The linear model for regression and classification, Gradient Descent, Properties of the Gradient

Alexander Binder
with slides from Andreas Kleppe and Are C. Jensen

University of Oslo (UiO)

January 25, 2023



UiO : Department of Informatics
University of Oslo

Learning goals

- the four components for structuring many types of machine learning problems
- loss functions: ℓ_2-loss, zero$-$1-loss, Hinge-loss
- recap on generalization as goal: low loss on unseen data points.
- **?!** relationship between generalization and the central limit theorem
- **?!** recap on overfitting versus generalization

Learning goals

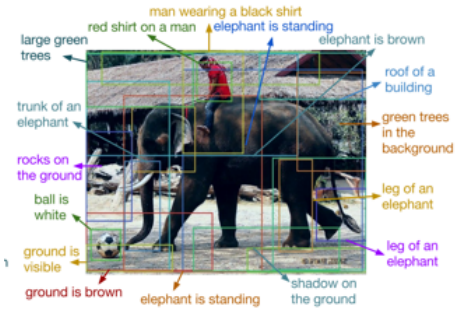
- **?!** be able to explain how the set of constant points $\{x : f(x) = c\}$ looks like for f being the linear model

Outline

1. Motivation
2. The four components to define a machine learning problem
3. Ordinary Least Squares as an intro example
4. Overfitting, Generalization and CLT
5. What does a linear mapping represent?
6. Using the Linear model for classification
7. A first method to solve linear regression

These section on motivation does not need to be understood fully (only this one section, the other sections you should try to understand) .
Many of the content of the motivation section will be treated in the next lectures in more detail.

Many tasks require information from images



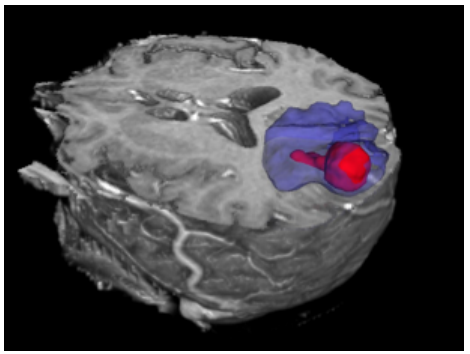




badminton







Object detection

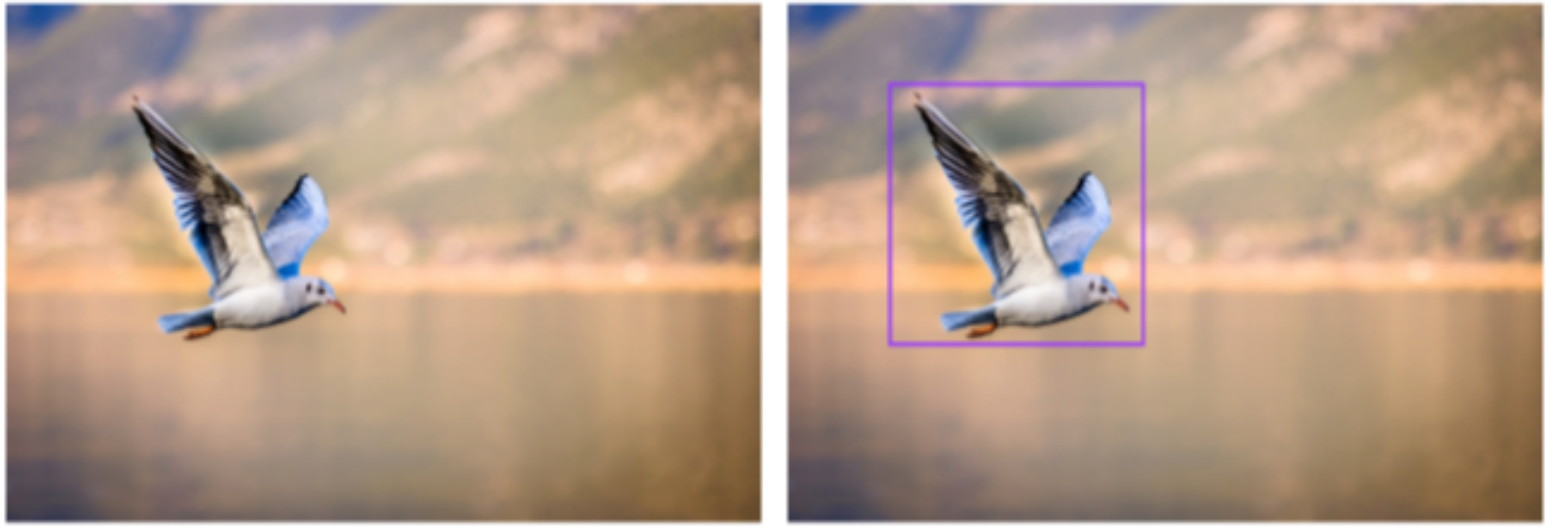
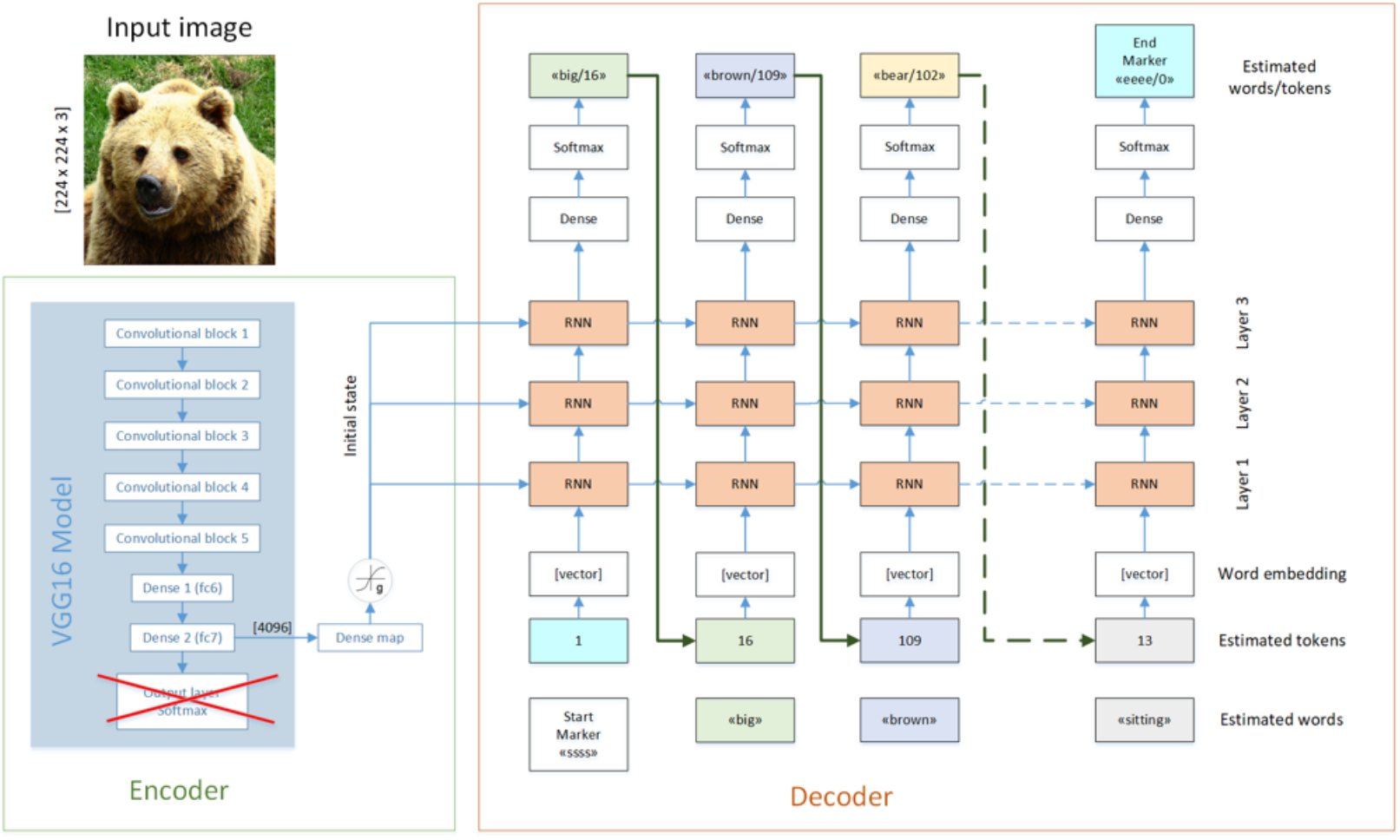


Figure 1: Seagull. Image source: https://www.pixabay.com

Instance segmentation



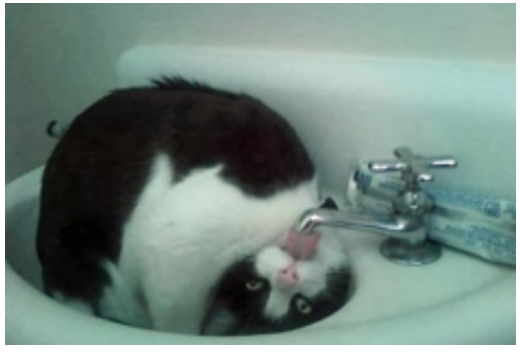
Combining recurrent and convolutional networks for image captioning





source: T. Karras et al. https://arxiv.org/pdf/1912.04958
https://www.youtube.com/watch?v=BIZg_PPuj_0

- The human visual system is so good in recognizing objects
- It is almost impossible to specify all variations of objects that can occurr
- Writing new code for each variation is not feasible

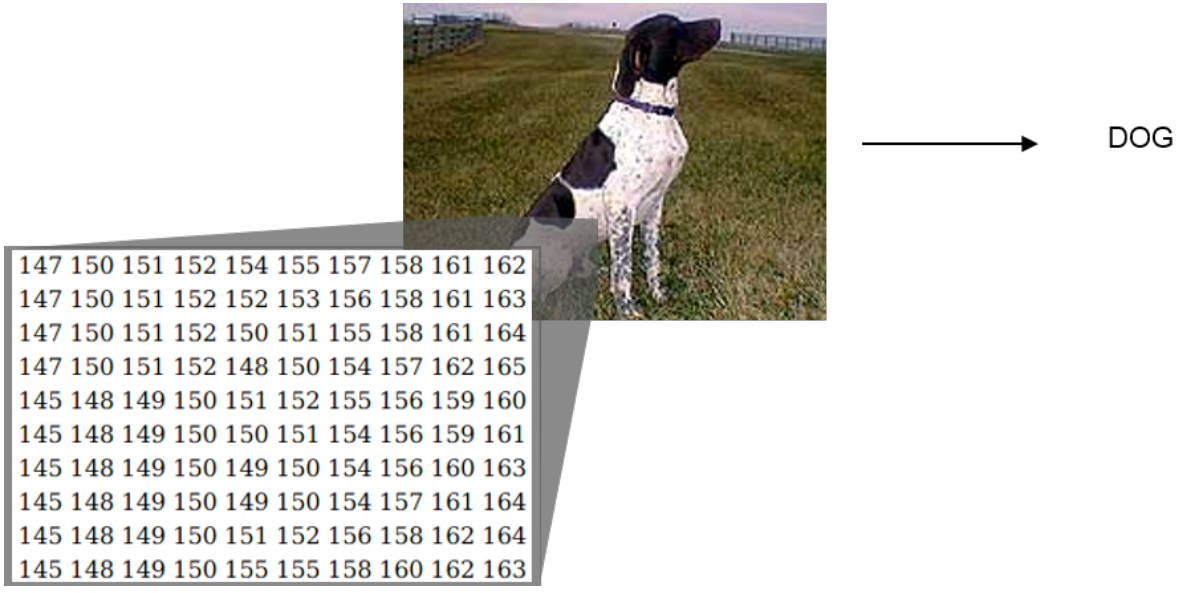








Example case: Image classification



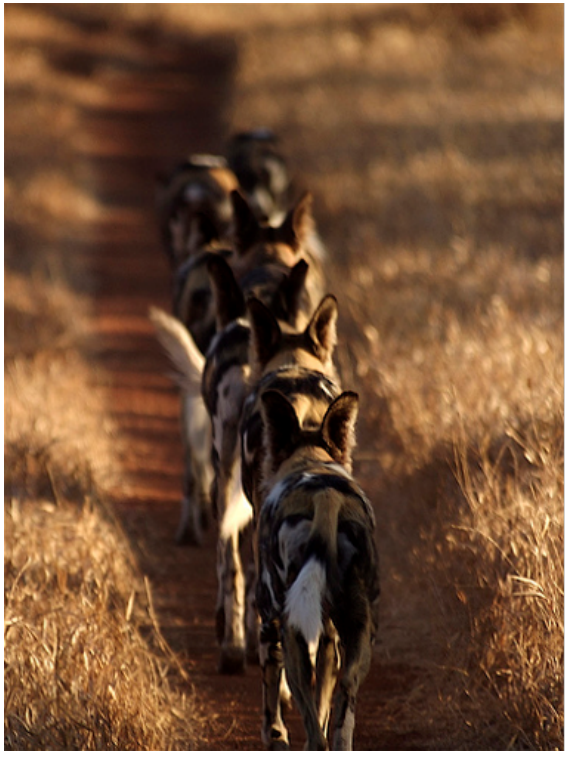
Task: use the entire image to classify the image into one of a set of known classes.

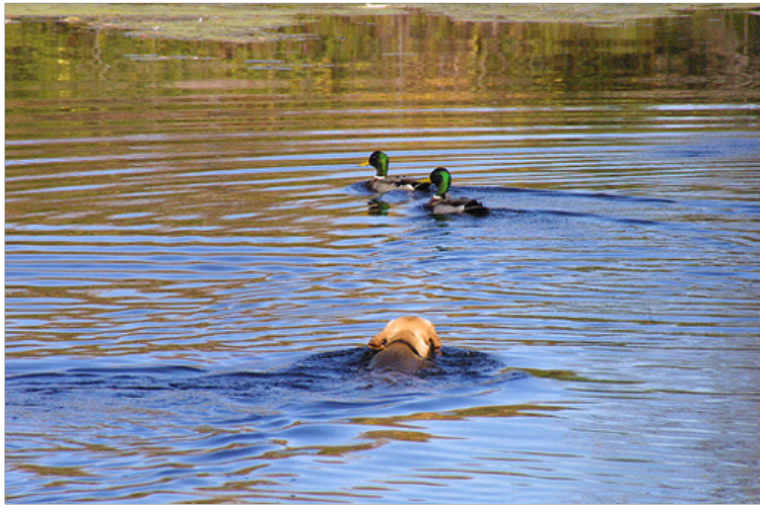
Which object is present in the image?

