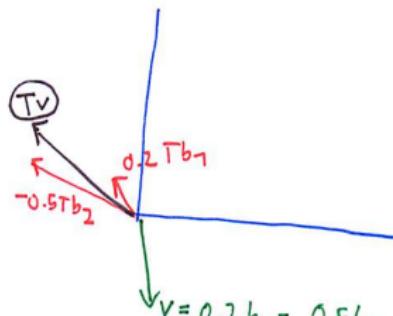
Typical linear mappings are: stretching, rotation, projections, etc., and combinations thereof.

Note: to figure out what a linear mapping does, it is enough to know what it does on the basis vectors: for $v = \sum_i a_i b_i$ we know by linearity that $T(v) = T(\sum_i a_i b_i) = \sum_i a_i T(b_i)$

Linear mappings (2)



Basis vectors and their
images



Linear mappings (3)

Linear mappings correspond to **matrices**:

Intuition: the columns of the matrix contain the images of the basis vectors:

Basis e_1, \dots, e_d , linear mapping T

~ corresponding matrix: $\begin{pmatrix} | & | & | \\ T_{e_1} & T_{e_2} & \dots & T_{e_d} \\ | & | & | \end{pmatrix} =: A$

- ▶ Matrix-vector multiplication is then the same as applying the mapping to the vector.
- ▶ Multiplication of two matrices is the same as applying the mappings one after the other.

Norms and scalar products

Some vector spaces have additional structure: norms or even scalar products. In particular, this is true for \mathbb{R}^d .

Given two vectors $v = (v_1, \dots, v_n)^t$ and $w = (w_1, \dots, w_n)^t \in \mathbb{R}^n$, their **scalar product** is defined as $\langle v, w \rangle = \sum_{i=1}^n v_i w_i$.

The **norm** $\|v\|$ of a vector $v \in \mathbb{R}^d$ is defined as $\|v\|^2 = \langle v, v \rangle$.

Intuition:

- ▶ The scalar product is related to the angle between the two vectors:
 - ▶ $\langle v, w \rangle = 0 \iff v \perp w$ (vectors are orthogonal)
 - ▶ If v and w have norm 1, then $\langle v, w \rangle$ is the cosine of the angle between the two vectors.
- ▶ The norm is the length of a vector.

Norms and scalar products (2)

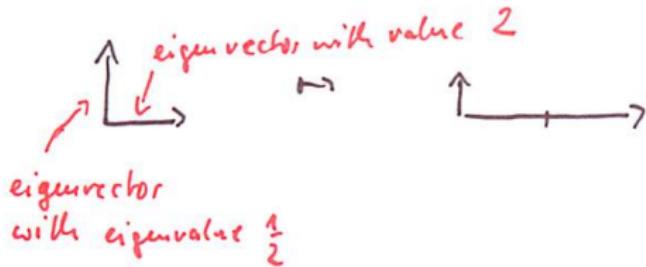
A matrix A is called **orthogonal** if all its columns are orthogonal to each other. It is called **orthonormal** if additionally, all its columns have norm 1.

For orthogonal matrices, we always have $A^t = A^{-1}$.

Eigenvalues and eigenvectors

A vector $v \in \mathbb{R}^n$, $v \neq 0$ is called an **eigenvector** of $A \in \mathbb{R}^{n \times n}$ with **eigenvalue** λ if $Av = \lambda v$.

Intuition: in the direction of v , the linear mapping corresponding to A is stretching by factor λ .



Taken together, all eigenvectors with eigenvalue λ form a subspace called the **eigenspace** associated to eigenvalue λ . The dimension of this subspace is called the **geometric multiplicity** of λ .

Eigenvalues and eigenvectors (2)

Just for completeness:

Eigenvectors can also be defined as the roots of the **characteristic polynomial** $f(\lambda) := \det(A - \lambda I) = 0$. The degree of this polynomial is d (the dimension of the space). The multiplicity of this root is called the **algebraic multiplicity of λ** .

We always have that the algebraic multiplicity is larger or equal to the geometric one. In case of strict inequality, the matrix cannot be diagonalized.

Simple example where the two multiplicities do not agree:
the nilpotent matrix $[0, 1; 0, 0]$ has eigenvalue 0 with geometric multiplicity 1, but algebraic multiplicity 2. It cannot even be diagonalized over \mathbb{C} .

Eigenvalue decomposition of a symmetric matrix

Diagonalization:

- ▶ A matrix A is called **diagonalizable** if there exists a basis of eigenvectors.
- ▶ In this case, we can write the matrix in the form

$$A = VDV^t$$

where V is an orthonormal matrix that contains the eigenvectors as columns, and D is a diagonal matrix.

- ▶ Intuitively, a matrix is diagonalizable if it performs “Strecken und Spiegeln”, but no rotation.

Eigenvalue decomposition of a symmetric matrix (2)

Symmetric matrices are always diagonalizable and have real-valued eigenvalues.

Eigenvectors of different eigenvalues are always perpendicular to each other (only true for symmetric matrices).

Positive Definite Matrices

A symmetric matrix A is called **positive semi-definite** if all its eigenvalues are ≥ 0 . In case of strict inequality it is called **positive definite**.

Being positive definite is equivalent to the condition that $v^t A v > 0$ for all $v \in \mathbb{R}^n \setminus \{0\}$.

(similarly for positive semi-definite with \geq)

Singular Value Decomposition (SVD)

If a matrix is not square, we cannot compute eigenvalues. But there exists a closely related concept, the singular values:

Any matrix $\Phi \in \mathbb{R}^{n \times d}$ can be decomposed as follows:

$$\Phi = U\Sigma V^t$$

where $U \in \mathbb{R}^{n \times n}$ is orthogonal, $\Sigma \in \mathbb{R}^{n \times d}$ is a diagonal matrix containing the singular values $\sigma_1, \dots, \sigma_d$ on the diagonal, and $V \in \mathbb{R}^{d \times d}$ is an orthogonal matrix.

The columns of U (and V , respectively) are called the left (and right) singular vectors, the entries of Σ are called the singular values.

Singular Value Decomposition (SVD) (2)

There is a close relation between the singular values of Φ and the eigenvalues of the (symmetric!) matrices $\Phi\Phi^t$ and $\Phi^t\Phi$:

- ▶ The left singular vectors of Φ are the eigenvectors of $\Phi\Phi^t$.
- ▶ The right singular vectors of Φ are the eigenvectors of $\Phi^t\Phi$.
- ▶ The non-zero singular values of Φ are the square roots of the non-zero eigenvalues of both $\Phi^t\Phi$ and $\Phi\Phi^t$.

Inverse and generalized inverse

Consider a symmetric matrix $A \in \mathbb{R}^{d \times d}$.

- Let $\lambda_1, \dots, \lambda_d$ the eigenvalues and v_1, \dots, v_d a corresponding set of eigenvectors of A . We can write A in the spectral decomposition as

$$A = \sum_{i=1}^d \lambda_i v_i v_i^t$$

- In case the matrix has rank d , all its eigenvalues are non-zero. Then we can write the inverse of A as

$$A^{-1} = \sum_{i=1}^d \frac{1}{\lambda_i} v_i v_i^t$$

Inverse and generalized inverse (2)

- ▶ In case the matrix is not of full rank, it is not invertible. However, we can define the **generalized inverse** as

$$A^+ := \sum_{i:\lambda_i \neq 0} \frac{1}{\lambda_i} v_i v_i^t$$

(intuitively, this is the inverse of the matrix A restricted to the subspace orthogonal to its nullspace).

Inverse and generalized inverse (3)

Properties of the generalized inverse:

In general we don't have that $AA^+ = Id$ or $A^+A = Id$.

But we have the following slightly weaker properties:

- ▶ $AA^+A = A$ and $A^+AA^+ = A^+$
- ▶ $(A^+)^+ = A$
- ▶ A^+A and AA^+ are both symmetric.
- ▶ If A is invertible, then $A^{-1} = A^+$.
- ▶ AA^+ is an orthogonal projection on the $\text{ran}(A)$ (the image of the matrix A), and A^+A is an orthogonal projection on $\text{ran}(A^t)$.

Rayleigh principle

Theorem 1 (Rayleigh principle)

Let $A \in R^{n \times n}$ be a symmetric matrix with eigenvalues $\lambda_1 \geq \dots \geq \lambda_n$ and eigenvectors v_1, \dots, v_n . Then

$$\lambda_1 = \max_{v \in \mathbb{R}^n} \frac{v^t A v}{\|v\|^2} = \max_{v \in \mathbb{R}^n : \|v\|=1} v^t A v.$$

The eigenvector v_1 is the vector for which this maximum is attained. Moreover,

$$\lambda_{k+1} = \max_{v \perp \{v_1, \dots, v_k\} : \|v\|=1} v^t A v.$$

This theorem holds analogously for minimization problems. In this case, the solution is given by the smallest eigenvalue / vector.

Rayleigh principle (2)

Proof intuition.

- ▶ Let λ be any eigenvalue with eigenvector v . Then $v^t A v = v^t (\lambda v) = \lambda$ (because $v^t v = 1$).
- ▶ So among all eigenvectors v_1, \dots, v_n , the eigenvector v_1 leads to the largest value λ_1 .
- ▶ Now consider an arbitrary unit vector $w \in \mathbb{R}^n$. Because A is symmetric, there exists a basis of eigenvectors v_1, \dots, v_n . In particular, there exist coefficients c_i such that $w = \sum_i c_i v_i$ and $\|c\| = 1$.
- ▶ Then $w^t A w = \dots = \sum_{i,j=1}^n c_i c_j v_i^t A v_j$
- ▶ But for $i \neq j$ we get $v_i^t A v_j = v_i^t \lambda v_j = 0$ (because different eigenvectors are perpendicular to each other).
- ▶ so $w^t A w = \sum_i c_i^2 v_i^t A v_i = \sum_i c_i^2 \lambda_i$.

Rayleigh principle (3)

- ▶ Among all c with $\|c\| = 1$, the maximum of this expression is attained for $c_1 = 1, c_2 = \dots = c_n = 0$.



Supervised learning

Formal setup

Statistical and Bayesian Decision theory

Literature:

- ▶ Hastie, Section 2.4 - 2.9 (parts only)
- ▶ Devroye, Section 2
- ▶ Duda/Hart, Section 2 (only parts of it, very technical)

The statistical setup

- ▶ Let \mathcal{X} and \mathcal{Y} be some input and output space.
Our goal will be to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$.
- ▶ Consider a **probability distribution P** over $\mathcal{X} \times \mathcal{Y}$. All training and test points are going to be sampled according to this probability distribution. P is the joint distribution over (input,output)-pairs.
- ▶ In particular, note that the output values Y_i are not necessarily deterministic functions of X_i . Instead, the output can be non-deterministic or noisy.

The statistical setup (2)

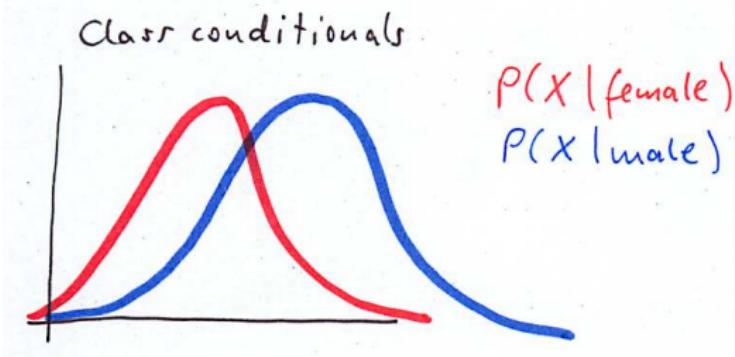
Example:

- ▶ You want to classify people as “male” or “female” based on their height.
 - ▶ X = height of the person, Y is the label “male” or “female”.
 - ▶ Obviously, the label cannot be deterministic (there exist both men and women of height 170 cm, say).
-
- ▶ At the time of learning, the probability distribution P is unknown, all we get are samples $(X_i, Y_i)_{i=1,\dots,n}$ from this distribution.

Bayesian decision theory, intuitively

Bayesian decision theory: intuition

Let's continue with the example to classify "male" vs. "female". We consider a couple of different approaches.



GIVEN THIS INFORMATION, HOW WOULD YOU LABEL THE INPUT $X = 160$?

Bayesian decision theory: intuition (2)

Decide based on prior probabilities $P(Y)$.

- ▶ If you don't have any clue what to do, you could simply use the following rule:
You always predict the label of the "larger class", that is

$$f_n(X) = \begin{cases} 1 & \text{if } P(Y = 1) > P(Y = 0) \\ 0 & \text{otherwise} \end{cases}$$

- ▶ This rule is independent of X , its probability to err is $P(\text{err}) = \min\{P(Y = 1), P(Y = 0)\}$.

Bayesian decision theory: intuition (3)

Decide based on the likelihood functions $P(X|Y)$ (maximum likelihood approach).