Challenges: Illumination



Challenges: Occlusion





Challenges: Deformation





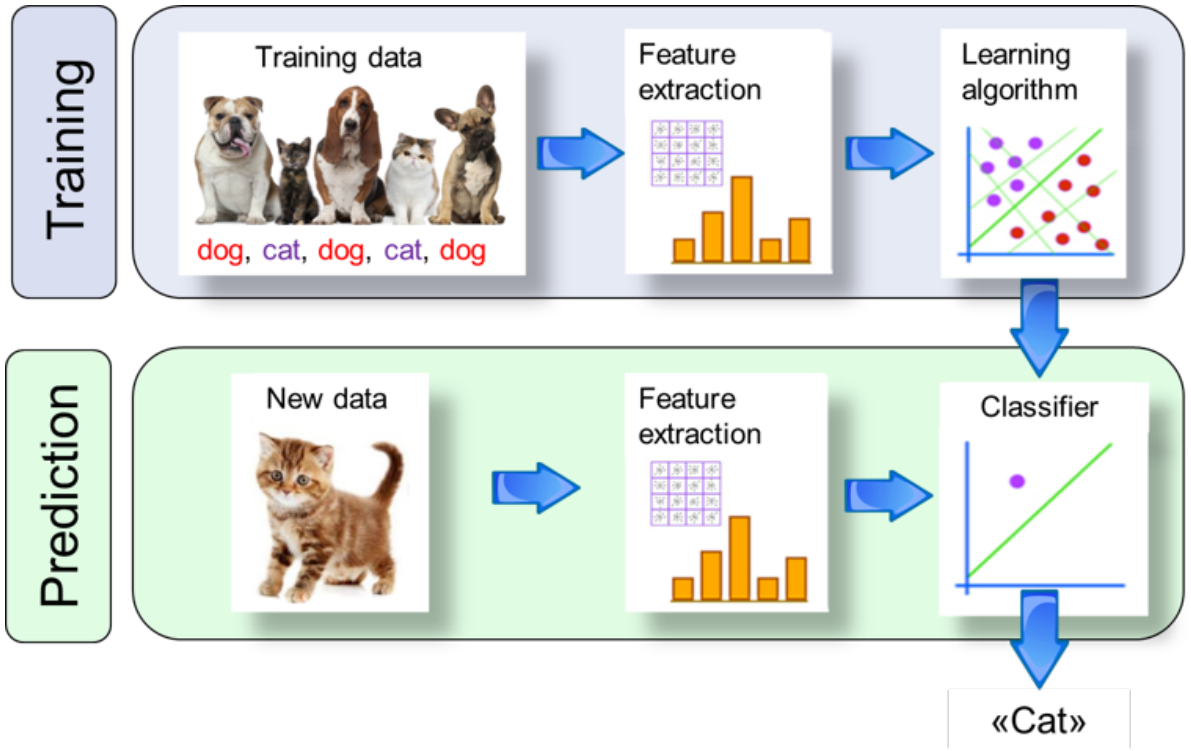


Challenges: Background clutter



Challenges: Intraclass variation





next goal: show why too simple methods may not work well.

- Image F, value $F(k, i, j)$ for pixel (i, j) in channel $k \in \{r, g, b\}$.
- **L1-distance:**

$$d_1(F_1, F_2) = \sum_k \sum_i \sum_j |F1(k, i, j) - F2(k, i, j)|$$

- **L2-distance:**

$$d_2(F_1, F_2) = \sqrt{\sum_k \sum_i \sum_j (F1(k, i, j) - F2(k, i, j))^2}$$

L2 is called Euclidean distance[1]

[1] 1. How it is related to a similarity measure? 2. which non-math limitation does that have for image data?

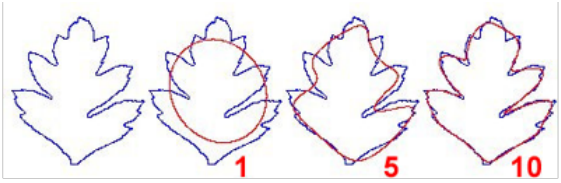
A baseline: K-Nearest-Neighbor classification

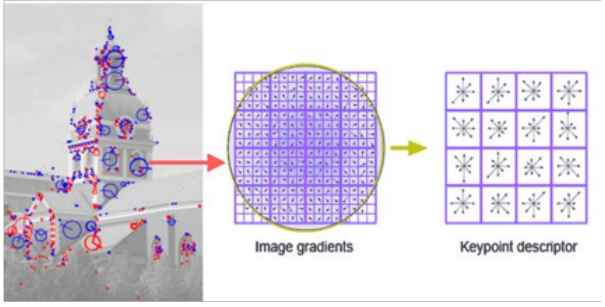
- Given a set of m training images $F_{(i)}$ with known class labels $y_{(i)}, i \in \{1, m\}$
- Classification of a new sample $F_{(new)}$ is done as follows:
  - Out of the m training images, find the k images with smallest distance (measured by L1 or L2) to the new sample $F_{(new)}$
  - Out of these k samples, identify the most frequent class label
  - Assign the class label for the sample $F_{(new)}$ as the most frequent class label among the m. Denote the predicted class as $\hat{y}_{(i)}$.
- Take a test set: measure Classification error on test set by counting the number of samples where the predicted class is unequal to the true class $\hat{y}_{(i)} \neq y_{(i)}$
- k must be selected a priori (try different values of k and choose the one with the lowest classification error using crossvalidation) (for 2-class problems k to be odd)

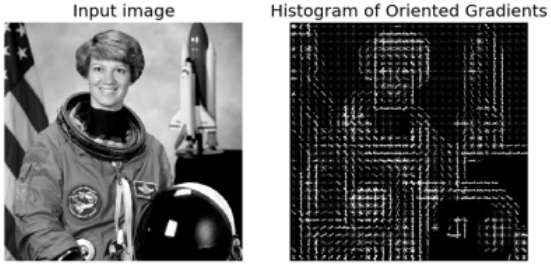
- If $k = 1$ (1NN-classification), each sample is assigned to the class of the closest sample in the training data set.
- In the limit of near infinite training data, this is theoretically a very good classifier for k very large,.
- This classifier involves no "training time", but the time needed to classify one pattern will depend on the number of training samples, as the distance to all points in the training set is computed (naive, non-hierarchical case).
- **Problem:** For image classification using the original pixel values as feature values, the results is not invariant to object position, scale, contrast!
- $\Rightarrow$ dont use raw pixels. Extract information, compute some features from the image

Good features are essential

- Features = Data representation
- Good features capture posterior belief about explanatory causes and underlying factors of variation
- Multitude of hand-designed features currently in use







Tradional: Handcrafted features + supervised classifier



Mainstream approach until recently for image (and speech) recognition:



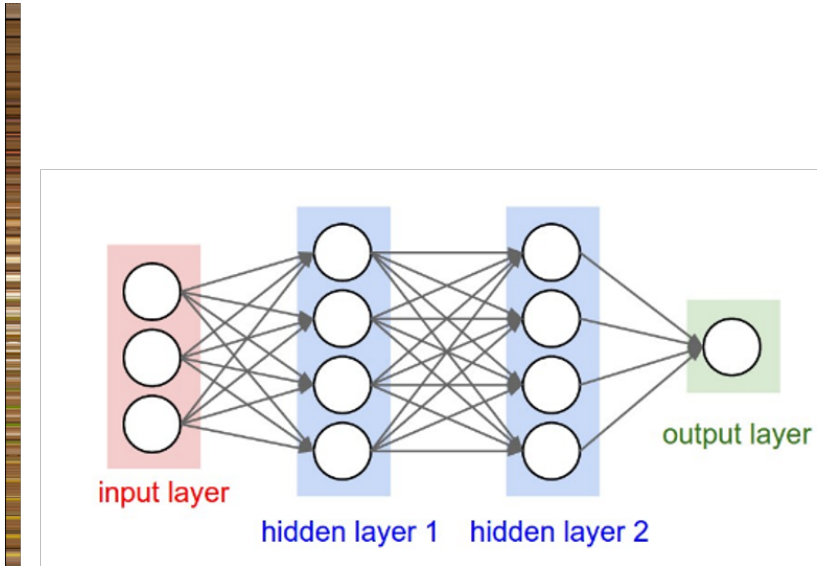
Deep learning: Train multiple stages end-to-end



Deep Learning: Why going beyond a fully connected net? Good Features?

Unwrap 2D image a 1D vector. One input node per pixel. Neurons in each layer are connected to every output from the previous layer.





Good Features? Fully connected neural network on images

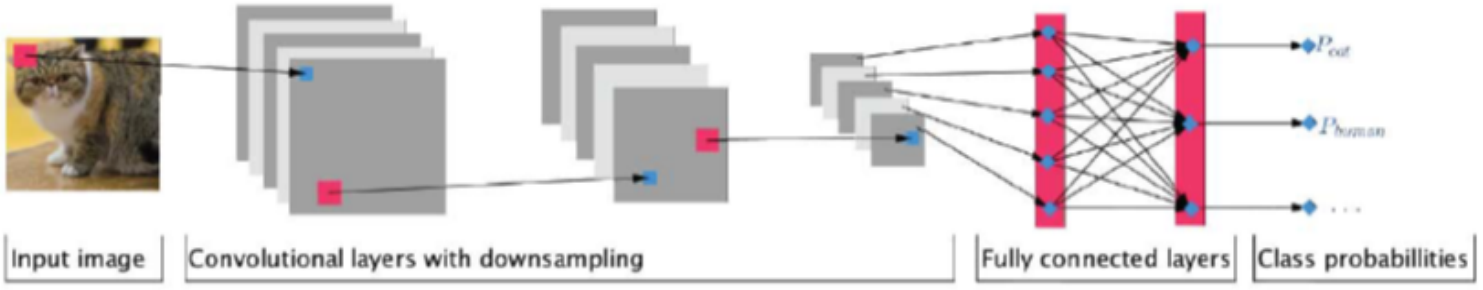
- Most image applications are absolute position invariant
- A fully connected network will have too many parameters and not be able to scale to normal size images and generalize





Good Features! Convolutional networks enforce similar response to neighboring pixels

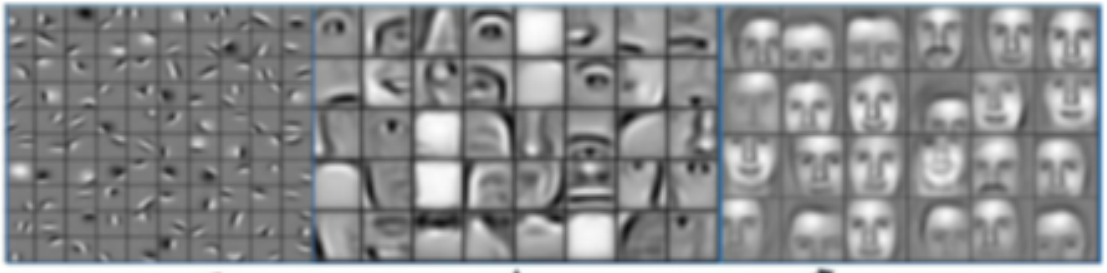
Apply same detection step in sliding windows.



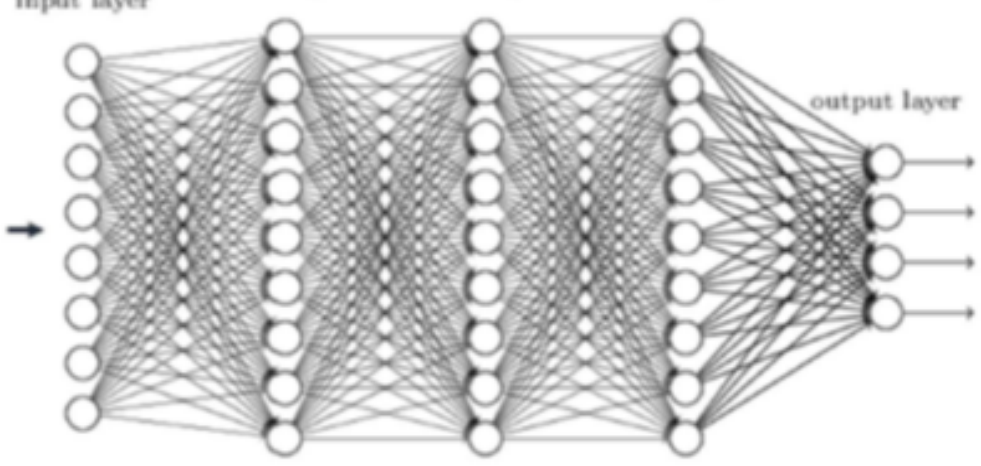
Able to detect same pattern in many locations $\rightarrow$ translation invariance

How do deep networks create invariance?

Deep neural networks learn hierarchical feature representations







What do the layers learn?