- ▶ Compute the class conditional distributions $P(X|Y=1)$ and $P(X|Y=0)$.
- ▶ Then predict the label with the higher likelihood:

$$f_n(x) = \begin{cases} 1 & \text{if } P(X=x|Y=1) > P(X=x|Y=0) \\ 0 & \text{otherwise} \end{cases}$$

Bayesian decision theory: intuition (4)

Decide based on the posterior distributions $P(Y|X)$ ("Bayesian maximum a posteriori approach"):

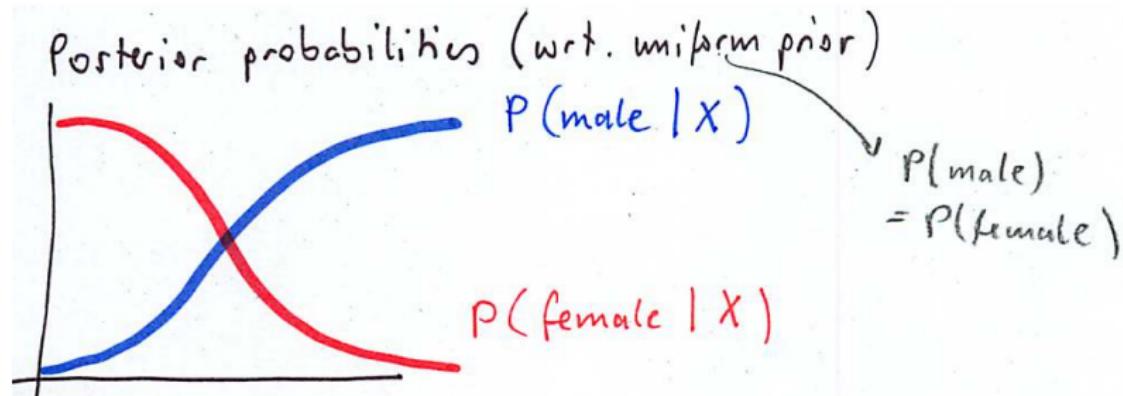
- ▶ Compute the posterior probabilities

$$P(Y = 1|X = x) = \frac{P(X = x|Y = 1) \cdot P(Y = 1)}{P(X = x)}$$

- ▶ Predict by the following rule:

$$f_n(x) = \begin{cases} 1 & \text{if } P(Y = 1|X = x) > P(Y = 0|X = x) \\ 0 & \text{otherwise} \end{cases}$$

Bayesian decision theory: intuition (5)



- ▶ Probability of error is

$$P(\text{error} | X = x) = \min\{P(Y = 1 | X = x), P(Y = 0 | X = x)\}.$$

Bayesian decision theory: intuition (6)

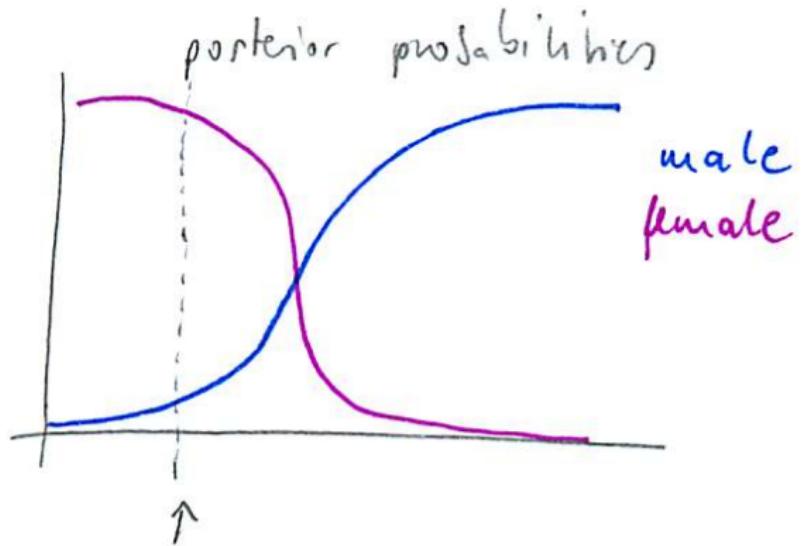
Take the “costs” of errors into account:

- ▶ Define a loss function $\ell(f_n(X) = a_i | Y = y_i)$ that specifies the costs of predicting a_i when the true label is y_i
- ▶ Consider the expected conditional risk

$$\begin{aligned} R(a_i | X = x) &= \ell(a_i | Y = 1)P(Y = 1 | X = x) \\ &\quad + \ell(a_i | Y = 0)P(Y = 0 | X = x) \end{aligned}$$

- ▶ Use Bayes decision rule: Select the label $f_n(X)$ for which the conditional risk is minimal.

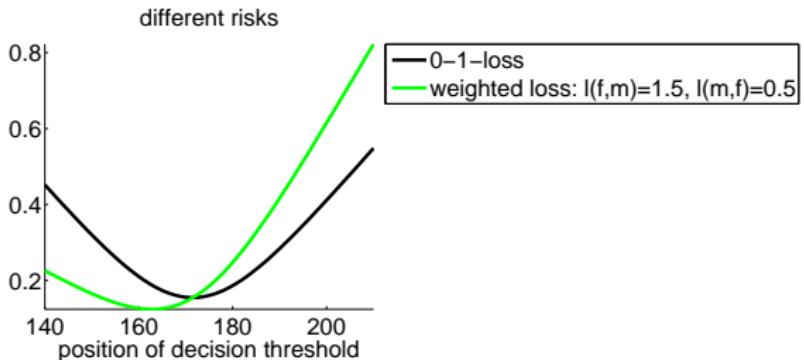
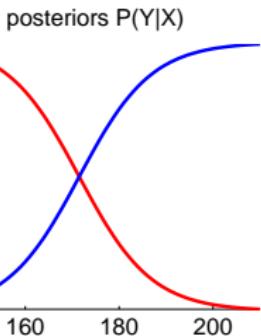
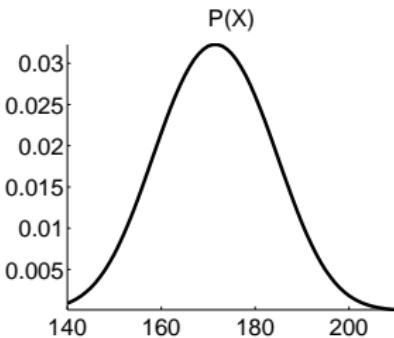
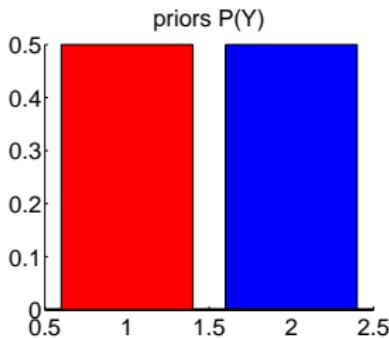
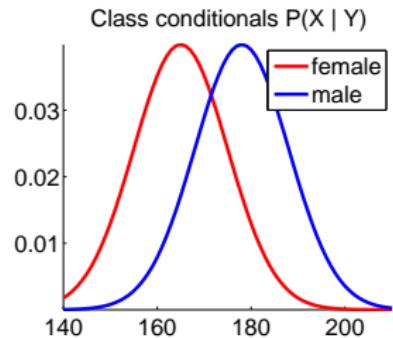
Bayesian decision theory: intuition (7)



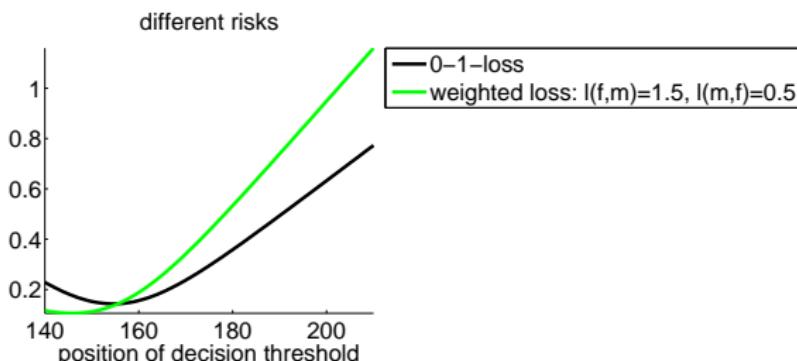
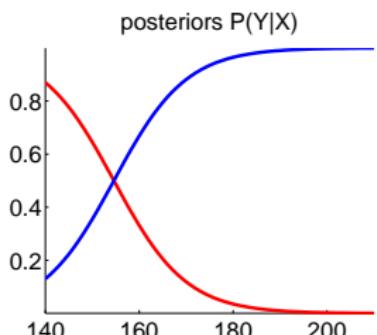
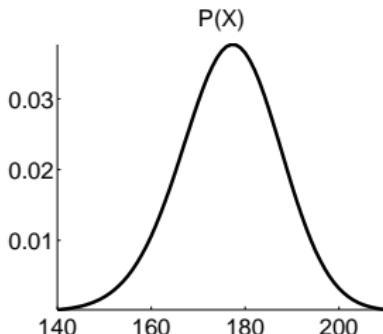
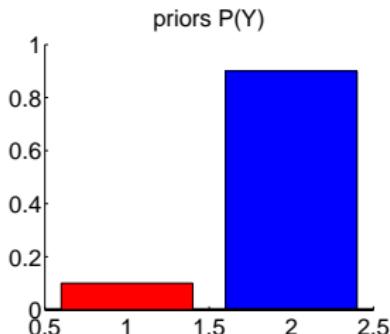
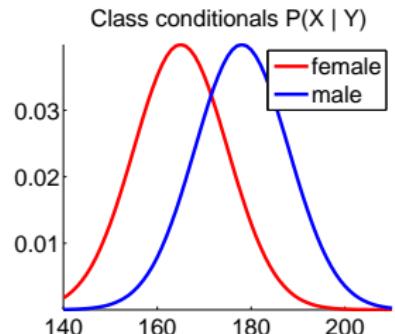
Risk optimal decision threshold if

$$\ell(\text{male} | \text{female}) > \ell(\text{female} | \text{male})$$

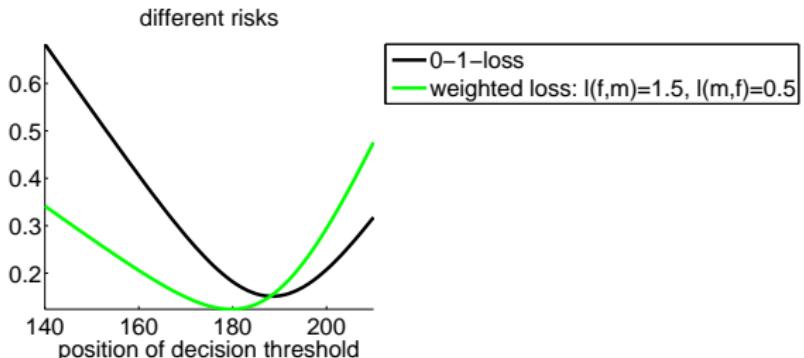
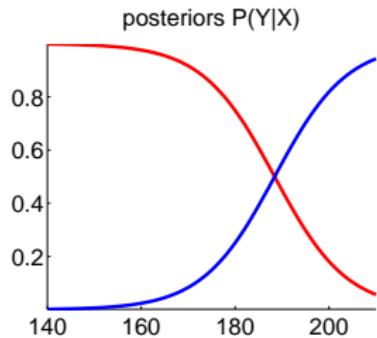
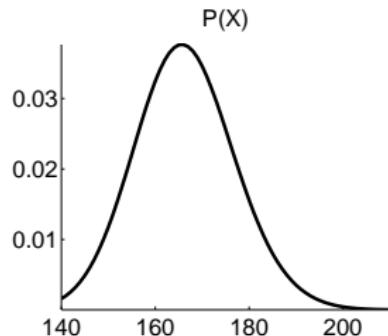
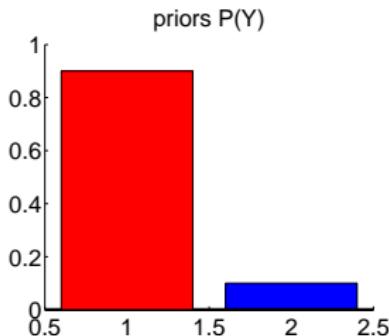
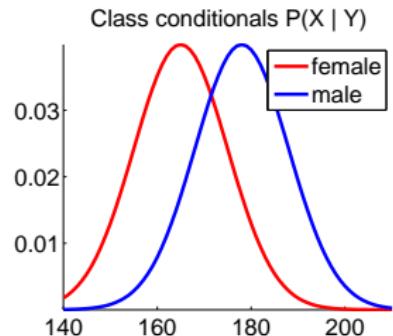
Bayesian decision theory: intuition (8)



Bayesian decision theory: intuition (9)



Bayesian decision theory: intuition (10)



Formal setup

Loss function

The loss function measures how “expensive” an error is:

A **loss function** is a function $\ell : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$.

Example:

- The **0-1-loss function** for classification is defined as

$$\ell(x, y, y') = \begin{cases} 0 & \text{if } y = y' \\ 1 & \text{otherwise} \end{cases}$$

- The **squared loss** for regression is defined as

$$\ell(x, y, y') = (y - y')^2$$

- In some applications, it is important that the loss also depends on x or the order of y and y' (for example, in a medical context)

True Risk

The **true risk** (or **true expected loss**) of a learning rule $f : \mathcal{X} \rightarrow \mathcal{Y}$ (with respect to loss function ℓ) is defined as

$$R(f) := E(\ell(X, f(X), Y))$$

where the expectation is over the random draw of (X, Y) according to the probability distribution P on $\mathcal{X} \times \mathcal{Y}$.

Bayes risk

What is the best function we can think of?