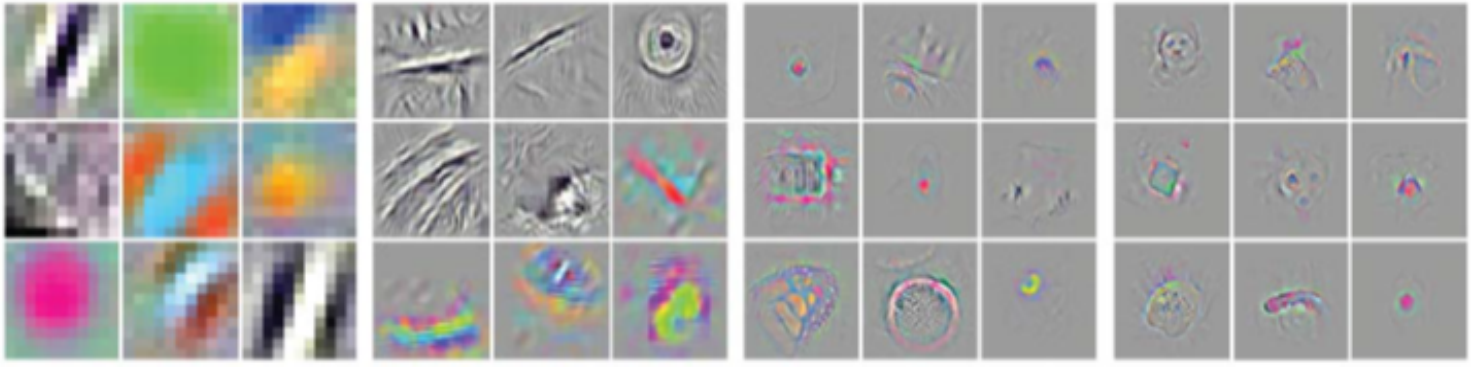
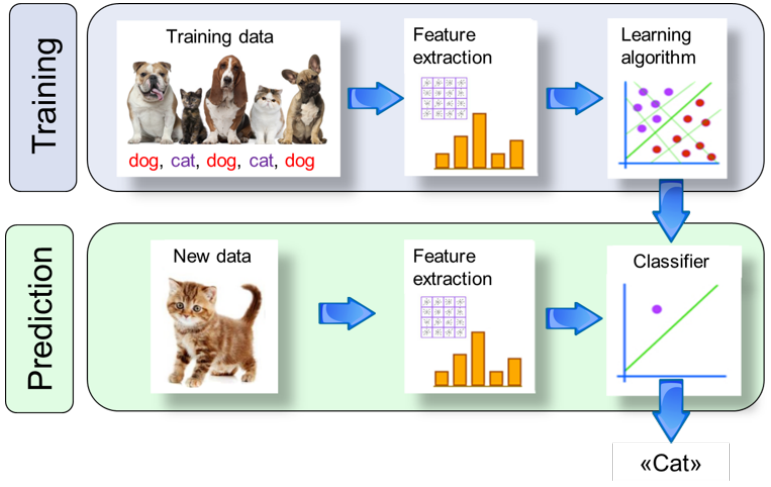


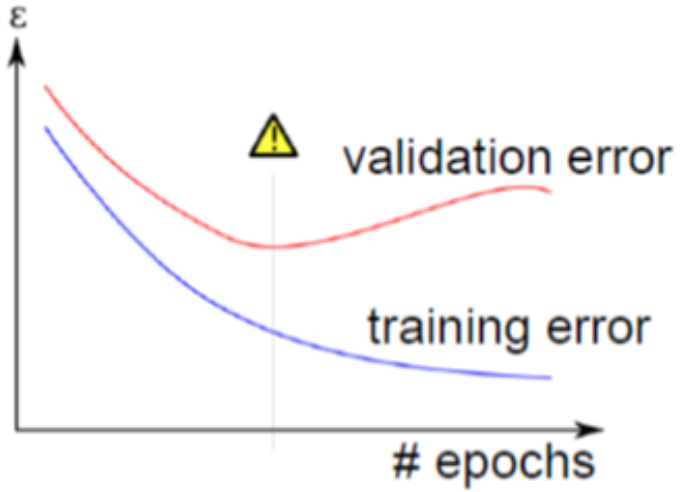
Figure 2: Visualized filter responses for some of the nodes of the first four layers of Alex-net (Krizhevsky et al. (2012)). The layers 1-4 are displayed from left to right. Since the filter responses cannot be directly related to the input image (due to the non-linearity in the network) the visualization technique by Zeiler and Fergus (2014) was used. The filter responses in this figure originally appeared in Zeiler and Fergus (2014) and is printed with permission from the authors of the original article.

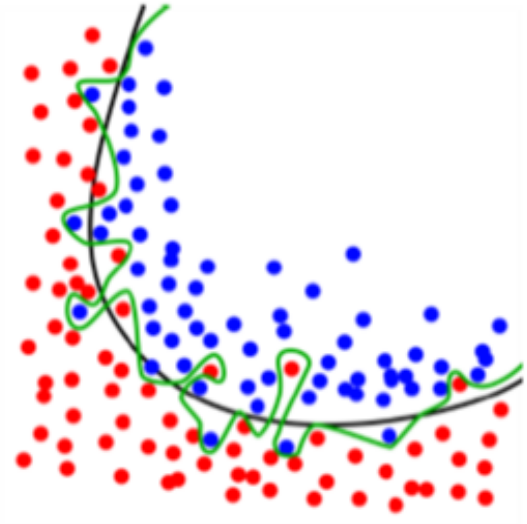
Training a machine learning system



- Training data: data with known prediction labels used to find the parameters of the classifier
- Validation data: data with known prediction labels used to select 1. hyperparameters and 2. architectures
- Test data: data with known prediction labels used once to estimate the error rate of the system

The danger of overfitting to training data

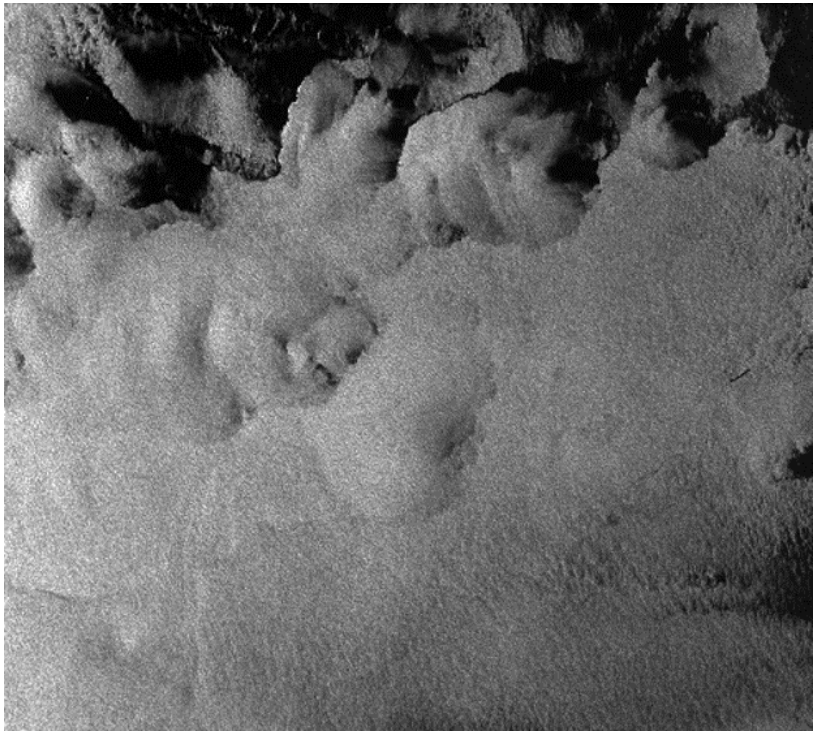




High variance

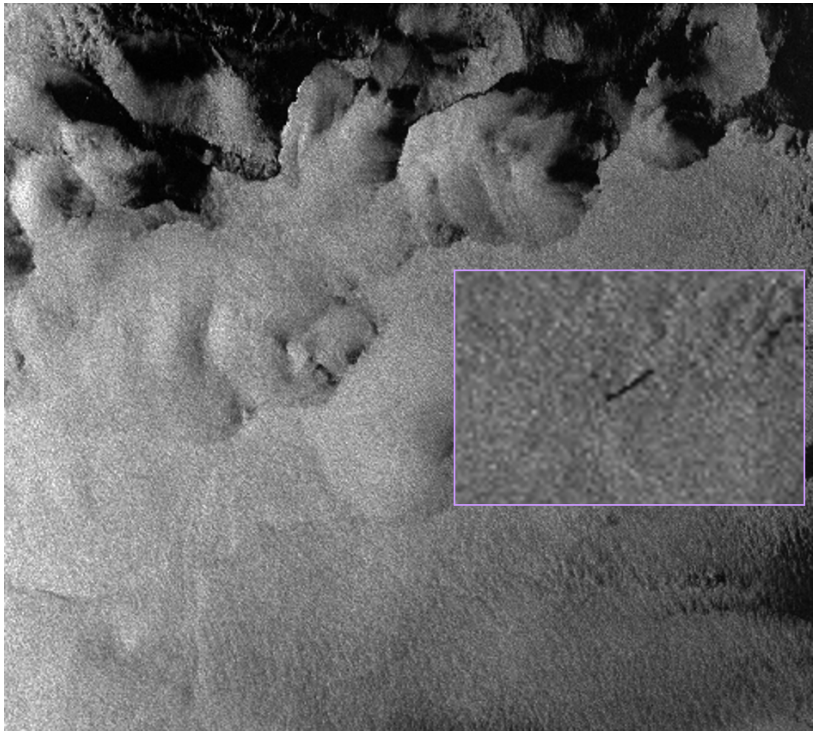
Working with skewed data

- Challenge: mostly data from "normal" classes, few samples from other classes (e.g. cancer, pollution etc.)
- Problem: How can we avoid that the networks fits the normal class and ignores the small but important class.



Working with skewed data - cont.

- Challenge: mostly data from "normal" classes, few samples from other classes (e.g. cancer, pollution etc.)
- Problem: How can we avoid that the networks fits the normal class and ignores the small but important class.



Domain shift

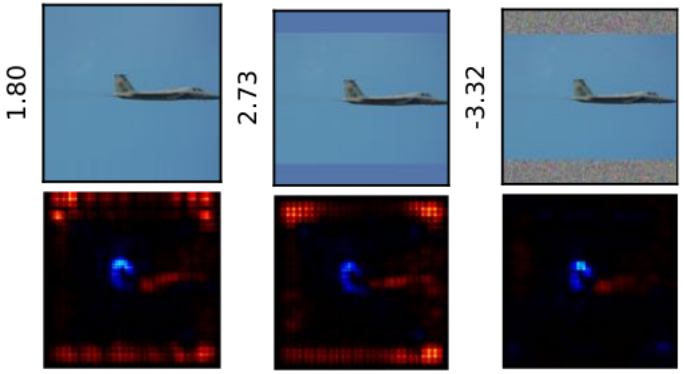




Ohen ghoor!

With power comes responsibility!

- Models with millions of parameters can fit to almost anything!
- Risk: make prediction based on artefacts/undesired queues which correlate with training labels



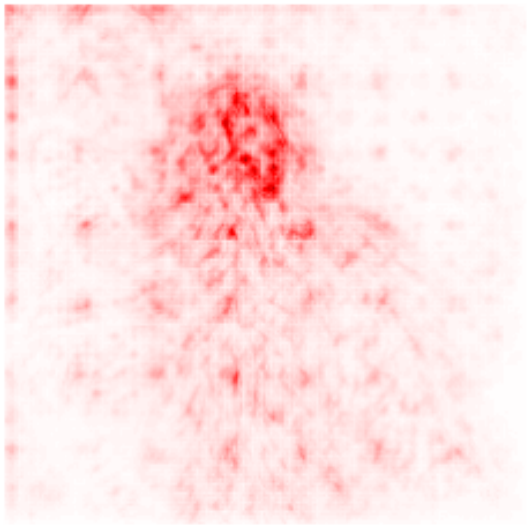
https://www.reuters.com/article/us-amazon-com-jobs-automation-insig amazon-scraps-secret-ai-recruiting-too

- Seeing inside the black box: why did the computer decide the given class?
- How do we evaluate the results? How to find out what we have not learned yet?
- When not to make a prediction ? – outliers
- What are the ethical sides of the problem and the decisions?

Explaining the results



Afghan hound, Afghan



Limitations of deep learning

- overfitting risk with small sample sizes (ideas: fine-tuning, cross-validation instead of train-val split)
- Complex tasks can be difficult to learn, particularly memory-intensive tasks
- Often easy to fool the network: images which to humans look like the right thing.
- You have only learnt what is covered by your training data
- Deep learning requires good knowledge of technical details, and how to train efficiently e.g. see for a more complex architecture https://arxiv.org/abs/1905.08233

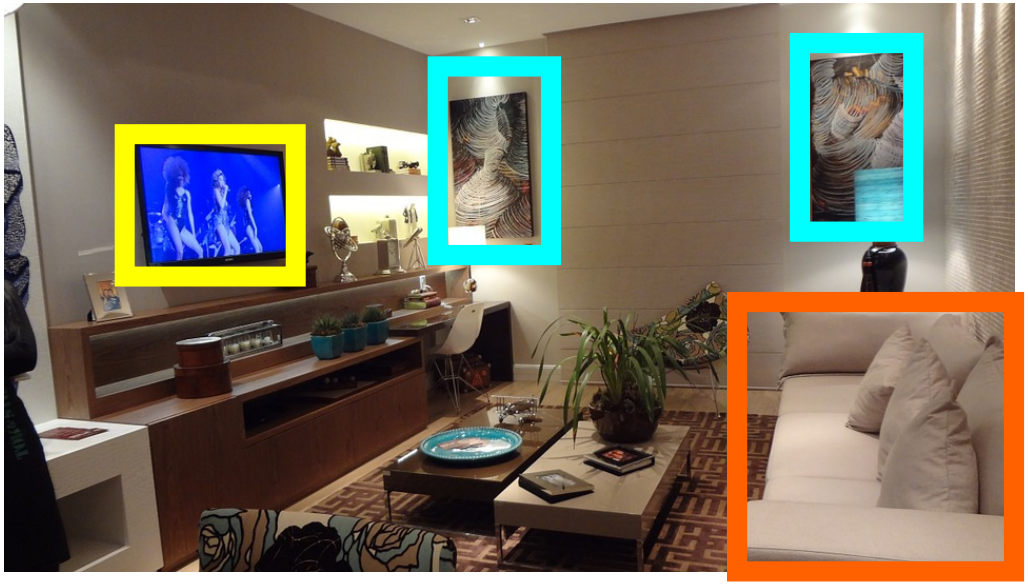
Outline

1. Motivation
2. The four components to define a machine learning problem
3. Ordinary Least Squares as an intro example
4. Overfitting, Generalization and CLT
5. What does a linear mapping represent?
6. Using the Linear model for classification
7. A first method to solve linear regression

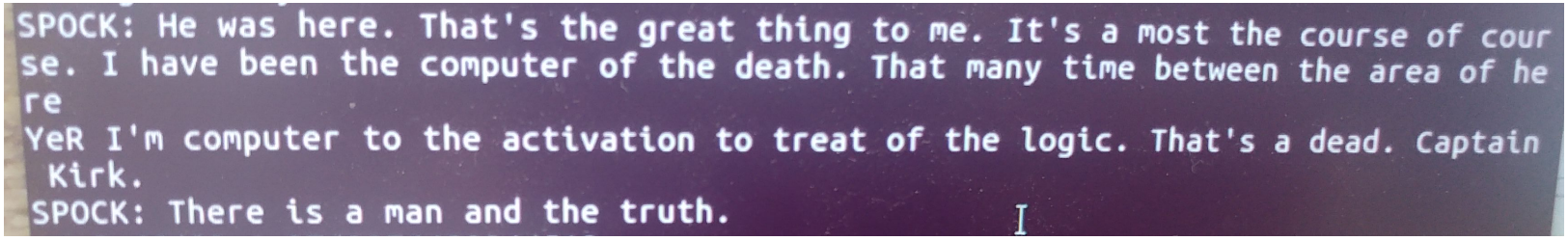
Predictor f : Input space $\longrightarrow$ Output space.

- model has trainable parameters w: $f = f_w$
- Use (labeled! ...) training data and a loss function to optimize parameters w
- Prediction is correct with a certain probability of mistake.

Detection in Images: class label and position



- Input: image.
- Output: a number of bounding boxes



- Input: sequence of words $(w_1, \ldots, w_K)$
- Output: sequence of words $(w_1, \ldots, w_L)$
  (Example: "The text is about Star Trek. It mentions Spock and James T. Kirk.")
- Input: sequence of words $(w_1, \ldots, w_K)$
- Output: set of words $\{w_1, \ldots, w_L\}$ – setup can be multi-label classification, often sequential outputs from a RNN
  (Example: "Star Trek, RNN-generated nonsense")



D3: Battle



D4: Battle 2

- Input: sequence of states (health, ammo, images of view)^K
- Output: next action (move left, fire, ...)
- Dosovitsky et al, ICLR 2017, https://arxiv.org/pdf/1611.01779.pdf



credit: visualqa.org

- Input: Image (+ sequence of words for VQA)
- Output: set of words – setup can be multi-label classification, often sequential outputs from a RNN

What does one need for defining a discriminative machine learning problem?

four basic components of a machine learning problem

- I/O: Model for Input space, Model for output space
- prediction model: define a class of prediction models
- Loss: define a loss function to measure difference: prediction of the model versus ground truth in the data points
- Optimizer: define an algorithm for updating model parameters using (labelled) training data / for finding a *good* prediction model

What input and output space can be used for the examples above?

Outline

1. Motivation
2. The four components to define a machine learning problem
3. Ordinary Least Squares as an intro example
4. Overfitting, Generalization and CLT
5. What does a linear mapping represent?
6. Using the Linear model for classification
7. A first method to solve linear regression

A. http://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set

- X_1 = the transaction date
- X_2 = the house age (unit: year)
- X_3 = the distance to the nearest MRT station (unit: meter)
- X_4 = the number of convenience stores in the living circle on foot (integer)
- X_5 = the geographic coordinate, latitude. (unit: degree)
- X_6 = the geographic coordinate, longitude. (unit: degree)
- The output is as follows Y = house price per unit of area

http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime
http://archive.ics.uci.edu/ml/datasets/Concrete+Slump+Test
https://en.wikipedia.org/wiki/Concrete_slump_test
http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset

- commonly: $\mathcal{X} = \mathbb{R}^d$
- single regression target: $\mathcal{Y} = \mathbb{R}^1$
- multiple regression targets: $\mathcal{Y} = \mathbb{R}^k$

The prediction mapping for Linear regression

Linear function without/with a bias

$$f_w(x) = x \cdot w \qquad = \sum_{d=1}^{D} x_d w_d \qquad , \; w \in \mathbb{R}^{D \times 1}$$

$$f_{w,b}(x) = x \cdot w + b \quad = \sum_{d=1}^{D} x_d w_d + b \quad , \; w \in \mathbb{R}^{D \times 1}, b \in \mathbb{R}^1$$

predicts using a weighted sum of features x_d

Loss function to measure quality of prediction for a data sample which is here a pair (x, y): squared ℓ_2-loss, squared euclidian distance

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

Reasons:

- $\ell(f(x), y) = 0 \Leftrightarrow f(x) = y$
- symmetry: capture deviations on both sides of the real ground truth value y
- simple derivative

Lets skip this step for now. Consider first:

- how can a linear mapping be used for classification
- what is the goal when optimizing? what is a good linear mapping given data?
- what does the linear mapping represent

The prediction mapping for Classification with a linear model

Linear function without/with a bias

$$f_w(x) = x \cdot w = \sum_{d=1}^{D} x_d w_d, \;\; w \in \mathbb{R}^{D \times 1}$$

$$f_{w,b}(x) = x \cdot w + b = \sum_{d=1}^{D} x_d w_d + b, \;\; w \in \mathbb{R}^{D \times 1}, b \in \mathbb{R}^1$$

$$g_{w,b}(x) = \text{sign}(f_{w,b}(x)) \in \{-1, +1\}$$

Loss function to measure quality of prediction for a data sample which is here a pair (x, y): 0-1-loss

$$\ell(f(x), y) = \mathbb{1}[f(x) \neq y]$$

- counts mispredictions
- contra: uninformative derivative

Outline

1. Motivation
2. The four components to define a machine learning problem
3. Ordinary Least Squares as an intro example
4. Overfitting, Generalization and CLT
5. What does a linear mapping represent?
6. Using the Linear model for classification
7. A first method to solve linear regression

- You train linear regression on training data D_n consisting of samples (x_i, y_i)
- you find parameters (w^*, b^*) which minimize your ℓ_2-loss on the training data:
  find (w^*, b^*) defining a mapping $f_{(w^*,b^*)}$ such that

$$\hat{L}(f_{(w^*,b^*)}, D_n) = \frac{1}{n} \sum_{(x_i,y_i)\in D_n} \ell(f_{(w^*,b^*)}(x_i), y_i) \to \text{low}$$

- What is a good mapping? Low ℓ_2-loss on the training data?

Goal of ML is to find prediction models f which "generalize well".

Generalization – qualitative view I

A mapping generalizes well, if it gives low prediction errors on unseen data samples.

To generalize well in a qualitative form:

- we find parameters (w^*, b^*) defining a mapping f which for most draws of new unseen data samples T

$$\hat{L}(f, T) = \frac{1}{n} \sum_{(x_i, y_i) \in T} \ell(f(x_i), y_i) \rightarrow \text{low}$$

Informal so far: Find parameters w, b of the model which generalize well to new, unseen data points.

We need a way to model *new, unseen data points*.

We need a way to model *new, unseen data points*.

Assumption 1: We can draw test data sets T_n from our data source. A test data set consists of n pairs (x_i, y_i):

$$T_n = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} = \{(x_i, y_i), i = 1, \ldots, n\}$$

x_i is the input feature, y_i is the ground truth label to it (here: the regression value which x_i is expected to have.)

Assumption 2: We cannot predict which samples (x_i, y_i) we will obtain from our data source as new, unseen data points. Therefore, we model the uncertainty of data points by drawing them from a probability distribution.

$$(x_i, y_i) \sim P_{test}.$$

Next: how can drawing $(x_i, y_i) \sim P$ be modeled in a concrete example ?

An Example for such a generating probability

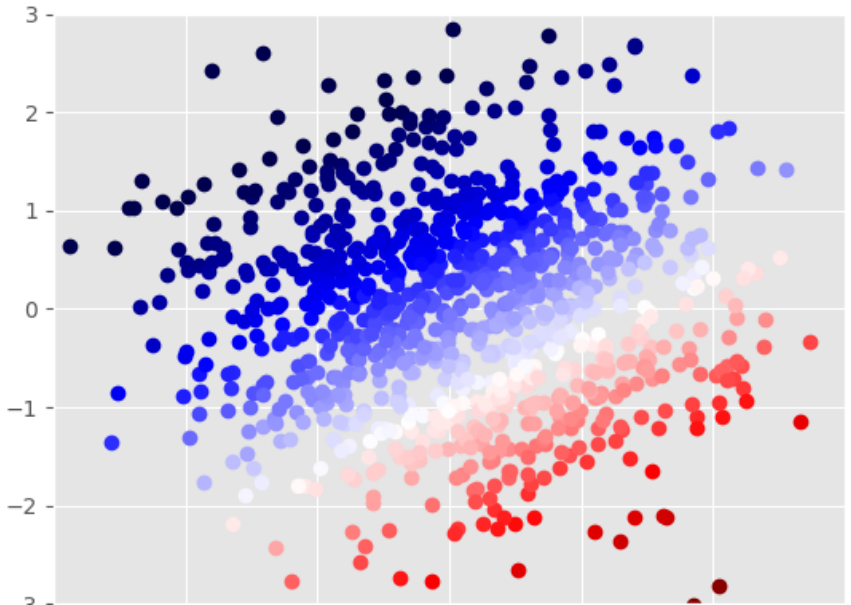
for input samples $x \in [0,1] \times [0,1] \subset \mathbb{R}^2$, and regression labels $y \in \mathbb{R}^1$:

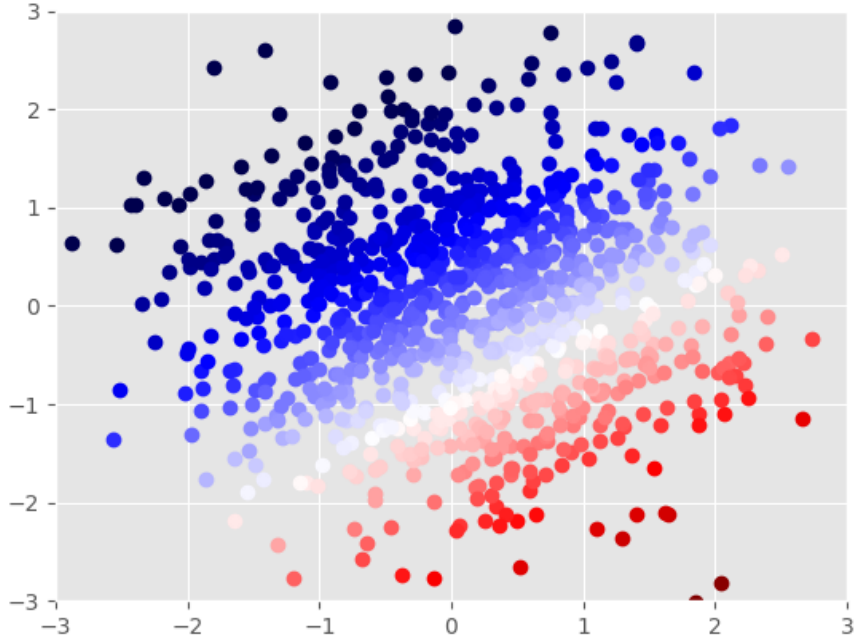
$$p(x) = Normal\left(\mu = (0,0), \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$$

$$p(y|x) = Normal(\mu(x), \sigma^2 = 0.1)$$

$$\mu(x) = 2x_1 - 3x_2 + 7 \in \mathbb{R}^1$$

$$P_{test}(x,y) = p(x)p(y|x)$$





Drawing in practice:

- draw $x \sim p(x)$
- draw $y \sim p(y|x)$
- what is the distribution of (x, y) concatenated??

side note:

- if you draw $x \sim p_x$, $y \sim p_y$, $z \sim p_z$ and concatenate the vectors into (x, y, z), then the resulting concatenation is distributed as the product

$$(x, y, z) \sim p_x p_y p_z$$

  We use that to conclude that if $x \sim p(x)$, $y \sim p(y|x)$ then $(x, y) \sim p(x)p(y|x)$
- the product $p(x)p(y|x)$ defines a joint distribution over (x, y). Reason: the math defining conditional and joint distributions:

$$p(x)p(y|x) = P(x, y)$$

  This will be our test distribution $P_{test}(x, y)$

An Example for such a generating probability

for input samples $x \in [0,1] \times [0,1] \subset \mathbb{R}^2$, and regression labels $y \in \mathbb{R}^1$:

$$
\begin{aligned}
p(x) &= \mathit{Unif}([0,1] \times [0,1]) \\
p(y|x) &= \mathit{Normal}(\mu(x), \sigma^2 = 0.1) \\
\mu(x) &= 2x_1 - 3x_2 + 7 \in \mathbb{R}^1 \\
P_{\mathit{test}}(x,y) &= p(x)p(y|x)
\end{aligned}
$$

Drawing in practice:

- draw $x \sim p(x)$
- draw $y \sim p(y|x)$
- this implies: have $(x,y) \sim p(x)p(y|x) = P_{\mathit{test}}(x,y)$

By drawing n times in this way, one can obtain a training dataset $D_n = \{(x_i, y_i), i = 1, \ldots, n\}$ or a test dataset T_n of independent samples.

Goal of ML is to find prediction models f which "generalize well".

Generalization – qualitative view II

A mapping generalizes well, if it gives low prediction errors on unseen data samples.

To generalize well in a qualitative form:

- we find parameters (w^*, b^*) defining a mapping f which for most draws of test datasets $T_n \sim P_{test}$ produces predictions with low average errors on the test sets T_n:

$$\hat{L}(f, T_n) = \frac{1}{n} \sum_{(x_i, y_i) \in T_n} \ell(f(x_i), y_i) \to \text{low}$$

Generalization for regression with ℓ_2-loss

Plug in the ℓ_2 loss from above:

$$\hat{L}(f, T_n) = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2 \to \text{low}$$

on new unseen test data for most draws of test data sets.

We do not specify here what "low" means. From a practitioners perspective it would be predictions, with errors such that the effort to correct them is low enough, so that using this predictor is productive.

The core assumption in machine learning

- we model the uncertainty in new unseen data by modeling it as coming from a probability distribution P_{test}.
- In practice we never know the probability distribution P_{test} which is the model for generating our data samples. We only have a finite set of data samples drawn from it.

important:

- trying to generalize information from a **finite-sized and noisy** test dataset to P_{test} and expectations leads to overfitting which means lack of generalization
- next: show this problem in a simple example

A simple classification setup:

- Example $X = [0, 1]$, $Y \in \{0, 1\}$, $\mathcal{F} = \{f_0(x) = 0, f_1(x) = 1\}$
- Assumption: The data (x, y) is distributed as: $P_y(Y = 0|X = x) = 0.6$. $P_x(x) = U([0, 1])$
- What is the best mapping ? Want $f \in \mathcal{F}$ with minimal classification error

$$E_{(x,y)\sim P_x P_y}[\mathbb{I}(f(x) \neq y)] = \begin{cases} 0.4 & \text{for } f_0 \\ 0.6 & \text{for } f_1 \end{cases}$$

$\Rightarrow$ choose $f_0(x) = 0$

- Example $X = [0, 1]$, $Y \in \{0, 1\}$, $\mathcal{F} = \{f_0(x) = 0, f_1(x) = 1\}$
- Assumption: The data (x, y) is distributed as:
  $P_y(Y = 0|X = x) = 0.6$. $P_x(x) = U([0, 1])$
- Question: If we have only one training sample (x_1, y_1),
  $+$ we use y_1 as statistic to choose between $f_0(x) = 0, f_1(x) = 1$,
  what is the probability to choose the suboptimal predictor $f_1(x) = 1$?

- Example $X = [0, 1]$, $Y \in \{0, 1\}$, $\mathcal{F} = \{f_0(x) = 0, f_1(x) = 1\}$
- Assumption: The data (x, y) is distributed as:
  $P_y(Y = 0|X = x) = 0.6$. $P_x(x) = U([0, 1])$
- Question: If we have only one training sample (x_1, y_1),
  $+$ we use y_1 as statistic to choose between $f_0(x) = 0, f_1(x) = 1$,
  what is the probability to choose the suboptimal predictor $f_1(x) = 1$?
- 0.4! Why?