- The Bayes risk is defined as

$$R^* := \inf_f \{ R(f) \mid f \text{ measurable} \}$$

(we won't discuss measurability, if you've never heard of it then simply assume that f can be any function you want).

- In case the infimum is attained, the corresponding function f^* is called the Bayes learner.

The goal of learning is to construct a function f_n that has true risk close to the Bayes risk, that is $R(f_n) \approx R^*$

Consistency

Consider an infinite sequence of data points $(X_i, Y_i)_{i \in \mathbb{N}}$ that have been drawn i.i.d. from distribution P over $\mathcal{X} \times \mathcal{Y}$. Denote by f_n the learning rule that has been constructed by an algorithm \mathcal{A} based on the first n training points.

- We say that the algorithm \mathcal{A} is **consistent** (for probability distribution P) if

$$\lim_{n \rightarrow \infty} R(f_n) = R^*$$

(where the convergence is in probability, for those who know what that means).

- We say that algorithm \mathcal{A} is **universally consistent** if it is consistent for all possible probability distributions P over $\mathcal{X} \times \mathcal{Y}$.

Consistency (2)

Ultimately, what we want to find are learning algorithms that are universally consistent: No matter what the underlying probability distribution is, when we have seen “enough data points”, then the true risk of our learning rule f_n will be arbitrarily close to the best possible risk.

Consistency (3)

For quite some time it was unknown whether universally consistent algorithms can exist. The first positive answer was only in 1977 when Stone proved that the kNN classifier is universally consistent.

Since then quite a number of algorithms have been found to be universally consistent, among them support vector machines, boosting, and many more. Understanding the underlying principles behind these algorithms is the focus of this course.

Two principles

There are two major approaches to supervised learning:

- ▶ Empirical risk minimization (ERM)
- ▶ Regularized risk minimization (RRM)

Empirical risk minimization

As we don't know P we cannot compute the true risk. But we can compute the **empirical risk** based on a sample $(X_i, Y_i)_{i=1,\dots,n}$

$$R_n(f) := \frac{1}{n} \sum_{i=1}^n \ell(X_i, Y_i, f(X_i))$$

The key point is that the empirical risk can be computed based on the training points only.

Empirical risk minimization (2)

Empirical risk minimization approach:

- ▶ Define a set \mathcal{F} of functions from $\mathcal{X} \rightarrow \mathcal{Y}$.
- ▶ Within these functions, choose the one that has the smallest empirical risk:

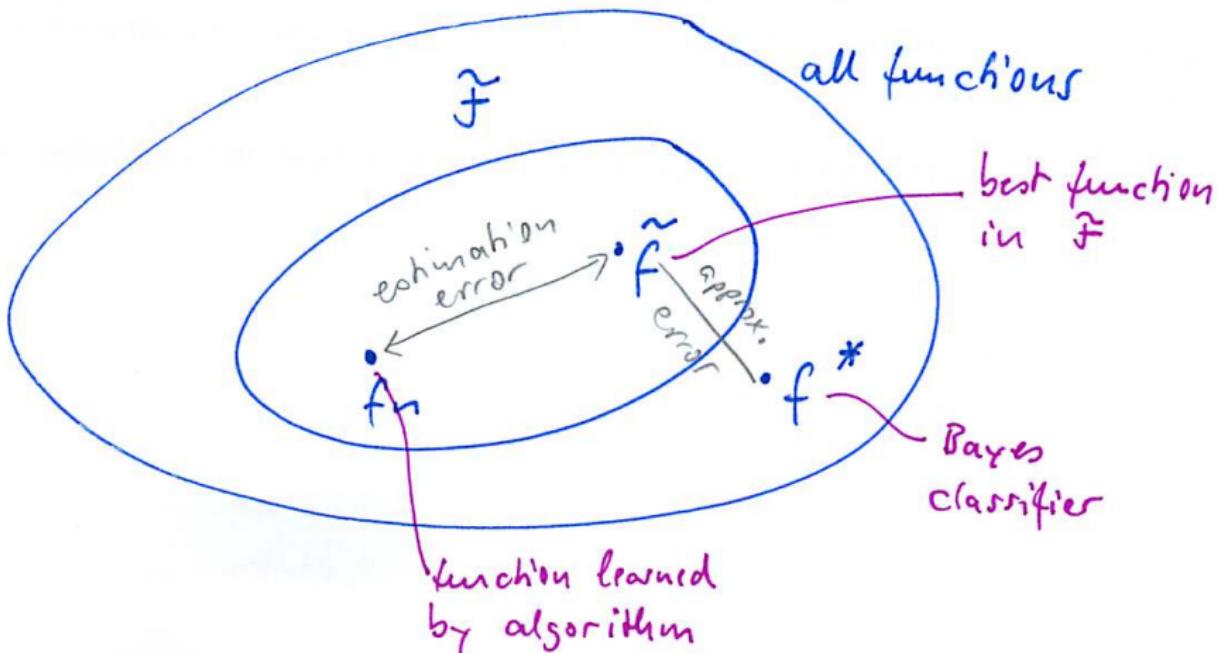
$$f_n := \operatorname{argmin}_{f \in \mathcal{F}} R_n(f)$$

Empirical risk minimization (3)

With this approach, we can make two types of error:

- ▶ Denote by \tilde{f} the true best function in the set \mathcal{F} , that is $\tilde{f} = \operatorname{argmin}_{f \in \mathcal{F}} R(f)$.
- ▶ The quantity $R(f_n) - R(\tilde{f})$ is called the **estimation error**. It is a random variable that depends on the random sample.
- ▶ The quantity $R(\tilde{f}) - R(f^*)$ is called the **approximation error**. It is a deterministic quantity that does not depend on the sample, but on the choice of the space \mathcal{F} .

Empirical risk minimization (4)



Empirical risk minimization (5)

Coming back to the words underfitting and overfitting:

- ▶ Underfitting happens if \mathcal{F} is too small. In this case we have a small estimation error but a large approximation error.
- ▶ Overfitting happens if \mathcal{F} is too large. Then we have a high estimation error but a small approximation error.

Empirical risk minimization (6)

Remarks:

- ▶ From a conceptual/theoretical side, ERM is a straight forward learning principle.
- ▶ The key to the success / failure of ERM is to choose a “good” function class \mathcal{F}
- ▶ From the computational side, it is not always easy (depending on function class and loss function, the problem can be quite challenging: finding the minimizer of the 0-1-loss is often NP hard. This is why in practice we use convex relaxations of the 0-1-loss function, see later.

Regularized risk minimization

Crucial problem in ERM: choose \mathcal{F}

Alternative approach:

- ▶ Let \mathcal{F} be a very large space of functions.
- ▶ Define a **regularizer** $\Omega(f)$ that measures how “complex” a function is. Examples:
 - ▶ $\Omega = \text{degree of the polynomial}$
 - ▶ $\Omega = \text{maximal slope of a differentiable function}$
- ▶ Then choose $f \in \mathcal{F}$ to minimize the **regularized risk**

$$R_{\text{reg},n}(f) := R_n(f) + \lambda \cdot \Omega(f)$$

Regularized risk minimization (2)

Intuition:

- ▶ If I can fit the data reasonably well with a “simple function”, then choose such a simple function.
- ▶ If all simple functions lead to a very high empirical risk, then better choose a more complex function.

Particular setup for classification

We now consider the particular case of binary classification, that is the case that $\mathcal{Y} = \{0, 1\}$.

For simplicity we also assume in this section that $\mathcal{X} = \mathbb{R}^d$, but the results also hold more generally.

Regression function

Consider a probability distribution over $\mathcal{X} \times \{0, 1\}$. We want to describe this distribution in terms of two other quantities:

- ▶ Let μ be the marginal distribution of X , that is $\mu(A) = P(X \in A)$.
- ▶ Define the function

$$\eta(x) = P(Y = 1 \mid X = x)$$

It is called the “regression function”.

Regression function (2)

Theorem 2 (Unique decomposition)

The probability distribution P is uniquely determined by μ and η .

Proof. Any set $C \subset \mathcal{X} \times \{0, 1\}$ can be written of the form $C_0 \times \{0\} \cup C_1 \times \{1\}$. Now:

Regression function (3)

$$\mathbb{P}((X, Y) \in C) = \mathbb{P}(X \in C_0, Y=0) + \underbrace{\mathbb{P}(X \in C_1, Y=1)}$$

■

Intuition

" $\sum_{x \in X} \mathbb{P}(Y=1 \mid X=x) \cdot \mathbb{P}(X=x)$ "

$$= \int_{C_0} (1 - \eta(x)) d\mu(x) + \overbrace{\int_{C_1} \eta(x) d\mu(x)}$$



The Bayes classifier

Consider the 0-1-loss function. Note that the risk of a classifier under the 0-1-loss counts “how often” the classifier fails, that is

$$R(f) = P(f(X) \neq Y)$$

Now let us define the following classifier, called the **Bayes classifier**:

$$f^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geq 1/2 \\ 0 & \text{otherwise} \end{cases}$$

The Bayes classifier (2)

Theorem 3 (Bayes classifier is optimal)

Let $f : \mathcal{X} \rightarrow \{0, 1\}$ be any (measurable) decision function. Then $R(f) \geq R(f^*)$.

Intuitively, this theorem means that all we need to know for classification is the function $\eta \dots$

Proof.

The Bayes classifier (3)

For every $x \in \mathcal{X}$,

$$P(g(x) \neq Y \mid X=x) = 1 - P(Y=g(x) \mid X=x) \quad \leftarrow \text{split into } g(x)=1, g(x)=0$$

$$= 1 - P(Y=1, g(x)=1 \mid X=x) - P(Y=0, g(x)=0 \mid X=x)$$

$$= 1 - \underbrace{\prod_{g(x)=1} P(Y=1 \mid X=x)}_{\eta(x)} - \underbrace{\prod_{g(x)=0} P(Y=0 \mid X=x)}_{1-\eta(x)}$$

[now note
that $g(x)$ is
deterministic]

The Bayes classifier (4)

Hence,

$$\begin{aligned}
 & P(g(x) \neq Y | X=x) = P(g^*(x) \neq Y | X=x) \\
 &= \eta(x) (\mathbb{1}_{g^*(x)=1} - \mathbb{1}_{g(x)=1}) + (1-\eta(x)) (\mathbb{1}_{g^*(x)=0} - \mathbb{1}_{g(x)=0}) \\
 &= (2\eta(x)-1) (\mathbb{1}_{g^*(x)=1} - \mathbb{1}_{g(x)=1}) \\
 &\geq 0 \quad \text{because:} \quad \bullet \text{ either } g^*(x)=1, \text{ then } \eta(x) > \frac{1}{2}, \text{ hence } 2\eta(x)-1 \geq 0 \\
 &\quad \bullet \text{ or } g^*(x)=0, \text{ then both terms get negative.}
 \end{aligned}$$



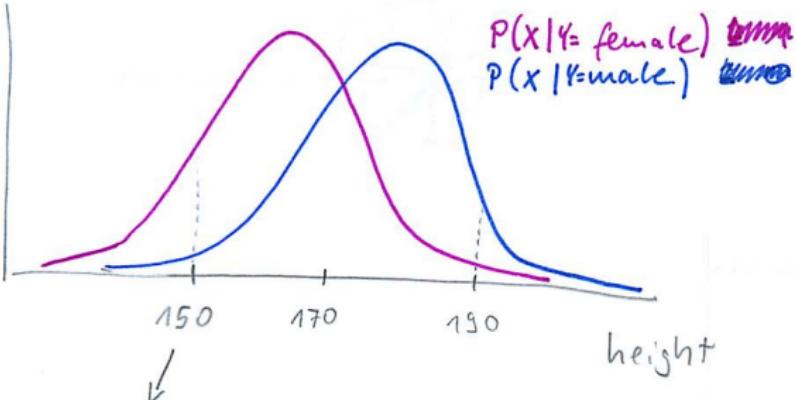
The Bayes classifier (5)

We call f^* the **Bayes classifier** and $R^* := R(f^*)$ the **Bayes risk**.

Some intuition on regression function and Bayes classifier:

- ▶ Consider the example of predicting the sex of a person based on its height. The distributions roughly look as follows:

The Bayes classifier (6)



$$\begin{aligned} P(Y=\text{male} \mid X=150 \text{ cm}) &= 0.2 \\ P(Y=\text{female} \mid X=150 \text{ cm}) &= 0.8 \end{aligned} \quad \Rightarrow f^*(150) = \text{female}$$

- There are some values of X where the class label can be predicted with reasonably low error rate (say, 150 cm or 190 cm). There are the values for which the regression function is close to 0 or close to 1.

The Bayes classifier (7)

- ▶ There are some values of X where the class label cannot reasonably be predicted, say for 170 cm. These are the values for which the regression function is close to 0.5.

Naive approaches

Plugin-approach. Simply estimate the regression function $\eta(x)$ by some quantity $\eta_n(x)$ and build the plugin-classifier

$$f_n := \begin{cases} 1 & \text{if } \eta_n(x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Theoretical considerations:

- ▶ It can be shown that the plugin-approach is universally consistent. ☺

Practical considerations:

- ▶ Estimating densities is notoriously hard, in particular for high-dimensional input spaces. ☹

Naive approaches (2)

Density estimation approach. Even estimate the joint probability distribution $P(X, Y)$ (and thus in particular the class conditional densities $P(X|Y)$ and the regression function).

Even harder than the plugin-approach.

Loss functions for classification

- ▶ The most important loss function for classification is the 0-1-loss. This is what we are really interested in.
- ▶ However, it can be hard to minimize zero-one-losses (\leadsto discrete optimization problem, NP hard).
- ▶ For this reason, all successful machine learning algorithms do the following two things:
 - ▶ Instead of a binary output they return real valued outputs, and use the sign of the output as the final label.
 - ▶ They consider a convex relaxation, a “surrogate loss function”, of the zero-one-loss function.

Loss functions for classification (2)

The most important convex surrogate loss functions are:

- ▶ **Hinge loss (soft margin loss)**, used in support vector machines:

$$\ell(x, f(x), y) = \max\{0, 1 - y \cdot f(x)\}$$

- ▶ **exponential loss**, used in boosting: $\exp(-y \cdot f(x))$
- ▶ **logistic loss**, used in logistic regression: $\log(1 + \exp(-y \cdot f(x)))$

Remarks:

- ▶ Surrogate loss functions lead to optimization problems that are computationally tractable. But do they also give the correct result? Does it hurt to replace the 0-1-loss by a surrogate?

This problem has been solved in *Bartlett, Jordan, McAuliffe: Convexity, classification and risk bounds. Journal of the American Statistical Association, 2006.*

Loss functions for classification (3)

- ▶ Similarly as for the 0-1-loss, we can compute the Bayes classifiers for some of the other loss functions explicitly in terms of the regression function.

If time permits, we revisit these issues later in the course.

Particular setup for regression

Loss functions for regression

While in classification, there is a “natural loss function” (the 0-1-loss), there exist many loss functions for regression and it is not so obvious which one is the most useful one:

- ▶ Squared loss (L_2 -loss): $\ell(x, f(x), y) = (f(x) - y)^2$
- ▶ L_1 -loss function: $\ell(x, f(x), y) = |f(x) - y|$
- ▶ ε -insensitive loss function:

$$\ell(x, f(x), y) = \begin{cases} |f(x) - y| - \varepsilon & \text{if } |f(x) - y| \geq \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

- ▶ Huber's robust loss function:

$$\ell(x, f(x), y) = \begin{cases} \frac{1}{2\varepsilon}(f(x) - y)^2 & \text{if } |f(x) - y| \geq \varepsilon \\ |f(x) - y| - \varepsilon/2 & \text{otherwise} \end{cases}$$

Regression function again

As in the classification setting, we define the regression function:

$$\eta(x) = E(Y \mid X = x)$$

Theorem 4 (Decomposition)

We always have

$$E(|f(X) - Y|^2) = E(|f(X) - \eta(X)|^2) + E(|\eta(X) - Y|^2).$$

Note: Getting a related inequality with \leq is trivial (by the triangle inequality), but the equality in this statement is not obvious.

Regression function again (2)

Proof. (see GKKW p.2)

$$\begin{aligned}
 & E(|f(x) - \gamma|^2) \\
 &= E\left(\underbrace{|f(x) - \eta(x)|}_{=} + \underbrace{\eta(x) - \gamma}_{=}^2\right) \\
 &= E\left((f(x) - \eta(x))^2\right) + E\left((\eta(x) - \gamma)^2\right) + 2E\left((f(x) - \eta(x))(\eta(x) - \gamma)\right) \\
 &= E\left((f(x) - \eta(x))^2\right) + E\left((\eta(x) - \gamma)^2\right) \quad \textcircled{\#} = 0
 \end{aligned}$$

Regression function again (3)

$$\begin{aligned}
 \# &= E((f(x) - \eta(x))(\eta(x) - \gamma)) = \textcircled{1}: E(z) = E(E(z|G)) \\
 &= E\left(E((f(x) - \eta(x))(\eta(x) - \gamma)|X)\right) = \textcircled{2} E(s(x)g(\gamma, x)|x) = g(x) \cdot E(g(x)|G) \\
 &= E((f(x) - \eta(x))) \cdot \underbrace{E(\eta(x) - \gamma | X)}_{= 0} = \\
 &\quad = \eta(x) - \underbrace{E(\gamma | X)}_{=\eta(x)} = 0
 \end{aligned}$$



Theorem 5 (Optimal L_2 -regression)

The regression function minimizes the L_2 -risk.

Regression function again (4)

Proof. Follows directly from Proposition 4:

- ▶ Second expectation on the rhs does not depend on f .
- ▶ First expectation is always ≥ 0 , and it is $= 0$ for $f(X) = \eta(X)$.
- ▶ So the whole rhs is minimized by $f(X) = \eta(X)$.



Bias-Variance tradeoff

Let f_n be the function constructed by an algorithm on n points, and $f : \mathbb{R}^d \rightarrow \mathbb{R}$ the true function (regression function). Then we can decompose the pointwise expected L_2 error in two terms:

$$E(|f_n(x) - f(x)|^2) = \underbrace{E\left((f_n(x) - E(f_n(x)))^2\right)}_{\text{Variance term}} + \underbrace{\left(E(f_n(x)) - f(x)\right)^2}_{\text{Bias term}}$$

Note: we always have \leq (for any loss function), but for the L_2 -error we get equality.

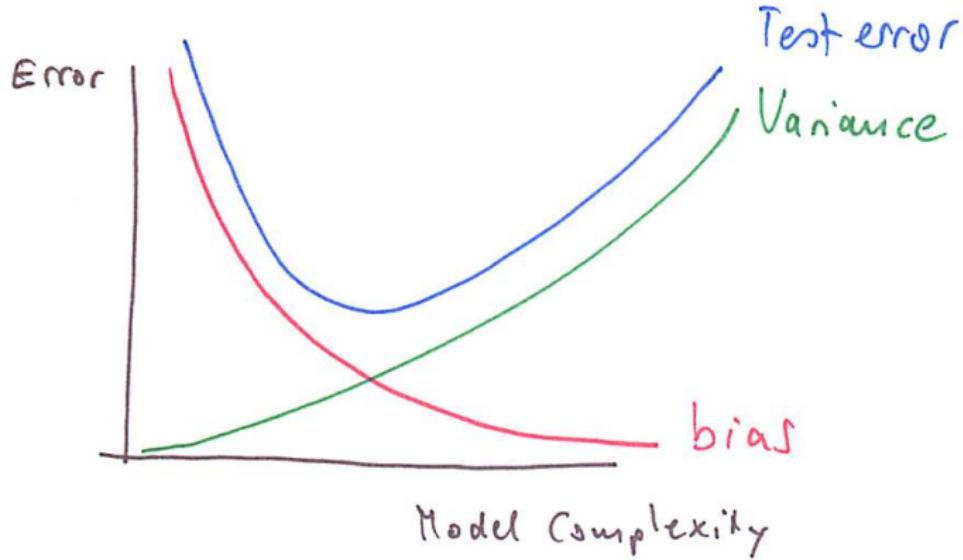
Proof. Exactly the same as the one of Proposition 4. ☺

Intuition:

Bias-Variance tradeoff (2)

- ▶ The variance term measures how much the regression function f_n varies with the noise in the data.
It has the same connotation as the estimation error we discussed earlier.
- ▶ The bias term measures how good the approximation is in case we don't have noise.
It has the same connotation as the approximation error we discussed earlier.

Bias-Variance tradeoff (3)



Linear Methods for regression

Linear least squares regression

Literature:

- ▶ Hastie/Tibshirani/Friedman Section 3
- ▶ Bishop Sec 3

Starting point

- ▶ We want to solve a regression problem with respect to the least squares loss (also called L_2 -loss).
- ▶ Have seen: if we optimized over the space of all (measurable) functions and had infinite amount of data, the solution would be given by the regression function $\eta(x) = E(Y | X = x)$.
- ▶ But now: only have a finite sample, and we don't want to optimize over all functions (overfitting).

Linear setup

- ▶ Assume we have training data (X_i, Y_i) with $X_i \in \mathcal{X} := \mathbb{R}^d$ and $Y_i \in \mathcal{Y} := \mathbb{R}$.
- ▶ We want to find the “best” *linear function*, that is a function of the form

$$f(x) = \sum_{i=1}^d w_i x^{(i)} + b$$

where $x = (x^{(1)}, \dots, x^{(d)})^t \in \mathbb{R}^d$.

The w_i are called “weights” and b the “offset” or “intercept” or “threshold”.

Linear setup (2)

- To write everything in a more concise form, we stack the training inputs into a big matrix (each point is one row) and the output in a big vector:

$$X = \left(\begin{array}{ccc} x_{11} & \dots & x_{1d} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{nd} \end{array} \right) \Big\}^n \quad d \quad y = \left(\begin{array}{c} y_1 \\ \vdots \\ y_n \end{array} \right) \Big\}^n \quad w = \left(\begin{array}{c} w_1 \\ \vdots \\ w_d \end{array} \right) \Big\}^d$$

- Now we can write much more concisely:

$$f(X_i) = \langle w, X_i \rangle + b$$

Linear setup (3)

- ▶ Formally, the linear least squares problem is the following:
(##) Determine $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ as to minimize the empirical least squares error

$$\frac{1}{n} \sum_{i=1}^n \left(Y_i - (\langle w, X_i \rangle + b) \right)^2$$

Getting rid of b

We want to write the problem even more concisely.

- ▶ Define the matrices

$$\tilde{X} = \begin{pmatrix} X_{11} & \dots & X_{1d} & 1 \\ \vdots & & \vdots & \vdots \\ X_{n1} & \dots & X_{nd} & 1 \end{pmatrix}^{\text{d+1}}_n \quad \tilde{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_d \\ b \end{pmatrix}^{\text{d+1}}$$

- ▶ Then we have $\langle \tilde{w}, \tilde{X}_i \rangle = \sum_{k=1}^{d+1} \tilde{w}_k \tilde{X}_{ik} = \sum_{k=1}^d w_k X_{ik} + b$
- ▶ Hence, there is a unique correspondence between the original problem to the following new problem: find a vector $\tilde{w} \in \mathbb{R}^{d+1}$ such that $\frac{1}{n} \sum_{i=1}^n (\tilde{Y}_i - \langle \tilde{w}, \tilde{X}_i \rangle)^2$ is minimized.

Getting rid of b (2)

- So without loss of generality we consider the following simplified problem: (#) determine $w \in \mathbb{R}^d$ as to minimize the empirical least squares error

$$\frac{1}{n} \sum_{i=1}^n \left(Y_i - \langle w, X_i \rangle \right)^2 = \frac{1}{n} \| Y - Xw \|^2$$

ML \leadsto Optimization problem

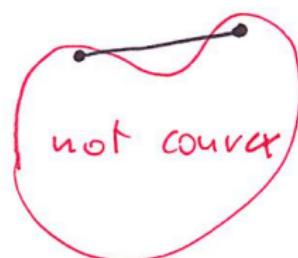
We can see:

- ▶ In order to solve $(\# \#)$, we need to solve an optimization problem
- ▶ In this particular case, we will see in a minute that we can solve it analytically.
- ▶ For most other ML algorithms, we need to use optimization algorithms to achieve this.

Excursion: convex optimization problems

Convex sets:

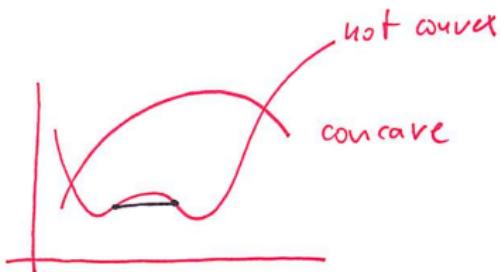
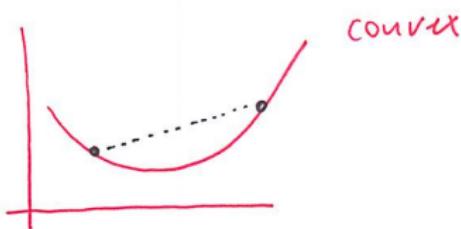
- ▶ A subset S of a vector space is called convex if for all $x, y \in S$ and for all $t \in [0, 1]$ it holds that $tx + (1 - t)y \in S$.
- ▶ In words: for any two points $x, y \in S$, the straight line connecting these two points is contained in the set S .



Excursion: convex optimization problems (2)

Convex functions:

- ▶ A function $f : S \rightarrow \mathbb{R}$ that is defined on a convex domain S is called convex if for all $x, y \in S$ and $t \in [0, 1]$ we have
$$f(tx + (1 - t)y) \leq tf(x) + (1 - t)f(y).$$
- ▶ Intuitively, this means that if we look at the graph of the function and we connect two points of this graph by a straight line, then this line is always above the graph.



Excursion: convex optimization problems (3)

Convex optimization problem:

- ▶ An optimization problem of the form

$$\text{minimize } f(x)$$

$$\text{subject to } g_i(x) \leq 0 \quad (i = 1, \dots, k)$$

is called convex if the functions f and g_i ($i = 1, \dots, k$) are all convex.

- ▶ Convex optimization problems have the desirable property that any local minimum is already a global minimum.

Least squares regression is convex

Theorem 6 (Least squares is convex)

The least squares optimization problem (##) is a convex optimization problem.

Proof. Exercise

Solution, part 1

Theorem 7 (Solution, case $\text{rank}(X) = d$)

Assume that X has rank d . Then the solution w of linear least squares regression ($\#\#$) is given by $w = (X^t X)^{-1} X^t Y$.