Example $X = [0, 1]$, $Y \in \{0, 1\}$, $\mathcal{F} = \{f(x) = 0, f(x) = 1\}$

Assumption: The data (x, y) is distributed as:
$P_y(Y = 0|X = x) = 0.6$. $P_x(x) = U([0, 1])$

- If we have three training samples (x_i, y_i), and, we use y_1 as statistic to choose between $f_0(x) = 0, f_1(x) = 1$, what is the probability to choose the suboptimal predictor $f_1(x) = 1$?

Example $X = [0, 1]$, $Y \in \{0, 1\}$, $\mathcal{F} = \{f(x) = 0, f(x) = 1\}$

Assumption: The data (x, y) is distributed as:
$P_y(Y = 0|X = x) = 0.6$. $P_x(x) = U([0, 1])$

- If we have three training samples (x_i, y_i), and, we use y_1 as statistic to choose between $f_0(x) = 0, f_1(x) = 1$, what is the probability to choose the suboptimal predictor $f_1(x) = 1$?
- if 2 oder 3 of the $y_i = 1$ take value 1
- $0.4^3 + \binom{3}{1} 0.4^2\, 0.6 = 0.352 < 0.4$

Example $X = [0, 1]$, $Y \in \{0, 1\}$, $\mathcal{F} = \{f(x) = 0, f(x) = 1\}$

Assumption: The data (x, y) is distributed as:
$P_y(Y = 0|X = x) = 0.6$. $P_x(x) = U([0, 1])$

- If we have three training samples (x_i, y_i), and, we use y_1 as statistic to choose between $f_0(x) = 0, f_1(x) = 1$, what is the probability to choose the suboptimal predictor $f_1(x) = 1$?
- if 2 oder 3 of the $y_i = 1$ take value 1
- $0.4^3 + \binom{3}{1} 0.4^2\, 0.6 = 0.352 < 0.4$
- For 5 examples: if 3,4,5 of the $y_i = 1$ take value 1
- For 5 examples: $0.4^5 + \binom{5}{1} 0.4^4\, 0.6 + \binom{5}{2} 0.4^3\, 0.6^2 = 0.31744 < 0.35$

Example $X = [0,1]$, $Y \in \{0,1\}$, $\mathcal{F} = \{f(x) = 0, f(x) = 1\}$

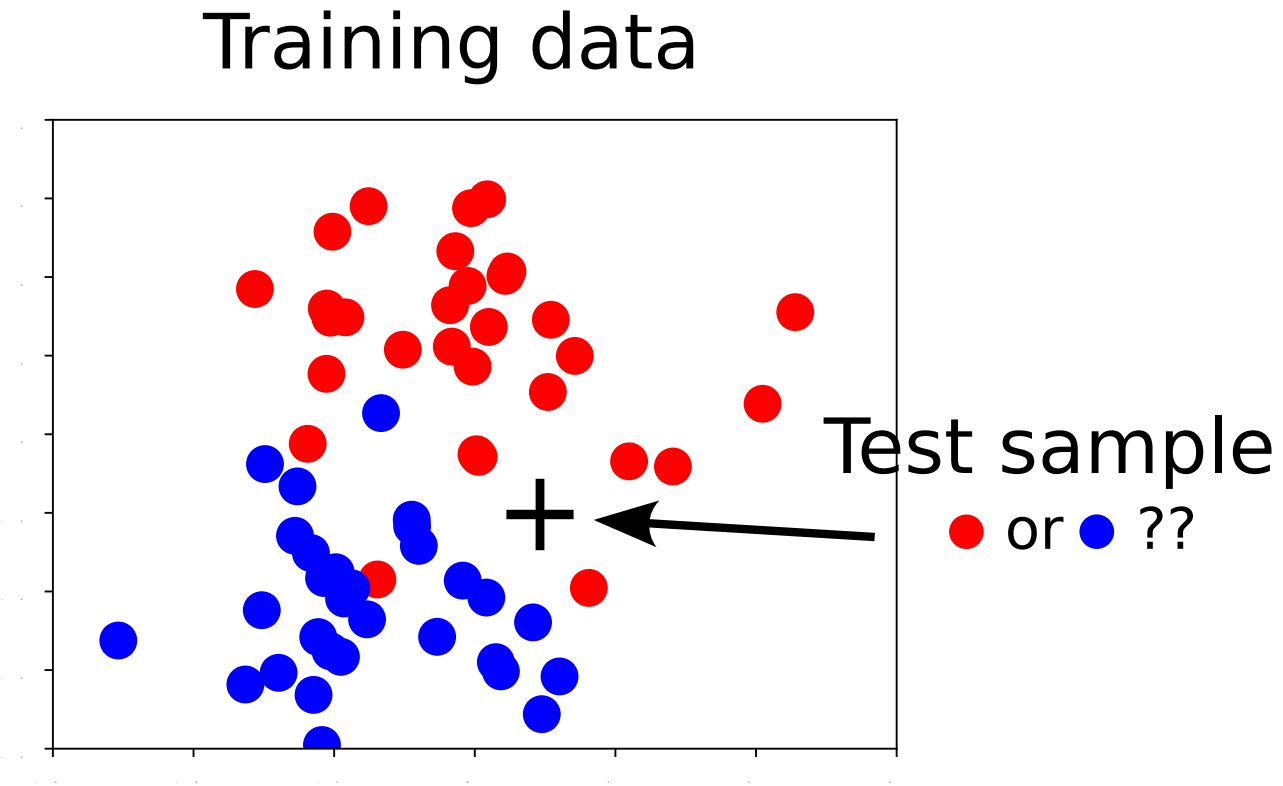
Assumption: The data (x, y) is distributed as:
$P_y(Y = 0|X = x) = 0.6$. $P_x(x) = U([0,1])$

- if we have $2n+1$ examples (x_i, y_i),what is the probability to choose the suboptimal predictor $f_1(x) = 1$?
- Binomially distributed with $p = 0.4$, $2n+1$ und observed values $\geq n+1$

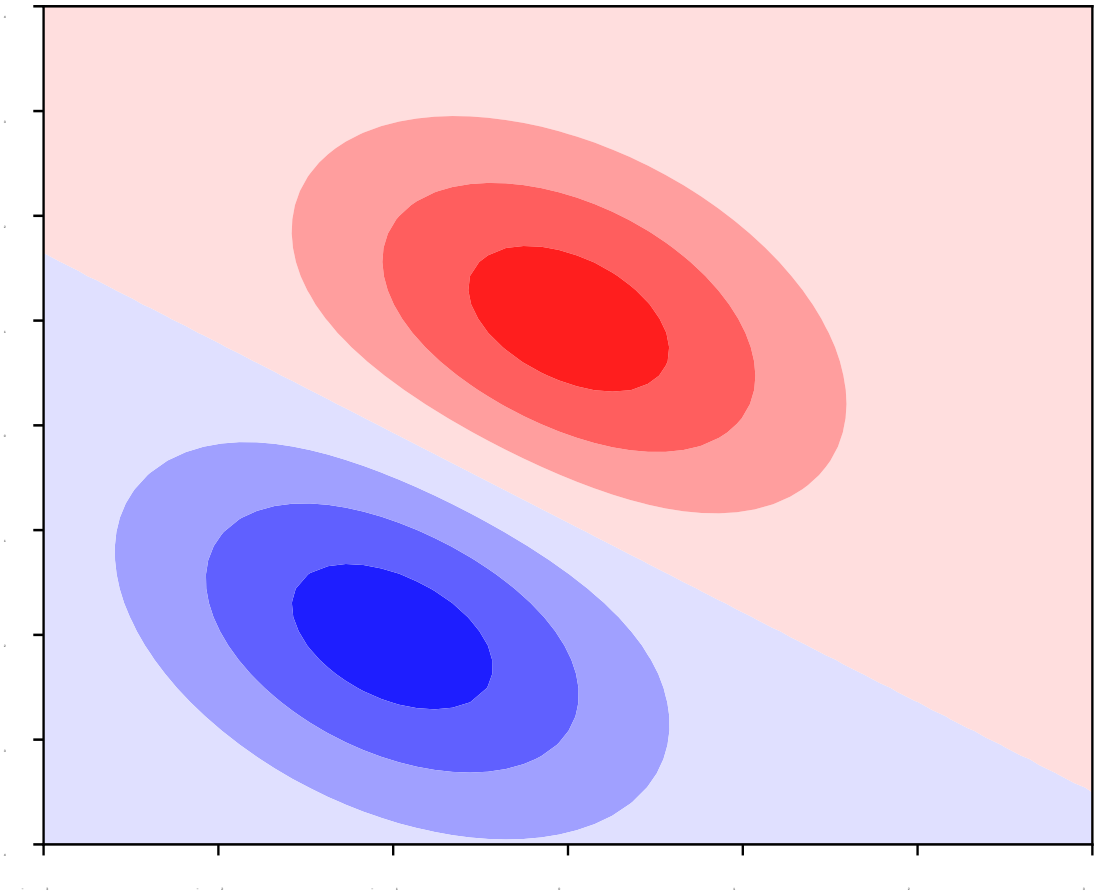
$$P(Y_{\text{Binom}}(2n+1, p = 0.4) \geq n+1) = \sum_{i=n+1}^{2n+1} \binom{2n+1}{i} (1-p)^{2n+1-i} p^i$$

- it gets better when using more data to decide
- out of class: Convergence for $n \to \infty$?

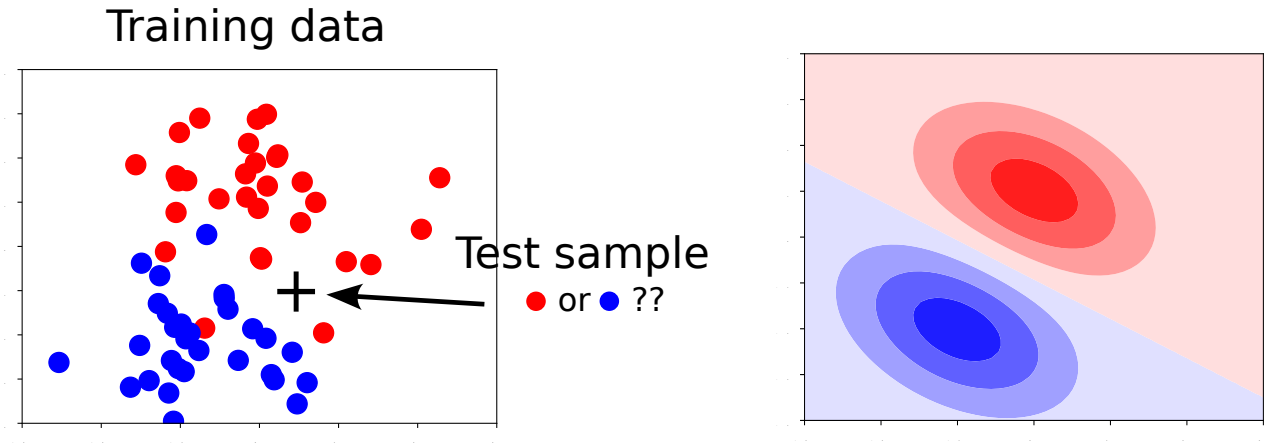
Observation: may choose suboptimal mappings, when using finite data to decide.



Would be easy, if we would know P_{test} ...



Why does Overfitting occur??



- because we use finite and noisy data to make decisions which would need to know expectations of loss functions $E_{(X,Y)\sim P_{test}}[l(f(X), Y))]$ under P_{test}
  (in this example the expected zero-one classification loss)

Search for model parameters is done on a training dataset D_n, and done by minimizing a loss $\hat{L}(f, D_n)$ computed on the training dataset.

$$(w^*, b^*) = \text{argmin}_{(w,b)} \hat{L}(f, D_n) = \text{argmin}_{(w,b)} \frac{1}{n} \sum_{(x_i,y_i)\in D_n} \ell(f(x_i), y_i)$$

$$= \text{argmin}_{(w,b)} \frac{1}{n} \sum_{(x_i,y_i)\in D_n} (f(x_i) - y_i)^2$$

Performance is measured on a test set of new unseen data
Overfitting can occur as a result of the training:

overfitting

Overfitting is when the loss on new unseen data points T_n (validation data, test data) is often (with high probability) much higher than the loss of the model on the training data: $\hat{L}(f, T_n) \gg \hat{L}(f, D_n)$

Lets consider another example:
True model for a regression problem:

$$x \in \mathbb{R}^1$$
$$y \in \mathbb{R}^1$$
$$p(x) = Normal(0, 1)$$
$$p(y|x) = Normal(2x - 1, 1)$$
$$P_{test}(x, y) = p(x)p(y|x)$$

- Question before we dig deeper: Why the labels have noise $p(y|x) = Normal(2x - 1, 1)$? Why we cannot know the label y exactly as a function of x?
- Modeling unaccounted additional effects on the outcome of y which cannot be known from only the input x!!
- Example: x current age. y time to death

Lets consider another example:
True model for a regression problem:

$$x \in \mathbb{R}^1$$
$$y \in \mathbb{R}^1$$
$$p(x) = Normal(0, 1)$$
$$p(y|x) = Normal(2x - 1, 1)$$
$$P_{test}(x, y) = p(x)p(y|x)$$

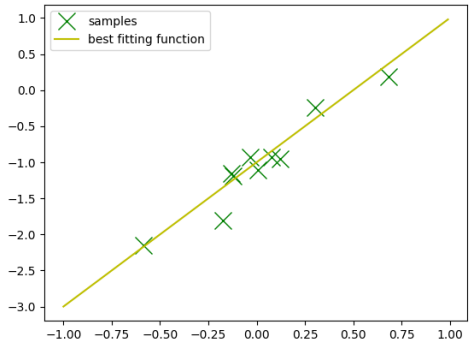
- Question before we dig deeper: Why the labels have noise $p(y|x) = Normal(2x - 1, 1)$? Why we cannot know the label y exactly as a function of x?
- Modeling unaccounted additional effects on the outcome of y which cannot be known from only the input x!!
- Example: x current age. y time to death

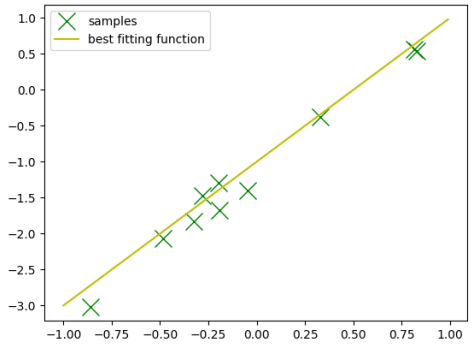
Lets consider one example:
True model for a regression problem:

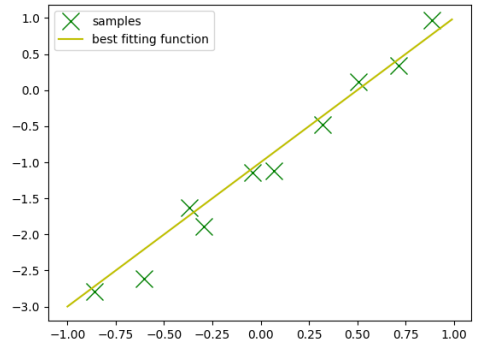
$$x \in \mathbb{R}^1$$
$$y \in \mathbb{R}^1$$
$$p(x) = Normal(0, 1)$$
$$p(y|x) = Normal(2x - 1, 1)$$
$$P_{test}(x, y) = p(x)p(y|x)$$

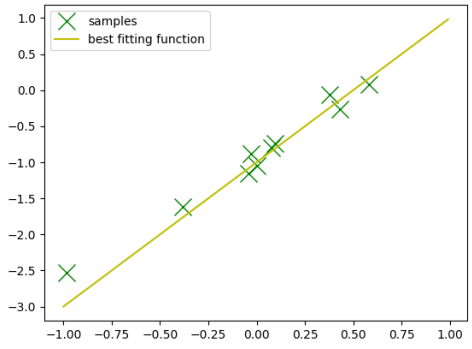
Why does overfitting happen at all?