Proof intuition.

- ▶ We need to take the derivative and set it to 0.
- ▶ Then we have to check that what we get is indeed a minimum (in 1d we would look at the second derivative for this).
- ▶ The minimum then has to be a global minimum because the objective function is convex.

We can either do all this by foot, coordinate-wise. Or we do it more elegantly as follows:

Solution, part 1 (2)

Proof, formally. We write all equations in matrix form:

- ▶ Objective function: $Obj(w) := \frac{1}{2} \|Y - Xw\|^2$
- ▶ Derivative: $\frac{\partial Obj(w)}{\partial w} = -X^t(Y - Xw).$
- ▶ Setting this to zero gives the necessary condition $X^t Y = (X^t X)w.$
- ▶ We always have $rank(X) = rank(X^t X) = rank(XX^t)$. In particular, the matrix $X^t X$ is a $d \times d$ matrix of rank d , hence invertible.
- ▶ So we can solve for $w = (X^t X)^{-1} X^t Y.$
- ▶ Now we need to figure out whether this is indeed a minimum. To this end, consider the Hessian matrix that contains all second derivatives: $H(Obj) = \frac{\partial^2 Obj}{\partial w \partial w'} = X^t X.$

Solution, part 1 (3)

- ▶ This matrix is positive definite (obviously all eigenvalues ≥ 0 , and because of the rank condition we have > 0). So the solution we computed above is indeed a local minimum.



Generalized inverse of a matrix

In order to treat the case where X does not have rank d , we need to define the following:

Consider a symmetric matrix $A \in \mathbb{R}^{d \times d}$.

- ▶ Let $\lambda_1, \dots, \lambda_d$ the eigenvalues and v_1, \dots, v_d a corresponding set of eigenvectors of A . We can write A in the spectral decomposition as

$$A = \sum_{i=1}^d \lambda_i v_i v_i^t$$

Generalized inverse of a matrix (2)

- In case the matrix has rank d , all its eigenvalues are non-zero. Then we can write the inverse of A as

$$A^{-1} = \sum_{i=1}^d \frac{1}{\lambda_i} v_i v_i^t$$

- In case the matrix is not of full rank, it is not invertible. However, we can define the **Moore-Penrose generalized inverse** as

$$A^+ := \sum_{i:\lambda_i \neq 0} \frac{1}{\lambda_i} v_i v_i^t$$

(intuitively, this is the inverse of the matrix A restricted to the subspace orthogonal to its nullspace).

Generalized inverse of a matrix (3)

Properties of the generalized inverse:

In general, $AA^+ \neq Id$ and $A^+A \neq Id$.

But we have the following slightly weaker properties:

- ▶ $AA^+A = A$ and $A^+AA^+ = A^+$
- ▶ $(A^+)^+ = A$
- ▶ A^+A and AA^+ are both symmetric.
- ▶ If A is invertible, then $A^{-1} = A^+$.
- ▶ AA^+ is an orthogonal projection on the $\text{ran}(A)$ (the image of the matrix A), and A^+A is an orthogonal projection on $\text{ran}(A^t)$.

Solution, part 2

Theorem 8 (Solution, case $\text{rank}(X) < d$)

Assume that X has $\text{rank} < d$. Then a solution w of linear least squares regression ($\#\#$) is given by $w = (X^t X)^+ X^t Y$. This solution is not unique. If w_1, w_2 are two solutions, then their predictions agree on the training data, that is $\langle w_1, X_i \rangle = \langle w_2, X_i \rangle$ for all $i = 1, \dots, n$.

Proof (sketch).

- ▶ As above we get the necessary condition $X^t Y = (X^t X)w$.
- ▶ One can check that one particular vector w that satisfies this condition is given as $w = (X^t X)^+ X^t Y$ (EXERCISE!)
- ▶ However, w is not unique: Let w be a solution and v any vector with $Xv = 0$ (exists because X has $\text{rank} < d$). Then $w + v$ is a solution as well.

Implementation in Matlab

- ▶ Never compute the solution using the `inv` function!
- ▶ Instead, solve the linear system $X^t Y = (X^t X)w$ for w .
- ▶ In Matlab you can solve the linear system $Ax = b$ by the backslash operator: $A \backslash b$.

Reason: computing the inverse of a matrix is numerically less stable and much more expensive (essentially, computing the inverse of a matrix is as expensive as solving n linear systems $Ax = e_i$).

Summary: Linear least squares regression

- ▶ Regression problem, $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$
- ▶ Loss function: L_2 -loss
- ▶ Function class \mathcal{F} : set of all linear functions over \mathcal{X} (this space is “pretty small”).
- ▶ No regularization.
- ▶ Finding the linear function that minimizes the empirical L_2 -loss is a convex optimization problem, and we can compute its solution analytically.

Least squares with linear combination of basis functions

Literature:

- ▶ Hastie/Tibshirani/Friedman Section 3
- ▶ Bishop Section 3

Using non-linear basis functions

Idea:

- ▶ Linear functions are often quite restrictive.
- ▶ Instead, want to learn a function of the form

$$f(x) = \sum_{i=1}^D w_i \Phi_i(x)$$

where the functions Φ_1, \dots, Φ_D are arbitrary “basis functions”.

- ▶ Note: f is linear in the parameters w , but if the functions Φ are non-linear, then so is f .
- ▶ Example:
 - ▶ If we want to learn a periodic function, the Φ_i might be the first couple of Fourier functions.

How to solve it

It is easy to rewrite the “standard” least squares problem in this more general framework:

- ▶ Define the design matrix as follows:

$$\Phi = \begin{pmatrix} \phi_1(x_1) & \dots & \phi_d(x_1) \\ \vdots & & \vdots \\ \phi_1(x_n) & \dots & \phi_d(x_n) \end{pmatrix}$$

- ▶ Then the least squares problem is to find w as to minimize $\|Y - \Phi w\|^2$.
- ▶ This has the solution $w = (\Phi^t \Phi)^{-1} \Phi^t Y$ (with exact inverse or generalized inverse) as we have seen above.

Advantages and disadvantages

- ▶ If we have prior knowledge about our data, we can select a “good” set of basis functions.
- ▶ However, we still just work in a D -dimensional function space. This is still quite restrictive.
Example: Assume our original data is in \mathbb{R}^d , and we want to have a function space with polynomials of degrees one and two. There are already $d(d - 1)/2 = \Theta(d)$ polynomials of degree two ...
- ▶ For any useful inference, the dimension D of the space has to be much smaller than the number n of data points.

Summary: Linear least squares regression

- ▶ Regression problem, \mathcal{X} arbitrary space, $\mathcal{Y} = \mathbb{R}$
- ▶ Loss function: L_2 -loss
- ▶ Function class \mathcal{F} : a linear combination of a fixed set of basis functions.
- ▶ No regularization.
- ▶ Finding the linear function that minimizes the empirical L_2 -loss is a convex optimization problem, and we can compute its solution analytically.

Ridge regression: least squares with L_2 -regularization

Literature: Hastie/Tibshirani/Friedman Section 3.4.3; Bishop Section 3

Idea

Want to improve standard L_2 -regression. Two points of view:

1. Want to have a unique solution, no matter what the rank of the design matrix is. This is going to improve numerical stability.
2. In the standard problem, the coefficients w_i can become very large. This leads to a very large variance of the results. To avoid this effect, we want to introduce regularization to force the coefficients to stay “small”.

Ridge regression problem

Consider the following regularization problem:

- ▶ Input space \mathcal{X} arbitrary, output space $\mathcal{Y} = \mathbb{R}$.
- ▶ Fix a set of basis functions $\Phi_1, \dots, \Phi_D : \mathcal{X} \rightarrow \mathbb{R}$
- ▶ As function space choose all functions of the form $f(x) = \sum_i w_i \Phi_i(x)$.
- ▶ As regularizer use $\Omega(f) := \|w\|^2 = \sum_{i=1}^D w_i^2$. Choose a regularization constant $\lambda > 0$.
- ▶ Then solve the problem

$$w_{n,\lambda} := \operatorname{argmin}_{w \in \mathbb{R}^D} \frac{1}{n} \sum_{i=1}^n \left(Y_i - \langle w, \Phi_i(x) \rangle \right)^2 + \lambda \cdot \sum_{i=1}^D w_i^2$$

Ridge regression problem (2)

In matrix notation, this problem has the form

$$w_{n,\lambda} := \underset{w \in \mathbb{R}^D}{\operatorname{argmin}} \frac{1}{n} \|Y - \Phi w\|^2 + \lambda \|w\|^2.$$

Solution

Theorem 9 (Solution of Ridge Regression)

The coefficients $w_{n,\lambda}$ that solve the ridge regression problem are given as

$$w_{n,\lambda} := \left(\Phi^t \Phi + n\lambda Id_D \right)^{-1} \Phi^t Y$$

where Id_D is the $D \times D$ identity matrix.

Solution (2)

Proof.

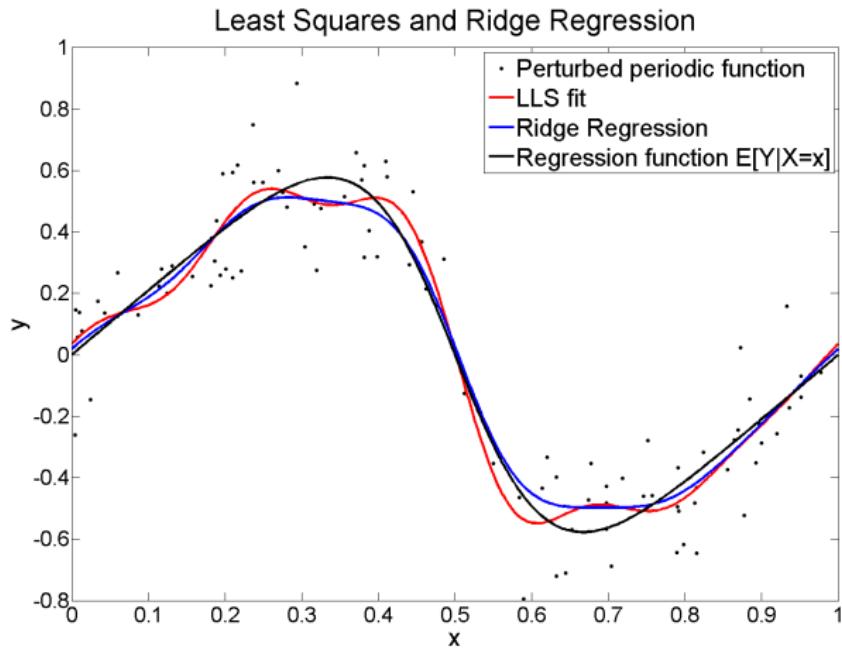
- ▶ Objective function is $Obj(w) := \frac{1}{n} \|Y - \Phi w\|^2 + \lambda \|w\|^2$.
- ▶ Note that this function is convex.
- ▶ Take the derivative with respect to w and set it to 0:

$$\begin{aligned}\frac{\partial Obj}{\partial w} Obj(w) &= -\frac{2}{n} \Phi^t (Y - \Phi w) + 2\lambda w \stackrel{!}{=} 0 \\ \implies (\Phi^t \Phi + n\lambda I_D) w_{n,\lambda} &= \Phi^t Y\end{aligned}$$

- ▶ Note that the matrix $(\Phi^t \Phi + n\lambda \mathbb{1}_D)$ has full rank whenever $\lambda > 0$. So we can take the inverse, and the theorem follows as in the standard L_2 -regression case. ☺

Example (by Matthias Hein)

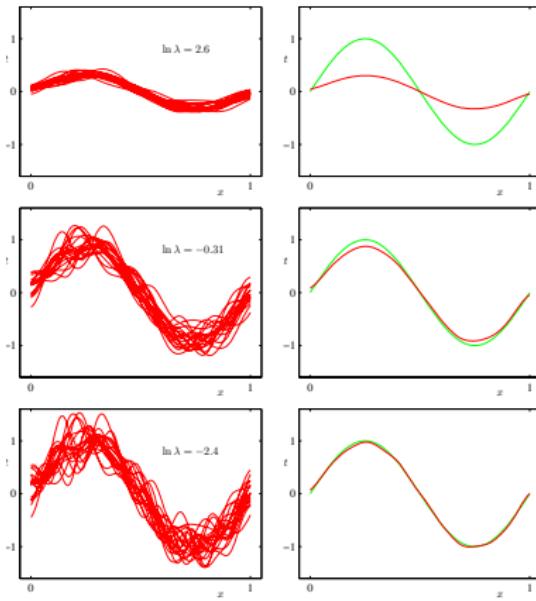
- ▶ True function: periodic function + noise
- ▶ Basis functions Φ : first 10 Fourier basis functions



Example (from Bishop's book)

Left: results for decreasing amount of regularization

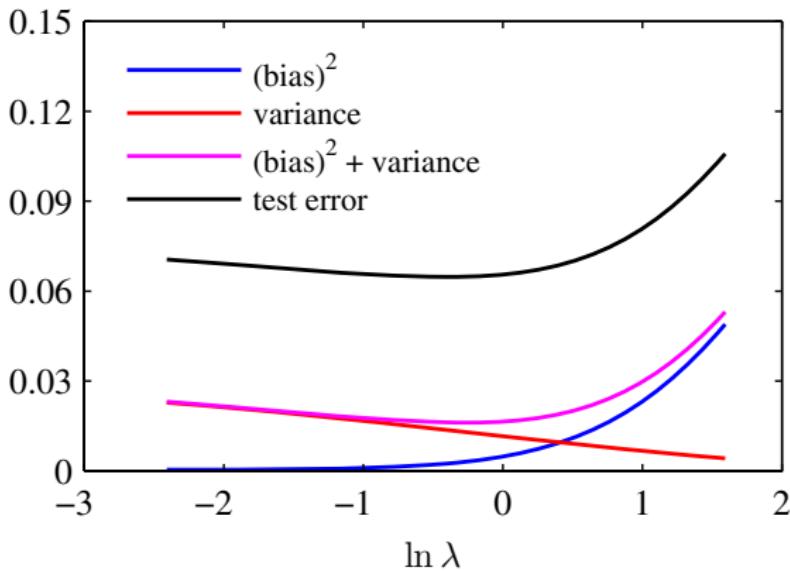
Right: True curve (green), average estimated curve (red)



Example (from Bishop's book) (2)

Same example, bias-variance decomposition:

Larger regularization constant λ leads to less complex functions:



Normalization

For standard linear least squares regression: if we shift or rescale the individual coordinates of our input data, this directly translates to a shift and rescaling of the parameters w and b . However, this is not true for regularized regression.

- ▶ Fix a constant α and replace all basis functions Φ_i by $\Psi_i := \alpha \cdot \Phi_i$. Then we get a different solution:
Replacing Φ_i in the formula by Ψ_i/α gives

$$w_{n,\lambda} = \alpha \left(\Psi^t \Psi + \alpha^2 n \lambda \mathbb{1}_D \right)^{-1} \Psi^t Y$$

- ▶ Similarly, if we replace each basis function Φ_i by $\Phi_i + c$ for some $c \in \mathbb{R}$, we get a very different solution (EXERCISE).

Normalization (2)

The point is that the rescaling and shifting also influences the regularization term in a non-linear way.

In order to make sure that all basis functions “are treated the same” it is thus recommended to **standardize** your basis functions:

1. Centering:

Replace Φ_i by $\Phi_i^{centered} := \Phi_i - \bar{\Phi}_i$ with $\bar{\Phi}_i := \frac{1}{n} \sum_{j=1}^n \Phi_i(X_j)$.

2. Normalizing the variance: rescale each basis function such that it has unit L_2 -norm (variance) on the training data:

$$\Phi_i^{rescaled} := \frac{\Phi_i^{centered}}{(\sum_{j=1}^n \Phi_i^{centered}(X_j))^{1/2}}$$

Normalization (3)

In case you work with a matrix X directly, you would center and normalize the columns of the matrix X to have center 0 and unit norm.

Geometric interpretation via SVD

- ▶ Consider the Singular Value Decomposition of the matrix $\Phi \in \mathbb{R}^{n \times d}$:

$$\Phi = U\Sigma V^t$$

- ▶ Plugging this into the formula for $w_{n,\lambda}$ leads to

$$w_{n,\lambda} = \dots = V \operatorname{diag} \left(\frac{\sigma_j}{\sigma_j^2 + \lambda} \right) U^t Y$$

- ▶ Standard least squares regression (without regularization) corresponds to $\lambda = 0$, and the fraction satisfies

$$\frac{\sigma_j}{\sigma_j^2 + \lambda} = \frac{1}{\sigma_j}$$

Geometric interpretation via SVD (2)

- ▶ Regularized case:
 - ▶ Case σ_i large: not much difference to non-regularized case:

$$\frac{\sigma_j}{\sigma_j^2 + \lambda} \approx \frac{1}{\sigma_j}$$

- ▶ Case σ_i small: here it makes a lot of difference whether we have σ_i^2 or $\sigma_i^2 + \lambda$ in the denominator. In particular,

$$\frac{\sigma_j}{\sigma_j^2 + \lambda} \ll \frac{1}{\sigma_j}$$

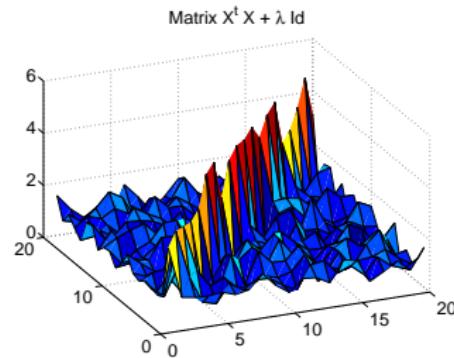
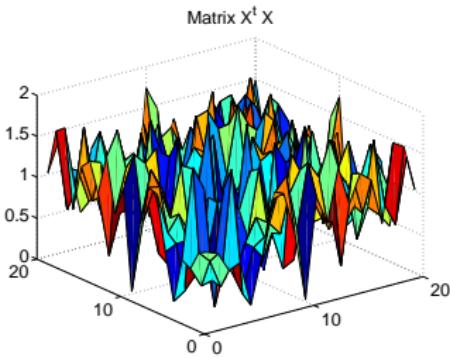
This means that the regularization “shrinks” the directions of small variance. Intuitively, these are the directions that mainly contain noise, no signal.

History and Terminology

- ▶ Invented by Andrey Tikhonov, 1943, in the context of integral equations.
Original publication: *Tikhonov, Andrey Nikolayevich. On the stability of inverse problems. Doklady Akademii Nauk SSSR, 1943*
For this reason, it is also called **Tikhonov regularization**.
- ▶ Introduced in statistics literature in the following paper:
Hoerl and Kennard. Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 1970.

History and Terminology (2)

- ▶ Originally, the intention was to make the solution of the least squares problem more stable and to achieve a unique solution.
 - ▶ Replace the matrix $\Phi^t \Phi$ in the least squares solution by the matrix $\Phi^t \Phi + \lambda I d$.
 - ▶ This is where the name “ridge” comes from (we add a little “ridge” on the diagonal of the matrix).



- ▶ Note that the matrix $\Phi^t \Phi + n\lambda I d$ is always of full rank, hence it can be inverted.

History and Terminology (3)

- ▶ The regularization interpretation we described above (adding the penalty term avoids overfitting) is more recent.

History and Terminology (4)

In statistics, related methods are often called “shrinkage methods” (because we try to “shrink” the weights w_i).

From a statistics point of view, they can be justified by what is called “Stein’s paradox” (discovered in the 1950ies). Essentially, this paradox says that if we want to estimate at least three parameters jointly, then it is better to “shrink them”. Here is a simple example:

- ▶ Assume you want to estimate the mean of a normal distribution $N(\Theta, Id)$ in \mathbb{R}^d , $d \geq 3$.
- ▶ Assume we have just a single data point $X \in \mathbb{R}^d$ from this distribution.
- ▶ Standard least squares estimator: $\hat{\Theta}_{LS} = X$.

History and Terminology (5)

- ▶ Now consider the following “shrinkage estimator” (it is called the James-Stein estimator): $\hat{\Theta}_{JS} = \left(1 - \frac{d-2}{\|X\|^2}\right)X$.
- ▶ One can prove that it always outperforms the standard least squares error:

$$E(\|\Theta - \hat{\Theta}_{LS}\|) \geq E(\|\Theta - \hat{\Theta}_{JS}\|)$$

Read it on wikipedia if you are interested ☺

Summary

- ▶ Regression problem, \mathcal{X} arbitrary space, $\mathcal{Y} = \mathbb{R}$
- ▶ Loss function: L_2 -loss
- ▶ Function class \mathcal{F} : a linear combination of a fixed set of basis functions.
- ▶ Regularizer: $\Omega(f) = \|w\|^2$
- ▶ Finding the function that minimizes the regularized risk is a convex optimization problem, and we can compute its solution analytically.

Lasso: least squares with L_1 -regularization

Literature: Hastie/Tibshirani/Friedman, Section 3.4.3; Bishop Section 3

Original paper: *Tibshirani: Regression shrinkage and selection via the lasso. J. Royal. Statist. Soc. B, 1996*

Sparsity

- ▶ Consider the setting of linear regression with basis functions Φ_1, \dots, Φ_D .
- ▶ It is very desirable to obtain a solution function $f_n := \sum_i w_i \Phi_i$ for which many of the coefficients w_i are zero. Such a solution is called “sparse”.
- ▶ Reasons:
 - ▶ Shrinkage in general leads to smaller errors (but this can also be true for non-sparse solutions).
 - ▶ Computational reasons: even if we have many basis functions, we just need to evaluate few of them.
 - ▶ Interpretability of the solution

Sparsity and the L_1 -norm

Naive way to enforce sparsity:

- ▶ Use the regularizer

$$\Omega_0(f) := \sum_{i=1}^D \mathbb{1}_{w_i \neq 0}.$$

It directly penalizes the number of non-zero entries w_i .

- ▶ $\|w\|_0$ is a discrete function, so it is not a good idea to try to minimize it...

Sparsity and the L_1 -norm (2)

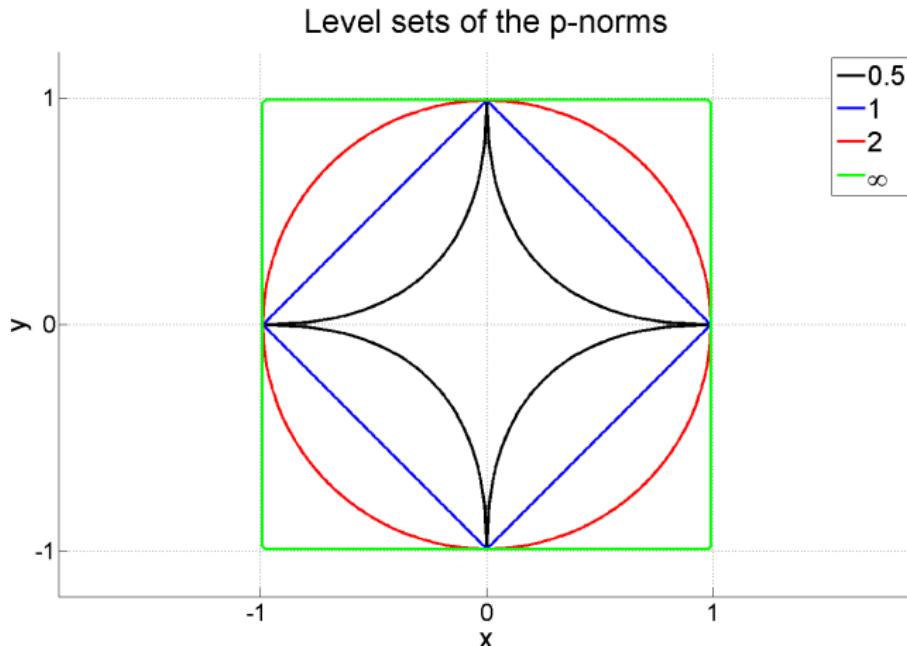
Alternatively: Observe that we can approximate the naive regularizer $\Omega_0(f)$ as follows:

- ▶ Define the p -norm of a vector $w \in \mathbb{R}^D$ as

$$\|w\|_p := \left(\sum_{i=1}^D |w_i|^p \right)^{1/p}.$$

For any $p \geq 1$, this is a norm and as such a convex function.

Sparsity and the L_1 -norm (3)



(Image by Matthias Hein)

Sparsity and the L_1 -norm (4)

- ▶ It is easy to see that $\Omega_0(f) = \lim_{p \rightarrow 0} \|w\|_p^p$
- ▶ Among all proper norms ($p \geq 1$), the L_1 -norm is “closest” to $\Omega_0(f)$.
- ▶ Hence, it might be a good idea to regularize by the L_1 -norm.

Sparsity and the L_1 -norm (5)

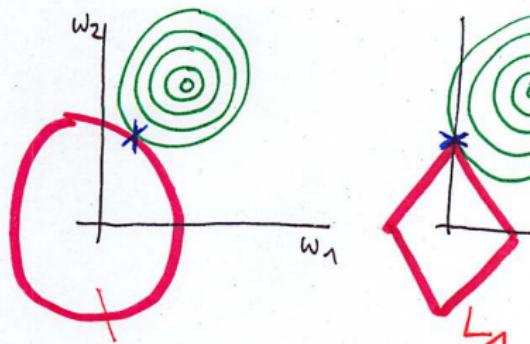
Another intuitive argument why solutions with L_1 -regularization might be sparser than L_2 -regularization:

- ▶ The L_2 -norm puts a particularly large penalty on large coefficients w_i . That is, to avoid a large L_2 -penalty, it is better to have many small w_i that are all non-zero than to have most $w_i = 0$ and a couple of large w_i .
- ▶ The L_1 -norm at least does not have this “preference” for many small weights. It punishes all weights linearly, not quadratic, and thus can afford to have a large weight if at the same time many small weights disappear...

Sparsity and the L_1 -norm (6)

Illustration: Assume we restrict the search to functions with $\|w\| \leq \text{const.}$ The blue cross shows the best solution $w = (w_1, w_2)^t$. It is not sparse for L_2 , but sparse for L_1 -norm regularization.

contour lines of the empirical error

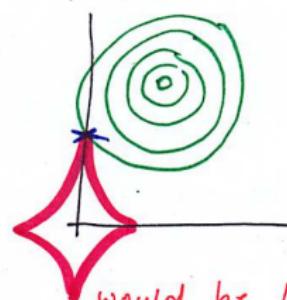


L_2 regularization:
set with $\|w\|_2 \leq \text{const}$

$w_1, w_2 \neq 0$, not sparse



$w_1 = 0$, sparse



would be L_p for
 $p < 1$

even sparser, but not
convex any more.