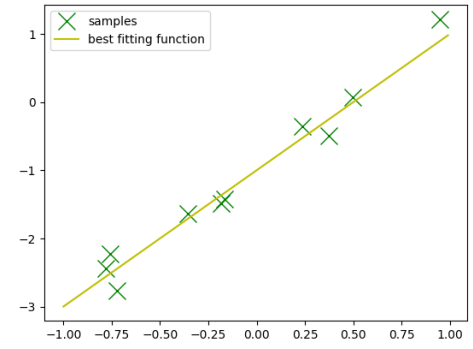
Draws of $n = 10$ samples look like that (best fitting function in the graphic is meant in the sense of expected loss under $P_{test}(x, y)$. Not in the sense of best linear fit to the particular realization of 10 points):

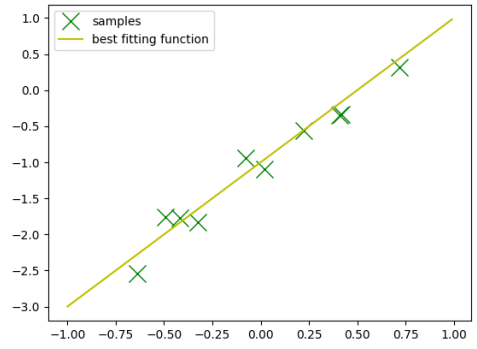






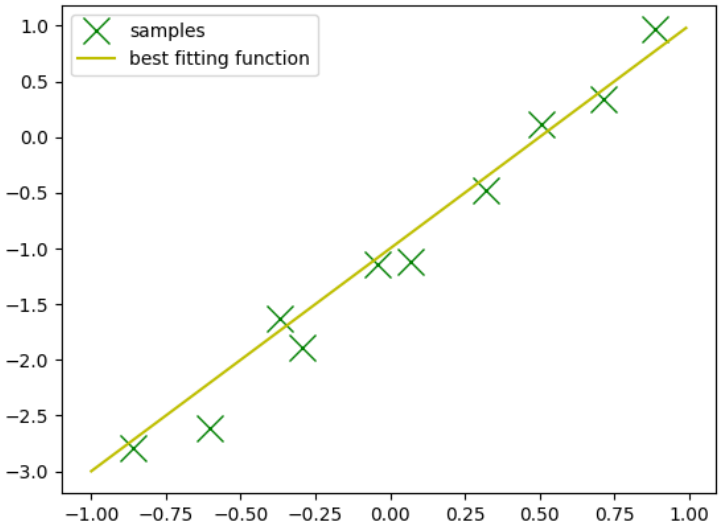






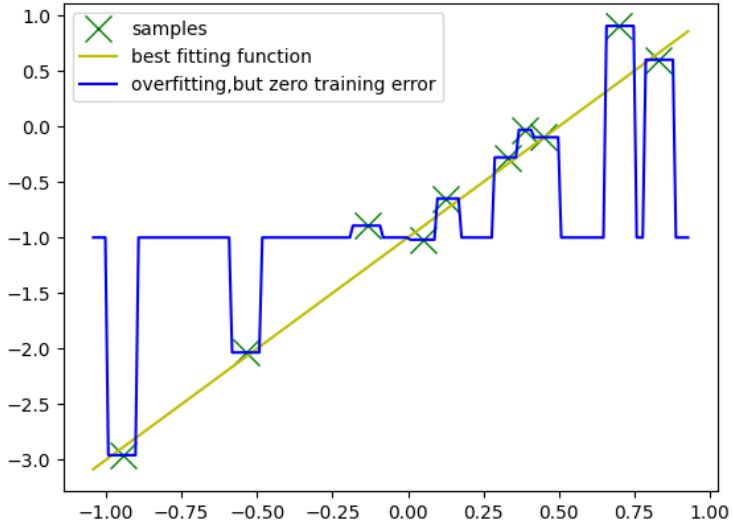
Why does overfitting happen at all?

Draws of $n = 10$ samples look like that:



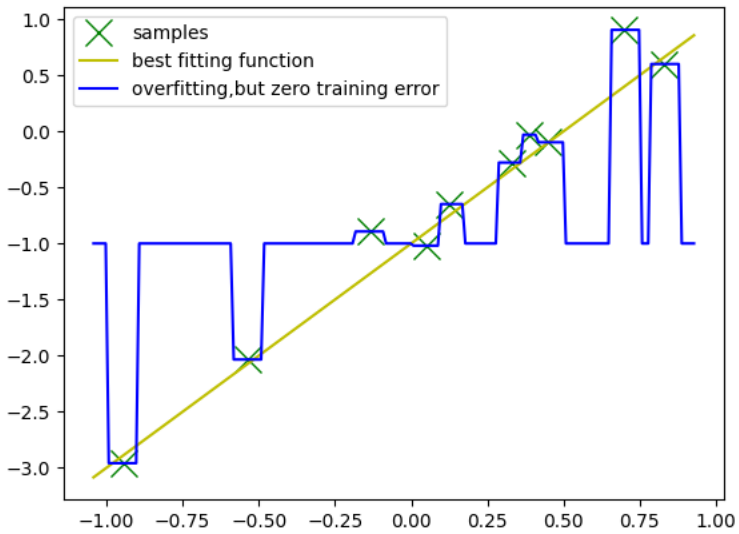
The yellow line would be a good learned model.

You can learn a model with zero training error like that:



This model will have high error under many draws from $P(x, y)$!

Why overfitting happens?

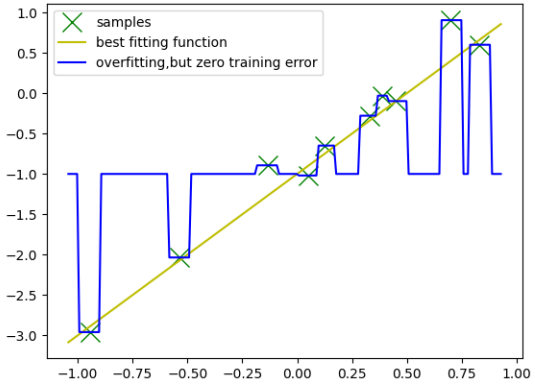


- Overfitting happens because you extract knowledge from a finite training dataset and you make assumptions about expected values.
- Additional difficulty: Often the expected values are about data not present in the training dataset when making a prediction on test sets. Then you have to make additional assumptions in your prediction model.
- Some of the insights from the training set might be specific to this finite dataset, and not be present in other draws of new, unseen data.

If one has learned a linear model on training data, can one have a guarantee that the model generalizes well without looking at test data?

Why overfitting happens?

If one has learned a linear model on training data, can one have a guarantee that the model generalizes well without looking at test data?



The blue line could be your true model P_{test}!! If you dont know P_{test}, then generalization can be only evaluated empirically on test data disjoint from training and validation data.

- this is why – whatever one has learned from training data, one must validate it in ML on disjoint validation data and test data

Result:

- Overfitting happens because you extract knowledge from a finite training dataset and you make assumptions about expected values.
- this is why – whatever one has learned from training data, one must validate it in ML on disjoint validation data and test data
- training data: fit parameters of your model
- **disjoint from both other sets** validation data: fit hyperparameters of your model
- **disjoint from both other sets** test data: the final test of the fitted model
- Overfitting always happens, question is only: to what extent?

- Training, validation and test data MUST be disjoint. Otherwise one cannot measure generalization on new, unseen data points.
- If they are not disjoint, we have only a guarantee to perform well on (partially) seen points instead – usually not of interest.

Next: show that a mapping with low expected loss is a good choice.

Why low expected loss is desirable?

- Suppose we have a trained model which will perform well in the following sense: its expected loss under the probability distribution P_{test} is low:

$$E_{(x,y)\sim P_{test}}[\ell(f(x), y)] \rightarrow \text{ low}$$

- Why do we have a chance to succeed in achieving low errors

$$\hat{L}(f, T_n) = \frac{1}{n} \sum_{(x_i, y_i) \in T_n} \ell(f(x_i), y_i)$$

on most test sets T_n made of data samples drawn from P_{test}?

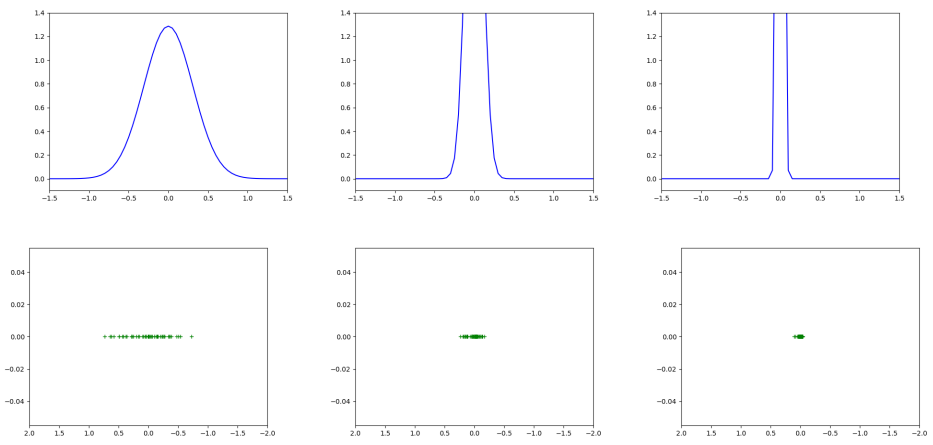
Why do we have a chance to succeed in achieving low errors for most test sets?

If a model has low expected loss, why do we have a chance to succeed in achieving low errors for most test sets?

Recap on central limit theorem

If you have a population with mean μ and standard deviation σ and you draw n random samples from the population - statistically independently, then the distribution of the sample means $\hat{\mu}$ will be **approximately** normally distributed **with mean μ and standard deviation $\frac{\sigma}{\sqrt{n}}$**

n small, n medium, n large



If a model has low expected loss, why do we have a chance to succeed in achieving low errors for most test sets?

Recap on central limit theorem

If you have a population with mean μ and standard deviation σ and you draw n random samples from the population - statistically independently, then the distribution of the sample means $\hat{\mu}$ will be **approximately** normally distributed **with mean μ and standard deviation $\frac{\sigma}{\sqrt{n}}$**

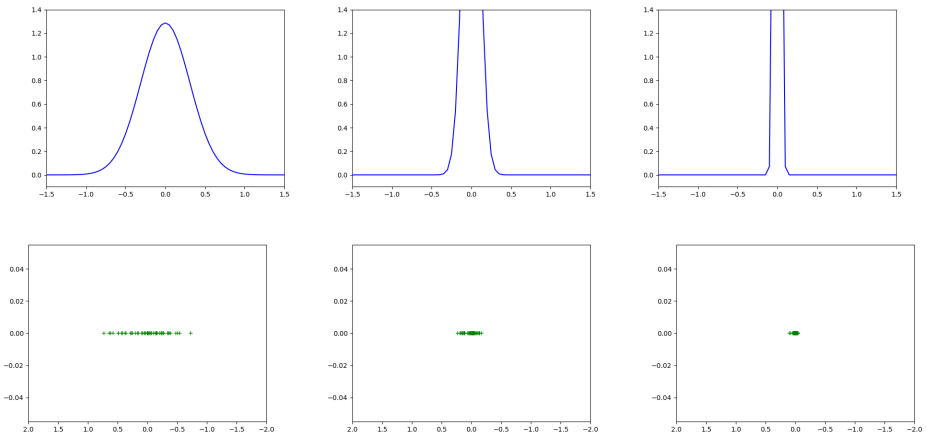
What does that imply when we compute a loss from a drawn testset T_n?

$$\hat{L}(f, T_n) = \frac{1}{n} \sum_{(x_i, y_i) \in T_n} \ell(f(x_i), y_i)$$

The loss $\hat{L}(f, T_n)$ is a sample mean $\hat{\mu}$ over n samples $z_i = \ell(f(x_i), y_i)$ drawn from the population for the random variable $Z = \ell(f(X), Y)$!

Why do we have a chance to succeed in achieving low errors for most test sets?

We draw a test set $T_n \sim P_{test}$ with $n = 900$ samples. We compute a test set loss $\hat{L}(f, T_n)$ / sample mean $\hat{\mu}$.



- The answer: If we repeat this experiment = draw 100 times a test set with $n = 900$ samples, and compute 100 times a test set loss, then many of the 100 test set losses will be close to μ, as each of them is approximately from a distribution with variance $\frac{\sigma}{\sqrt{n}} = \frac{\sigma}{\sqrt{900}}$ around the mean.
- number of "good" testsets ($\hat{\mu}$ close to μ) increases as the sample size $n \to \infty$ of the single testset increases

What is the probability for a test set loss to be outside $[\mu - 3\frac{\sigma}{\sqrt{n}}, \mu + 3\frac{\sigma}{\sqrt{n}}]$?
What about to be outside $[\mu - 6\frac{\sigma}{\sqrt{n}}, \mu + 6\frac{\sigma}{\sqrt{n}}]$?
https://en.wikipedia.org/wiki/68%E2%80%9395%E2%80%9399.7_rule
if n large enough and i.i.d drawn data. Note $\sigma_n = \frac{\sigma}{\sqrt{n}}$, so the probability to be outside gets smaller as our test set size n gets larger!
Note the rule is for an exact normal distribution. The distribution is approximately normal. (offtopic: concentration inequalities for probabilistic upper bounds)

Why do we have a chance to succeed in achieving average losses being close to the expectation μ under $P_{train/test}$ for most test sets?