The Lasso

Consider the following regularization problem:

- ▶ Input space \mathcal{X} arbitrary, output space $\mathcal{Y} = \mathbb{R}$.
- ▶ Fix a set of basis functions $\Phi_1, \dots, \Phi_D : \mathcal{X} \rightarrow \mathbb{R}$
- ▶ As function space choose all functions of the form $f(x) = \sum_i w_i \Phi_i(x)$.
- ▶ As regularizer use $\Omega(f) := \|w\|_1 = \sum_{i=1}^D |w_i|$. Choose a regularization constant $\lambda > 0$.
- ▶ Then solve the problem

$$w_{n,\lambda} := \operatorname{argmin}_{w \in \mathbb{R}^D} \frac{1}{n} \sum_{i=1}^n \left(Y_i - \langle w, \Phi_i(x) \rangle \right)^2 + \lambda \cdot \sum_{i=1}^D |w_i|$$

The Lasso (2)

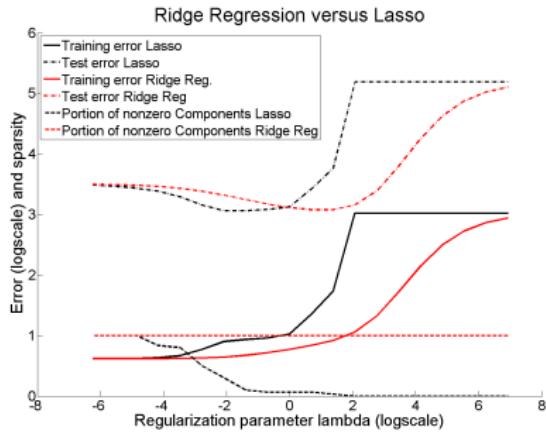
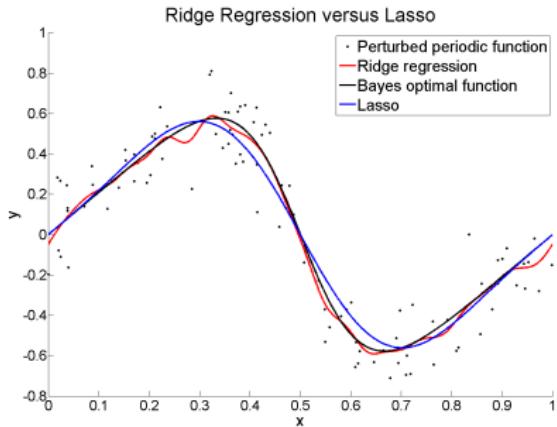
In matrix notation, this problem has the form

$$w_{n,\lambda} := \underset{w \in \mathbb{R}^D}{\operatorname{argmin}} \frac{1}{n} \|Y - \Phi w\|^2 + \lambda \|w\|_1.$$

Solution of the Lasso problem

- ▶ The Lasso objective function is convex (it is a sum of two convex functions).
- ▶ However, there does not exist a closed form solution.
- ▶ Hence it has to be solved by a standard algorithm for convex optimization.
 - ▶ In general, any convex solver can be used, but might be slow.
 - ▶ Observing that the problem can be recast as a quadratic problem might help already.
 - ▶ But many faster approaches exist, for example coordinate descent algorithms. We are not going to discuss them in the lecture.

Example



(Figure by Matthias Hein)

History

- ▶ The name LASSO stands for “least absolute shrinkage and selection operator”
- ▶ First invented by *Tibshirani: Regression shrinkage and selection via the lasso. J. Royal. Statist. Soc. B, 1996*
- ▶ For a short retrospective and some important literature pointers, see *Tibshirani: Regression shrinkage and selection via the lasso: a retrospective. J. R. Statist. Soc. B (2011)*

Summary: the Lasso

- ▶ Regression problem, \mathcal{X} arbitrary space, $\mathcal{Y} = \mathbb{R}$
- ▶ Loss function: L_2 -loss
- ▶ Function class \mathcal{F} : a linear combination of a fixed set of basis functions.
- ▶ Regularizer: $\Omega(f) = \|w\|_1$,
Enforces sparsity.
- ▶ Convex optimization problem, no analytic solution, but efficient solvers exist.

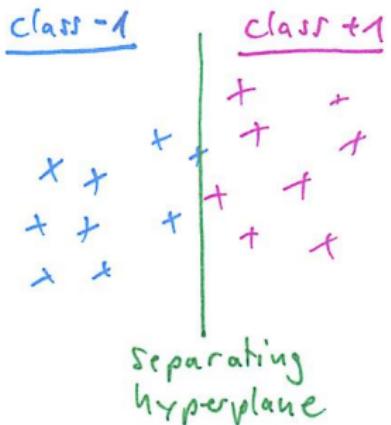
Linear Methods for classification

Intuition and feature representation

Intuition

Given:

- ▶ Want to solve a classification problem with input space $\mathcal{X} = \mathbb{R}^d$ and output space $\mathcal{Y} = \{\pm 1\}$ (for simplicity we focus on the two-class case for now).
- ▶ Idea is to separate the two classes by a linear function:



Feature representation of data

On the first glance, the assumption that the data points are in \mathbb{R}^d looks pretty restrictive. What if our data is not “numbers”?

It turns out that in many cases it is a good idea to represent “objects” by “feature vectors”.

Feature representation of data (2)

Example: bag of words representation for texts

- ▶ Make a list of all words occurring in the text
- ▶ Throw away all words that are too common ("the", "a", "for", "you", ...)
- ▶ Use "stemming" to throw away word endings (like the plural "s"): we want to consider the word "horse" the same as "horses")
- ▶ For each text, count how often each word occurs
- ▶ Represent each text as a vector: each dimension corresponds to one word, and the entry of the vector is how often this word occurs in the given text.

Feature representation of data (3)

T_1 : I like to play soccer.

T_2 : My soccer shoes are red.

T_3 : We moved to a new house.

	T_1	T_2	T_3
like	1	0	0
play	1	0	0
soccer	1	1	0
shoe	0	1	0
red	0	1	0
move	0	0	1
house	0	0	1

Feature representation of data (4)

Example: strings in a feature representation

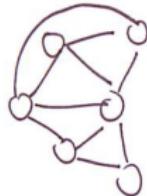
- ▶ Given a string
- ▶ Represent it by counting substrings (can also allow substrings with “gaps” in between)

Feature representation of data (5)

Example: motif representation of graphs (such as chemical molecules)

Count the occurrence of certain subgraphs (called motifs):

graph 1:



Motifs: # in graph 1 # in graph 2



10

in graph 2

7



5

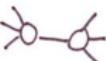
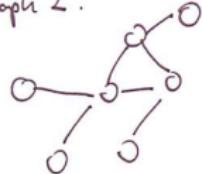
1



1

1

graph 2:



0

0

:

Feature representation of data (6)

Example: books and/or users in amazon.

- ▶ can describe a book by how often it was bought by each user
- ▶ or by how often it was bought together with each other book.

Representation of books by user data:

	user1	user2	user3
book 1	0	0	1
book 2	1	0	0
book 3	0	0	0
:	1	1	0
	0	0	0

how often did user1 buy book 4

Representation of books by books

	book 1	book 2	book 3	...
book 1	?	1	3	
book 2	1	5	2	
book 3	3	2	5	
:				

how often was book 3 bought together with book 2

Feature representation of data (7)

Example: images

- ▶ Can obviously represent images as vectors of greyscale values, or RGB values or CYMK values ...

Feature representation of data (8)

General procedure that works very often:

- ▶ Given a set of “objects” (texts, graphs, images, emails, ...)
- ▶ Describe the objects by simple “features” that can be expressed as numbers
- ▶ Together, these objects give a feature vector $\in \mathbb{R}^d$.
- ▶ Note that often, the dimension d is very large ...

Feature representation of data (9)

In machine learning, the mapping $\Phi : \mathcal{X} \rightarrow \mathbb{R}^d$ that takes an abstract object X to its feature representation is called the **feature map**. It is usually denoted by Φ .

We have seen such a mapping when we considered regression with basis functions Φ_1, \dots, Φ_D . Taken together, they build a feature map $\Phi(X) := (\Phi_1(X), \dots, \Phi_D(X))^t : \mathcal{X} \rightarrow \mathbb{R}^D$.

All in all, the assumption that “data is in \mathbb{R}^d ” does make sense in very many applications.

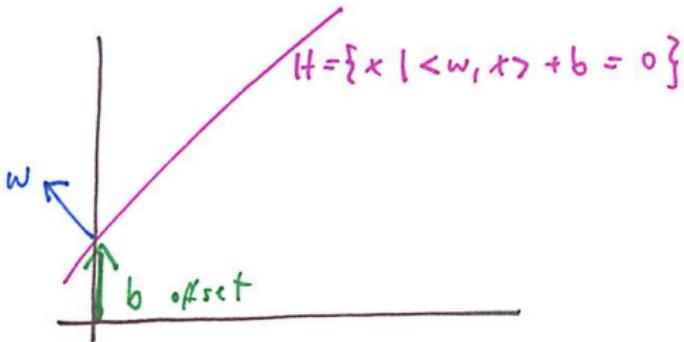
Hyperplanes in \mathbb{R}^d

Now let's come back to linear classification with hyperplanes.

- ▶ A hyperplane in \mathbb{R}^d has the form

$$H = \{x \in \mathbb{R}^d \mid \langle w, x \rangle + b = 0\}$$

where $w \in \mathbb{R}^d$ is the normal vector of the hyperplane and b the offset.



Classification using hyperplanes

To decide whether a point lies on the right or left side of a hyperplane, we use a decision function of the form

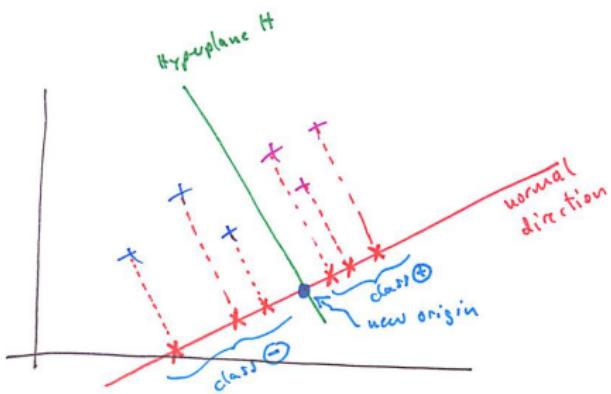
$$f(x) = \text{sign}(\langle w, x \rangle + b) \in \{\pm 1\}$$

Note that it is a convenient convention to use the class labels $+1$ and -1 (because we can then simply use the sign function).

Projection interpretation

Here is another way to interpret classification by hyperplanes:

- ▶ The function $\langle w, x \rangle$ projects the points x on a real line, the normal vector of the hyperplane.
- ▶ The term b shifts them along this line.
- ▶ Then we look at the sign of the result and classify by the sign



Crucial question

Now of course the crucial question is: given data, how to choose the separating hyperplane???

We are now going to see three basic approaches to do this:

- ▶ Linear discriminant analysis
- ▶ Logistic regression
- ▶ Linear support vector machines

Linear discriminant analysis

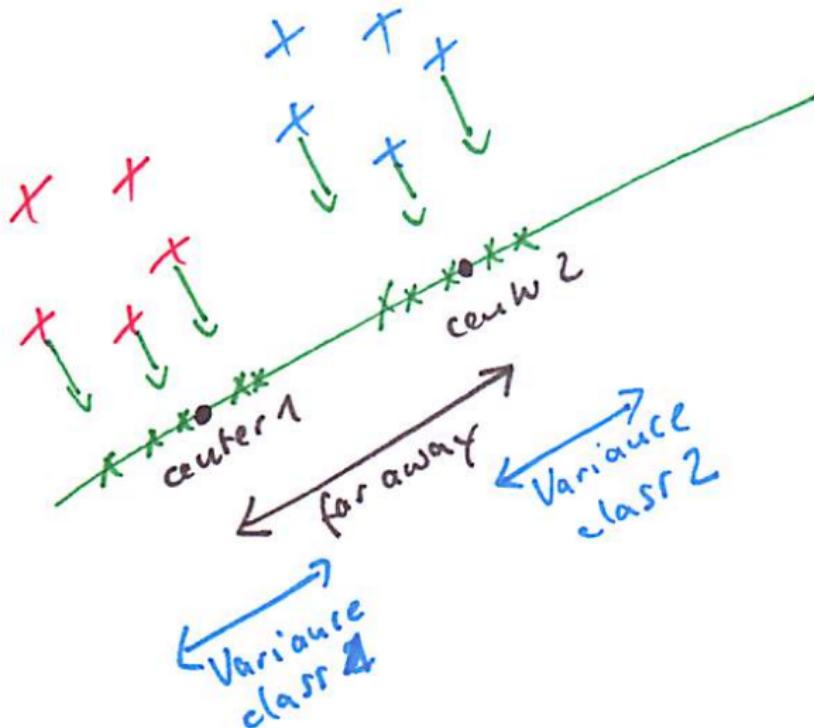
Literature:

- ▶ Hastie/Tibshirani/Friedman Sec. 4.3
- ▶ Duda / Hart

LDA: Geometric motivation

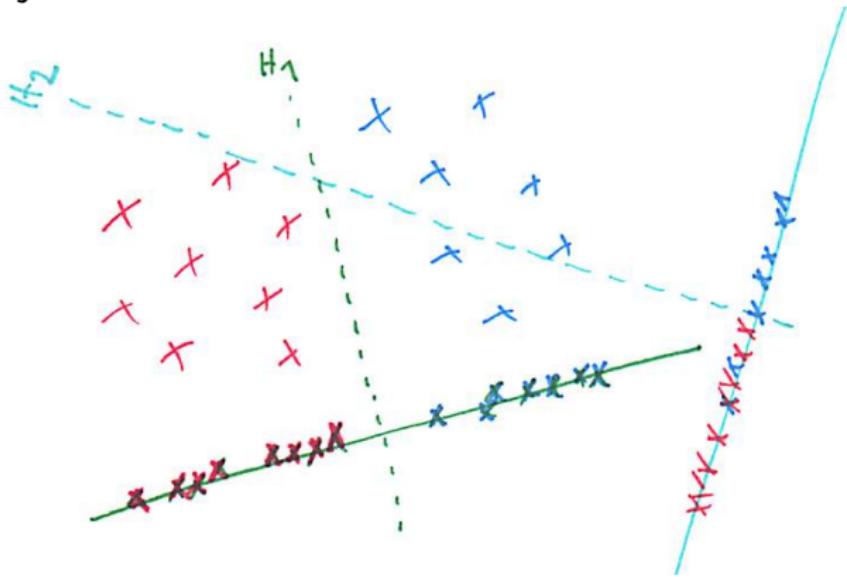
- ▶ Have seen: linear classification amounts to a one-dimensional projection.
- ▶ Here is now an idea to select the direction w on which we want to project. Namely, we try to achieve two things:
 - ▶ The class centers should be as far away from each other as possible
 - ▶ The class variances should be as small as possible.

LDA: Geometric motivation (2)



LDA: Geometric motivation (3)

Different projections: which one is better for classification?



Formally: the Fisher criterion

Define the following quantities for class +1:

- ▶ Let n_+ be the number of points in class 1
- ▶ Define the center of class 1 as

$$m_+ := \frac{1}{n_+} \sum_{\{i \mid Y_i=+1\}} X_i \in \mathbb{R}^d$$

Note that after projecting on w , the mean is given as $\langle w, m_+ \rangle$.

- ▶ Define the **within-class variance** after projecting on w as

$$\sigma_{w,+}^2 := \sum_{\{i \mid Y_i=+1\}} \left(\langle w, X_i \rangle - \langle w, m_+ \rangle \right)^2$$

Make the analogue definitions for class -1: ...

Formally: the Fisher criterion (2)

Now define the **Fisher criterion** as

$$J(w) = \frac{\langle w, m_+ - m_- \rangle^2}{\sigma_{w,+}^2 + \sigma_{w,-}^2}.$$

The idea of linear discriminant analysis is now to select w such that the Fisher criterion is maximized.

Fisher criterion in matrix form

We can write the Fisher criterion in matrix form as follows:

- ▶ Define the between-class covariance matrix as

$$C_B := (m_+ - m_-)(m_+ - m_-)^t \in \mathbb{R}^{d \times d}$$

- ▶ Define the total within-class covariance matrix as

$$\begin{aligned} C_W := & \sum_{\{i \mid Y_i=+1\}} (X_i - m_+)(X_i - m_+)^t \\ & + \sum_{\{i \mid Y_i=-1\}} (X_i - m_-)(X_i - m_-)^t \end{aligned}$$

- ▶ The Fisher criterion can now be rewritten as

$$J(w) = \frac{\langle w, C_B w \rangle}{\langle w, C_W w \rangle}$$

Solution vector w

Theorem 10 (Solution vector w^* of LDA)

The optimal solution of the problem $w^* := \operatorname{argmax}_{w \in \mathbb{R}^d} J(w)$ is given by

$$w^* = (C_W)^{-1}(m_+ - m_-).$$

Proof (sketch).

- Take the derivative:

$$\frac{\partial J}{\partial w}(w) = 2 \frac{C_B w \langle w, C_W w \rangle - C_W w \langle w, C_B w \rangle}{\langle w, C_W w \rangle^2}$$

Solution vector w (2)

- ▶ Setting this to 0 gives

$$\frac{\langle w, C_W w \rangle}{\langle w, C_B w \rangle} C_B w = C_W w$$

- ▶ Rewrite this (plug in the definition of C_B):

$$\underbrace{\frac{\langle w, C_W w \rangle}{\langle w, C_B w \rangle}}_{\in \mathbb{R}} (m_+ - m_-) \underbrace{(m_+ - m_-)^t w}_{\in \mathbb{R}} = C_W w$$

- ▶ Additionally, observe that $J(w)$ is invariant under rescaling of w , that is $J(w) = J(\alpha w)$ for $\alpha \neq 0$.
- ▶ So the solution is

$$w^* \propto (C_W)^{-1} (m_+ - m_-)$$

- ▶ We can check that the Hessian of $J(w)$ at w^* is negative definite, so w^* is indeed a maximum.



Determining b

So far, we only discussed how to find the normal vector w . How do we set the offset b ? (Recall that the hyperplane is $\langle w, x \rangle + b$).

The standard is to choose b , once w is known, as to minimize the training error.

LDA: motivation by Bayesian decision theory

We can also derive LDA as a special case of decision theory. Here is a sketch of the argument:

- ▶ Assume that the two classes follow a multivariate normal distribution with the same covariance matrix.
- ▶ According to Bayesian decision theory, we want to compare $P(Y = 1 \mid X = x)$ to $P(Y = -1 \mid X = x)$.
- ▶ To decide which one is larger, we consider the log-ratio of both terms: $\log \left(P(Y = 1 \mid X = x) / P(Y = -1 \mid X = x) \right)$.
- ▶ From this term one can analytically derive the Bayes classifier, which coincides with LDA :-)

LDA: Motivation by ERM

One can prove the following nice theorem:

Theorem 11 (LDA as ERM)

Consider the following problem of minimizing the least squares loss over the class of affine linear functions:

$$(w', b') := \underset{w \in \mathbb{R}^d, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^n (Y_i - \langle w, X_i \rangle + b)^2$$

Then $w' \propto w^*$, that is the solutions w' and w^* coincide up to a constant.

Proof: skipped.

LDA: Motivation by ERM (2)

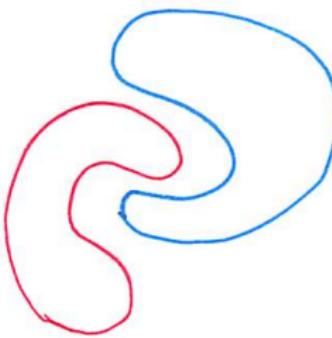
Comments:

- ▶ Important to note here: The least squares loss in this theorem is with respect to $\langle w, X_i \rangle + b$, not with respect to the sign of this expression (which is what we are ultimately interested in).
- ▶ In the theorem, the threshold b is chosen automatically as to minimize the L_2 -loss. However, the rule of minimizing the training error works better in practice.

Reason: the L_2 -error has not so much to do with the 0-1-loss (and our criterion for b tries to optimize the 0-1-loss rather than the L_2 -loss).

Limitations

- ▶ LDA does not work well if the classes are not “blobs”



- ▶ LDA does not work well if the variance of the two classes is very different from each other (remember, in the derivation of LDA based on Gaussian distributions we assumed equal variance for both classes).
- ▶ LDA tends to overfit (so far, we do not regularize). We'll discuss regularized versions later.

History

- ▶ A variant of this was first published by R. Fisher in 1936:
Fisher, R. A. (1936). The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics 7 (2): 179–188.
- ▶ LDA goes under various names: Linear discriminant analysis, Fisher's linear discriminant.
- ▶ R. Fisher is THE founder of modern statistics (design of experiments, analysis of variance, maximum likelihood, sufficient statistics, randomized tests, ...)

Further comments

- ▶ LDA can be generalized to multiclass problems as well. We'll skip it.

Summary: Linear discriminant analysis (LDA)

- ▶ Projection motivation: project in a direction that separates the classes well
- ▶ ERM motivation: minimize the least squares loss
- ▶ Model-based motivation: Bayes classifier under assumption of normal distributions with equal variances
- ▶ Leads to the Fisher criterion that should be minimized
- ▶ Can compute solution vector w analytically

Logistic regression

Literature: Hastie/Tibshirani/Friedman Section 4.4

Probabilistic approach to classification

- ▶ Bayesian decision theory: choose $f(x)$ according to whether $P(Y = +1 \mid X = x)$ is larger or smaller than $P(Y = -1 \mid X = x)$.
- ▶ Assume that the conditional probability $P(Y = +1 \mid X = x)$ has a certain functional form, that is it can be described by some function $f \in \mathcal{F}$ (for some appropriate \mathcal{F}).
- ▶ Now we want to find the function $f \in \mathcal{F}$ that “best explains our training data”.
- ▶ This amounts to selecting $f \in \mathcal{F}$ by

$$\operatorname{argmax}_{f \in \mathcal{F}} \prod_{i=1}^n P(f(X_i) = Y_i \mid X = X_i)$$

Probabilistic approach to classification (2)

- ▶ This is equivalent to the following problem (simply take $-\log$):

$$\operatorname{argmin}_{f \in \mathcal{F}} \sum_{i=1}^n \underbrace{-\log P(f(X_i) = Y_i \mid X = X_i)}_{=: \ell(X_i, f(X_i), Y_i)}$$

- ▶ This approach can be interpreted as **empirical risk minimization** with respect to this newly defined loss function ℓ .

Logistic regression works in this framework, for a particular linear model.

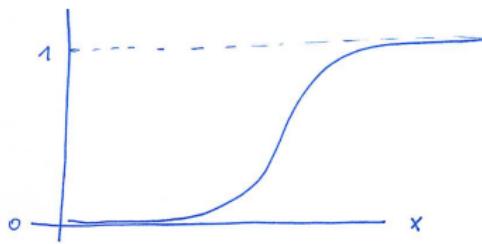
The logistic model

Logistic regression has a particular choice for the model:

- ▶ Assume that we have the following form:

$$P(Y = y \mid X = x) = \frac{1}{1 + \exp(-y(\langle w, x \rangle + b))}$$

where w and b are the parameters. Such a function is called a **logistic function** and looks as follows:



- ▶ Intuition: this is a “smoothed version” of a step function.

The logistic model (2)

- With this model, the “log odds” have the following linear form:

$$\log \left(\frac{P(Y = +1 \mid X = x)}{P(Y = -1 \mid X = x)} \right) = \langle w, \Phi(x) \rangle + b$$

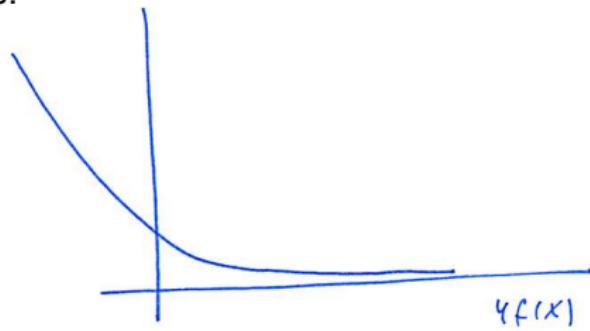
- By the argument on the previous slides, this corresponds to minimizing the following loss function:

$$\ell(X_i, f(X_i), Y_i) = \sum_{i=1}^n \log(1 + \exp(-Y_i f(X_i)))$$

with $f(X_i) = \langle w, X_i \rangle + b$.

The logistic model (3)

This loss function is called the **logistic loss function** and it looks as follows:



Logistic regression problem, formally

Formally, the problem of logistic regression is thus the following:

- ▶ Given $X_i \in \mathcal{X}$, $Y_i \in \mathbb{R}$.
- ▶ $\mathcal{F} = \left\{ \sum_{i=1}^D w_i \Phi_i(x) \mid w \in \mathbb{R}^D \right\}$.
(you can model an intercept b by adding a column of ones as we have seen before).
- ▶ Loss function: logistic loss function
- ▶ Learning principle: empirical risk minimization (no regularization yet)

Solution

- ▶ Bad news: there is no closed form solution for this problem.
- ▶ Good news: the logistic loss function is convex.
This can be proved by showing that the Hessian matrix is positive definite.
- ▶ So we can use our favorite convex solver to obtain the logistic regression solution.
- ▶ The standard technique in this case is the Newton-Raphson algorithm, but we won't discuss the details.

Adding regularization

- ▶ As in linear regression, we now might want to use a regularizer to avoid overfitting.
- ▶ Again we use $\Omega(f) = \|w\|_2^2$ (as in ridge regression) or $\Omega(f) = \|w\|_1$ (as in Lasso).
- ▶ Then the L_2 -regularized logistic regression is the problem to minimize

$$\frac{1}{n} \sum_{i=1}^n \log \left(1 + \exp(-Y_i \langle w, \Phi(X) \rangle) \right) + \lambda \Omega(f).$$

- ▶ This is now again a convex optimization problem in w and can be solved by standard convex solvers.
- ▶ More specialized (more efficient) solvers exist.

Summary: logistic regression

XXX