As a consequence from the CLT

$$E_{(x,y)\sim P_{test}}[\ell(f(X), Y)] \approx \frac{1}{n} \sum_{(x_i,y_i)\in T_n} \ell(f_w(x_i), y_i)$$

with a certain probability, provided that

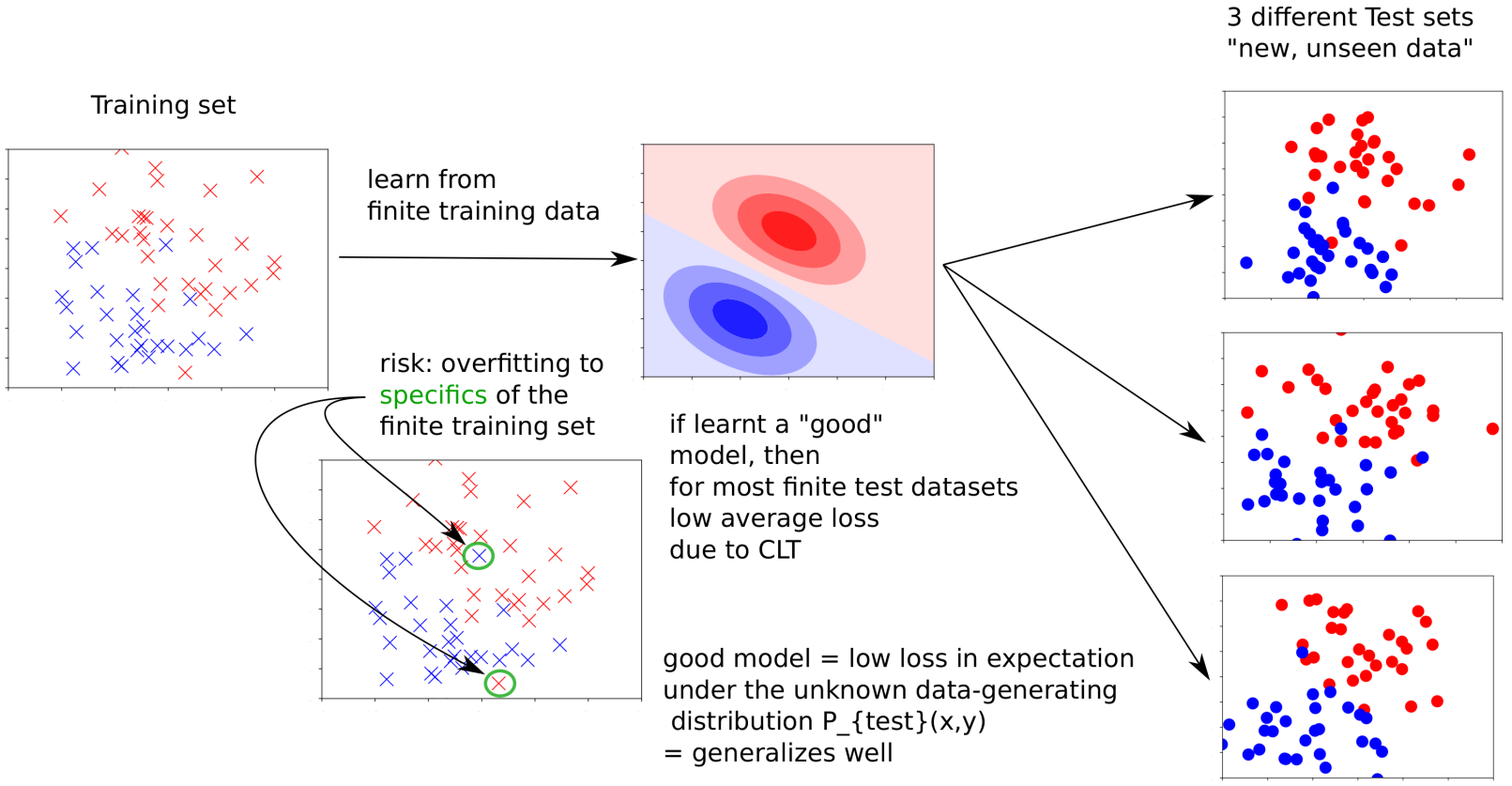
- 1. the samples (x_i, y_i) are drawn from P_{test}
- 2. the samples (x_i, y_i) are drawn independently

in the sense that $\frac{1}{n}\sum_{(x_i,y_i)\in T_n} \ell(f_w(x_i), y_i)$ will be a sample mean, drawn from an approximately normal distribution with variance $\frac{\sigma}{\sqrt{n}}$ around its mean $E_{(x,y)\sim P_{test}}[\ell(f(X), Y)]$

- Question: Why the goal of ML is not??? to have low loss on one test dataset $T_{50} = \{(x_i, y_i), i = 1, \ldots, 50\}$?

That is important to understand, because often errors/performance are measured only on a single test dataset ...

The core idea of ML



Outline

1. Motivation
2. The four components to define a machine learning problem
3. Ordinary Least Squares as an intro example
4. Overfitting, Generalization and CLT
5. What does a linear mapping represent?
6. Using the Linear model for classification
7. A first method to solve linear regression

Goal: understand what the mapping $f(\cdot)$ does.
Approach: characterizing the set of points x with a constant output $f(x) = c$.

Thinking task

What is the set of points $x = (x_1, x_2) \in \mathbb{R}^2$ such that

$$3x_1 - 2x_2 + 3 = 0 \ ?$$

What is the set of points $x = (x_1, x_2, x_3) \in \mathbb{R}^3$ such that

$$x_1 - x_2 - 2x_3 + 2 = 0 \ ?$$

A. What is the set of points x: $f_{w,b}(x) = 0$?
B. What is the set of points x the prediction is a constant c, that is $f_{w,b}(x) = c$?

$$f_{w,b}(x) = 0 \Leftrightarrow x \cdot w = -b$$

To understand how the bias b influences the zero set, lets consider three cases:

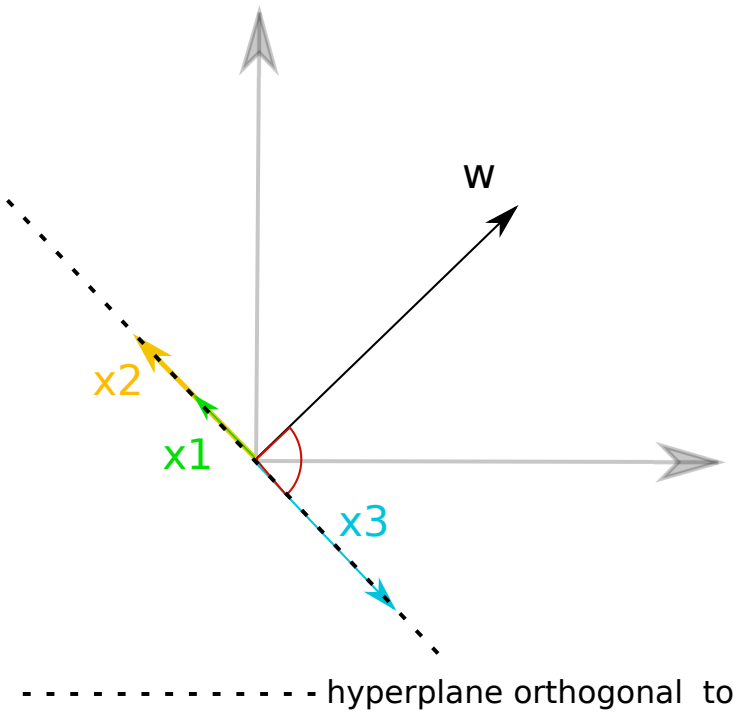
- $b = 0$
- $b > 0$
- $b < 0$

The set of points x: $f_{w,b}(x) = 0$ – The case $b = 0$

- We know that for $b = 0$: $f_{w,0}(x) = w \cdot x = 0$ holds for the zero vector $x = 0$.
- The set of points x such that the inner product

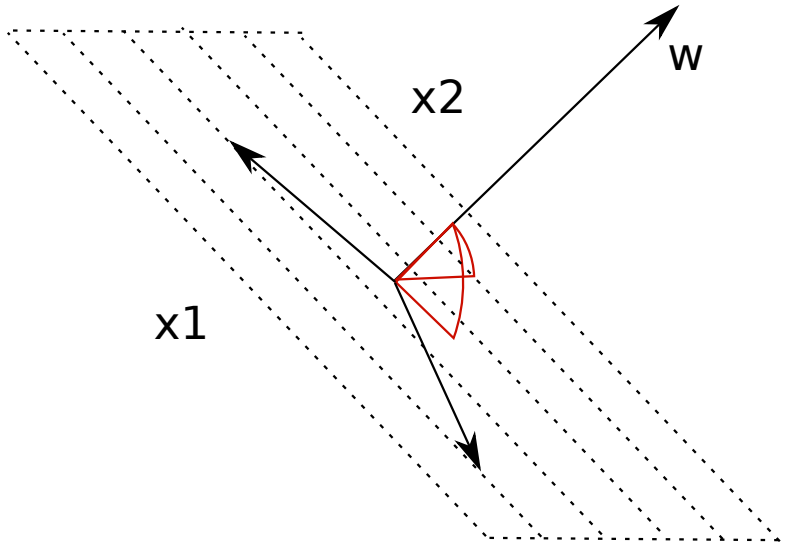
$$x \cdot w = 0$$

is in 2 dims a one-dimensional line, which goes through the origin $(x_1, x_2) = (0, 0)$, and which is orthogonal to w.



x_1,x_2,x_3 are all orthogonal to the vector w

The analogy also holds for 3 or more dimensions. So for 3 dims it is a two-dimensional plane, which goes through the origin $(x_1, x_2, x_3) = (0, 0, 0)$.



hyperplane
through origin
orthogonal to w
vectors x1 and x2 lie inside
this hyperplane

For n dimensions the plane of orthogonal vectors has $n - 1$ dimensions (+ goes through the origin), but is still a hyperplane

Recap hyperplane of dimension $n - 1$

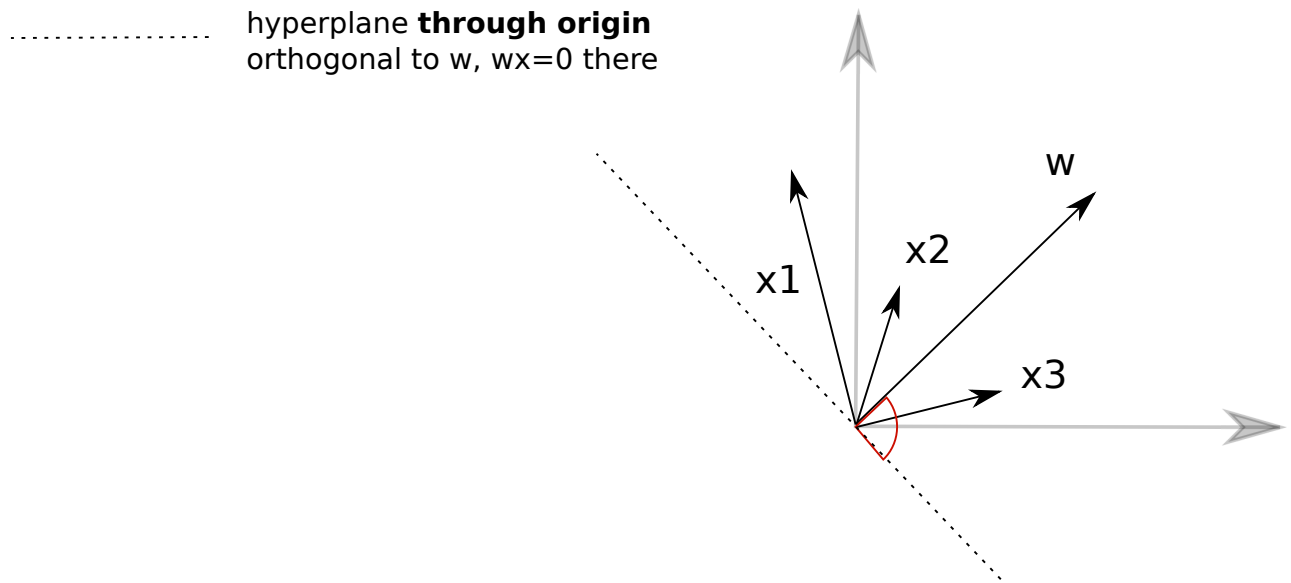
- P is a hyperplane (linear space) if it holds: $x_1 \in P, x_2 \in P \Rightarrow a_1 x_1 + a_2 x_2 \in P$ (space closed under linear operations)
- can find $n - 1$ basis vectors such that each point of P can be represented as a linear combination of the basis vectors)

$$\exists\, v_1, \ldots, v_{n-1} \text{ such that}$$

$$\forall x \in P \;\exists\, a_1, \ldots, a_{n-1} \text{ such that } x = \sum_{i=1}^{n-1} a_i v_i = \boldsymbol{a} \cdot \boldsymbol{V}$$

$$b < 0,\ w \cdot x + b = 0 \Rightarrow w \cdot x = -b > 0$$

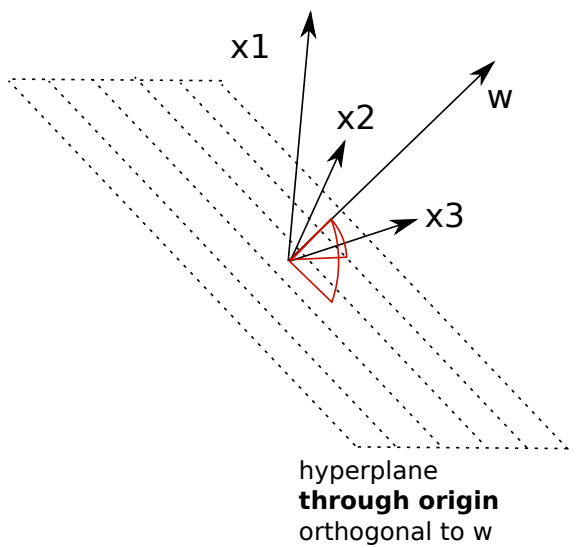
We know: $w \cdot x > 0$ for all points x that are on that side of the **hyperplane through the origin**, in which w points to.



In the above figure $\mathbf{x_1}$,$\mathbf{x_2}$,$\mathbf{x_3}$ all have $w \cdot x_i > 0$ because relative to the hyperplane orthogonal to w which goes through the origin, they all point towards the direction of w

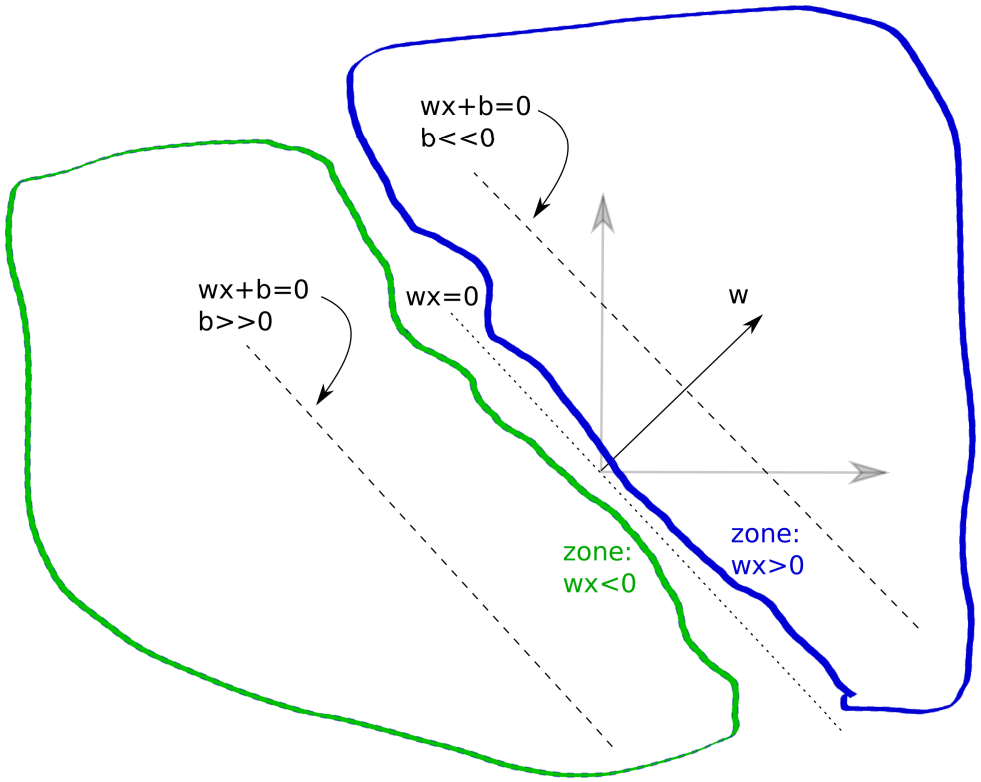
$$b < 0,\ w \cdot x + b = 0 \Rightarrow w \cdot x = -b > 0$$

We know: $w \cdot x > 0$ for all points x that are on that side of the **hyperplane through the origin**, in which w points to.
The same also holds for 3 or more dimensions. All the vectors below solve $wx + b = 0$ for some bias $b < 0$!



In above figure x_1,x_2,x_3 all have $w \cdot x_i > 0$

The bias b shifts the hyperplane corresponding to $w \cdot x + b = 0$ parallel/anti-parallel to the direction of w.



the hyperplane is parallel to the hyperplane orthogonal to w which goes through the origin

The set of points x such that

$$\{x : \ wx + b = 0\}$$

for $b < 0$ is a hyperplane which is parallel to the hyperplane $\{x : x \cdot w = 0\}$ orthogonal to w going through the origin, and which is shifted towards the direction of w

The set of points x such that

$$\{x : \ wx + b = 0\}$$

for $b > 0$ is a hyperplane which is parallel to the hyperplane $\{x : x \cdot w = 0\}$ orthogonal to w going through the origin, and which is shifted opposite to the direction of w

hyperplane dependency on bias b

- $\{x : \ w \cdot x = 0\}$ is a hyperplane orthogonal to w.
- Negative $b < 0$ shift the hyperplane $\{x : wx + b = 0\}$ towards the direction of w,
- positive $b > 0$ shift the hyperplane $\{x : wx + b = 0\}$ against the direction of w.
- Large absolute values $|b|$ shift it far away.

hyperplane explicit

The linear mapping $f(x) = w \cdot x + b$ has a zero set which is the plane of points

$$\{x : x = u + -b\frac{w}{\|w\|^2}, u \text{ such that } w \cdot u = 0\}$$

In this representation:

- u such that $w \cdot u = 0$ is the hyperplane of vectors u orthogonal to w.
- the vector $-b\frac{w}{\|w\|^2}$ shifts the hyperplane antiparallel to direction of w.

That holds because

$$w \cdot x + b = w \cdot \left(u + -b\frac{w}{\|w\|^2}\right) + b = 0$$

By subtracting its component parallel to w!

Be x any vector. Subtract its component parallel to w.
Important: use length-normalized w: $\frac{w}{\|w\|}$
$\left(x \cdot \frac{w}{\|w\|}\right)$ is the component of x in direction of $\frac{w}{\|w\|}$

$$u = x - \left(x \cdot \frac{w}{\|w\|}\right) \frac{w}{\|w\|} = x - (x \cdot w) \frac{1}{\|w\|^2} w$$

$$\Rightarrow u \cdot w = 0$$

cf. Gram-Schmid Orthonormalization to find an orthonormal basis of such vectors.

What is the set of points x where the prediction is a constant, that is $g_{w,b}(x) = c$?

Answered by reducing it to a zero set:

$$g_{w,b}(x) = wx + b = c$$
$$wx + (b - c) = 0$$

The set of points x such that $g_{w,b}(x) = c$ is just the set $x : \ g_{w,b-c}(x) = 0$.

thinking task

What is the set of points $x = (x_1, x_2) \in \mathbb{R}^2$ such that

$$3x_1 - 2x_2 + 3 = 0 \ ?$$

What is the set of points $x = (x_1, x_2, x_3) \in \mathbb{R}^3$ such that

$$x_1 - x_2 - 2x_3 + 2 = 0 \ ?$$

Outline

1. Motivation
2. The four components to define a machine learning problem
3. Ordinary Least Squares as an intro example
4. Overfitting, Generalization and CLT
5. What does a linear mapping represent?
6. Using the Linear model for classification
7. A first method to solve linear regression

- goal to predict a class of an input $x \in \mathcal{X}$



Credit: Pixabay user Loggawiggler

Which sort ? Fly agaric ? Death cap ? Chestnut bolete?

- output space $\mathcal{Y}$ is discrete and finite.
- binary classification: $\mathcal{Y} = \{0, 1\}$ or $\{-1, 1\}$
- C-class classification $\mathcal{Y} = \{0, 1, 2, \ldots, C-1\}$

Now: binary prediction with predicted label decided by
$f(x) > 0\,?\,-1 : +1$
First loss function for a pair (x, y):

zero-one-loss

$$y \in \{-1, +1\} \Rightarrow \ell(f(x), y) = \mathbb{1}[\mathrm{sgn}(f_{w,b}(x)) \neq y]$$

Problem: sign is unsuitable for gradient optimization.
Note here the possibility to rewrite the condition for $y \in \{-1, +1\}$: $\mathbb{1}[sign(f(x)) \neq y]$ as $\mathbb{1}[f(x)y < 0]$ without any sign – that is: we have an error if the prediction function $f(x)$ has opposite sign of the ground truth label y.

$$f(x)y = \begin{cases} > 0 & \text{if } f(x) > 0, y > 0 \text{ or } f(x) < 0, y < 0 \text{ no error} \\ < 0 & \text{if } f(x) > 0, y < 0 \text{ or } f(x) < 0, y > 0 \text{ error} \end{cases}$$

hinge-loss

$y \in \{1, +1\},\ \ell(f(x), y) = \max(0, 1 - f_{w,b}(x)y)$

The hinge-loss is an upper bound on the zero-one-loss (insight: fix $y = +1$ or $y = -1$. Plot $\mathbb{1}[zy < 0]$ and $\max(0, 1 - zy)$). Consequence from upper bound property: If the hinge loss is low, the zero-one loss must also become low. Therefore ok to minimize an average over the hinge loss over training samples

$$L(w, b) = \frac{1}{n} \sum_{(x_i, y_i) \in D} \max(0, 1 - y_i f_{w,b}(x_i))$$

$$\text{solve } (w^*, b^*) = \operatorname{argmin}_{w,b} L(w, b)$$

Then you predict using f_{w^*,b^*}.

now add again a quadratic penalty on the weights when learning parameters over a training set

$$\operatorname{argmin}_{(w,b)} \frac{1}{n} \sum_{(x_i,y_i)\in D_n} \max(0, 1 - g_{w,b}(x_i)y_i) + \lambda \|w\|^2$$

SVM:

- averaged hinge loss $\max(0, 1 - f(x_i)y_i)$
- with a linear/affine model $f(x_i) = w \cdot x_i + b$
- with quadratic penalty $\lambda\|w\|^2$ on the weights.

A different way to arrive at an SVM (remember $\frac{1}{\|w\|}$ is the margin to be maximized)

Generalize well means again the same as for regression:

generalization

Generalize well means in short form:

- we find parameters (w^*, b^*) defining a mapping f which for most draws of test datasets T_n produces predictions with low errors on them:

$\hat{L}(f, T_n) = \frac{1}{n} \sum_{(x_i, y_i) \in T_n} \ell(f(x_i), y_i) \rightarrow \textit{low}$

Take away: generalization

Generalization is the same idea for many different setups: learn parameters on training data, so that the loss on newly drawn test datasets will be low on average – for most draws of such datasets from a data source.

See also that the linear model is suitable for both classification and regression. What makes the difference whether we have a classification or a regression model?

Outline

1. Motivation
2. The four components to define a machine learning problem
3. Ordinary Least Squares as an intro example
4. Overfitting, Generalization and CLT
5. What does a linear mapping represent?
6. Using the Linear model for classification
7. A first method to solve linear regression

Next: Recap on one method of how to find parameters in linear regression

Assumption here: no bias. Parameters are w.

Goal: to find parameters which minimize the loss on this dataset:

$$(w^*) = \text{argmin}_w \sum_{i=1}^{n} L(f(x_i), y_i) = \text{argmin}_w \sum_{i=1}^{n} (x_i \cdot w - y_i)^2$$

for a given dataset $D_n = \{(x_i, y_i)\}$. Then $f_{w^*}(x) = x \cdot w^*$ is the selected mapping. Rare case: Can be solved explicitly for w.

- consider the loss L as function of w
- compute $\nabla L(w)$ - the gradient of the loss with respect to w
- solve $\nabla L(w) = 0$ for w.
- Solution can be maximum, minimum or saddle point. Verify that the Hessian in the solution point is positive definite. That is: the function has positive curvature in every direction. implies: must be a minimum.

Write in matrix form:

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1^{(1)}, \ldots, x_1^{(D)} \\ x_2^{(1)}, \ldots, x_2^{(D)} \\ \vdots \\ x_n^{(1)}, \ldots, x_n^{(D)} \end{pmatrix} \in \mathbb{R}^{n \times D}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

then:

$$L(w) = \sum_{i=1}^{n} (x_i \cdot w - y_i)^2 = (X \cdot w - Y)^T \cdot (X \cdot w - Y)$$

Solve the minimization problem by computing the gradient for w and setting it to zero.

$$\begin{aligned} D_w((X \cdot w - Y)^T \cdot (X \cdot w - Y)) &= 2X^T \cdot (X \cdot w - Y) \\ 2X^T \cdot (X \cdot w - Y) &= 0 \\ \Rightarrow (X^T \cdot X) \cdot w &= X^T \cdot Y \\ w &= (X^T \cdot X)^{-1} X^T \cdot Y \end{aligned}$$

if the matrix inverse $(X^T \cdot X)^{-1}$ exists.

explicit solution to linear regression without bias

Let X be the training data matrix.

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1^{(1)}, \ldots, x_1^{(D)} \\ x_2^{(1)}, \ldots, x_2^{(D)} \\ \vdots \\ x_n^{(1)}, \ldots, x_n^{(D)} \end{pmatrix} \in \mathbb{R}^{n \times D}, \quad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^{n \times 1}$$

The model is $f(x) = w \cdot x$. Then a solution is given as

$$w^* = (X^T \cdot X)^{-1} X^T \cdot Y$$

if the matrix inverse $(X^T \cdot X)^{-1}$ exists.

Prediction is done using: $f_{w^*}(x) = w^* \cdot x$

What kind of extremum is the solution? Compute the Hessian

$$D_w((X \cdot w - Y)^T \cdot (X \cdot w - Y)) = 2X^T \cdot (X \cdot w - Y)$$
$$D_w D_w((X \cdot w - Y)^T \cdot (X \cdot w - Y)) = 2X^T \cdot X$$

a matrix $A^T A$ is always non-negative definite. Thus the solution is either a minimum or a saddle point, but not maximum.

- How to extend this solution to the case with a bias?

$$x = \left(x^{(1)}, \ldots, x^{(D)}\right) \rightarrow \hat{x} = \left(x^{(1)}, \ldots, x^{(D)}, 1\right)$$

Then the parameter w also gets an additional dimension

$$\hat{w} = \left(w^{(1)}, \ldots, w^{(D)}, w^{(D+1)}\right)$$

then:

$$\hat{x} \cdot \hat{w} = w \cdot x + w^{(D+1)} = w \cdot x + b$$

$w^{(D+1)}$ acts as bias.

From Linear to Ridge regression

Overfitting: low training error, high test error. Reason: during learning one picks up too much of the noise in the training data, and learns weights that listen to noise signals.

One way to deal with it: avoiding weights w getting too large: add a penalty on the euclidean length of w

$$\text{argmin}_w \sum_{i=1}^{n} (x_i \cdot w - y_i)^2 + \lambda \|w\|^2$$

$$\sum_{i=1}^{n}(x_i \cdot w - y_i)^2 + \lambda \|w\|^2$$
$$= (X \cdot w - Y)^T \cdot (X \cdot w - Y) + \lambda w^T \cdot w$$
$$D_w((X \cdot w - Y)^T \cdot (X \cdot w - Y) + \lambda w^T \cdot w)$$
$$= 2X^T \cdot (X \cdot w - Y) + 2\lambda w$$
$$2X^T \cdot (X \cdot w - Y) + 2\lambda w = 0$$
$$X^T \cdot X \cdot w + \lambda I \cdot w = X^T \cdot Y$$
$$(X^T \cdot X + \lambda I) \cdot w = X^T \cdot Y$$
$$\Rightarrow w = (X^T \cdot X + \lambda I)^{-1} X^T \cdot Y$$

Ridge regression

Ridge regression is Linear regression with an added squared-ℓ_2 penalty term on weights

$$\lambda \|w\|^2$$

λ is a hyperparameter in this approach. The solution changes to

$$w = (X^T \cdot X + \lambda I)^{-1} X^T \cdot Y$$

In practice, one needs to find a good value for the hyperparameter λ on a validation set, before measuring the performance on the test set. The effect of this regularization will be discussed later.

Advantages of ridge regression over least squares:

- for any $\lambda > 0$ a solution always exists: $(X^T \cdot X + \lambda I)$ is always invertible because it is positive definite
- the Hessian of L is positive definite, thus one finds always a minimum and not a saddle point (or maximum)
- when in least squares without ridge regression a solution does not exist?

Questions?!