Machine Learning

2. Linear Regression

Yannick Le Cacheux

CentraleSupélec - Université Paris Saclay

September 2024

Outline

A regression problem

- Recall from last class that

Definition

Regression is a supervised learning task where the goal is to predict a *continuous* numerical output value based on input features.

- This is as opposed to *e.g.* classification, where the goal is to prediction a class among finite possibilities.

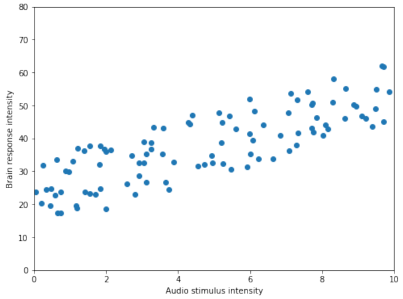
Problem statement

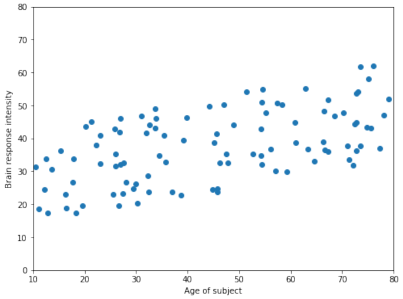
- Let's say we have some input variables $\mathbf{x}$, *e.g.* an auditory stimulus x_1 and the age of a subject x_2, and we want to predict the intensity of the brain response y.
- We have a dataset $\mathcal{D}$ of past observations $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

	$x_{n,1}$	$x_{n,2}$	y_n
	Stimulus intensity	Subject's age	Brain response
$\mathbf{x}_1$	3.7	12	21
$\mathbf{x}_2$	9.5	54	44
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{x}_n$	7.3	32	36
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{x}_N$	6.0	46	32

Looking at the data

- Let's look at the response y with respect to the inputs x_1 and x_2





- It looks like the response is roughly proportional to the input variable(s), with an offset.
- $y \approx a \cdot x + b$ for all inputs x

Hypothesis

- Generalization: we want to predict an output value $\hat{y} \in \mathbb{R}$ given a D-dimensional input $\mathbf{x} = (x_1, \dots, x_D)^\top \in \mathbb{R}^D$
  - For instance, predict the brain response w.r.t. the intensity of an audio stimulus and the age of the patient
- We are looking for a function $f : \mathbb{R}^D \rightarrow \mathbb{R}$ that maps our input $\mathbf{x}$ to our output y

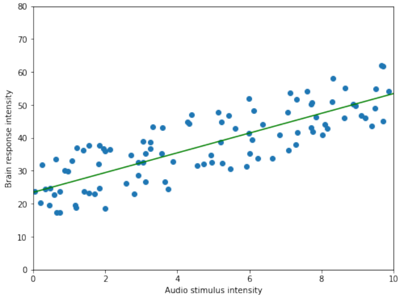
Assumption

f is linear with respect to $\mathbf{x}$

$$\begin{aligned} f_{\mathbf{w}}(\mathbf{x}) &= w_1 x_1 + w_2 x_2 + \cdots + w_D x_D + w_0 \\ &= \sum_{d=1}^{D} w_d x_d + w_0 \\ &= \mathbf{w}^\top \mathbf{x} + w_0 \end{aligned}$$

with $\mathbf{w} = (w_1, \dots, w_D) \in \mathbb{R}^D$

Examples of linear fit



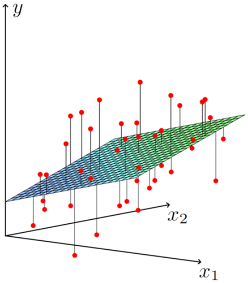
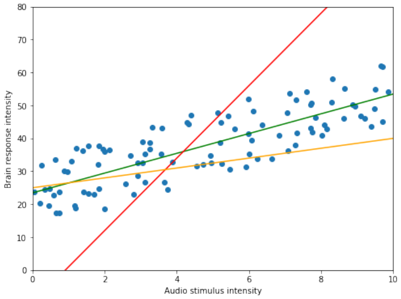
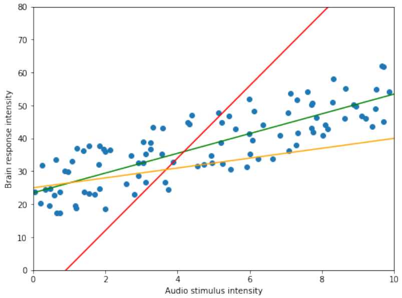


Figure: Left: linear fit of 1 target variable with respect to 1 input variable. Right: linear fit of 1 target variable with respect to 2 input variables.

How do we define a “good” fit?



How do we define a “good” fit?



- Green line looks like a good fit
- Orange line not so much
- Red line looks like a bad fit.

Intuition

When the line is a good fit, the prediction $\hat{y}_n = f(\mathbf{x}_n)$ is close to target y_n the for most points $\mathbf{x}_n$.

Cost function

- We can quantify this with a *cost function*, also called a *loss*, which we can minimize
- In regression, we often use least squares:

Training cost:

$$J_{\mathcal{D}}(\mathbf{w}) = \left(\frac{1}{N}\right) \sum_{n=1}^{N} (y_n - f(\mathbf{x}_n))^2$$

$$\text{with} \quad f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$

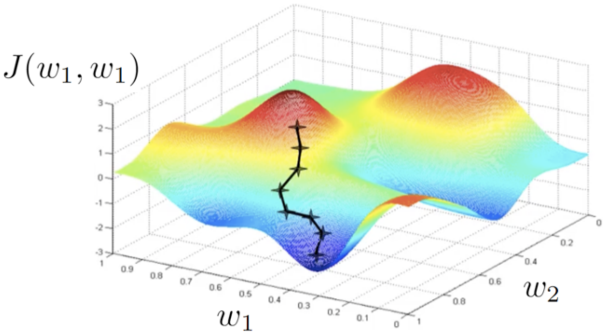
We want to find parameters $\mathbf{w}$ which minimize the training loss $J(\mathbf{w})$

How do we minimize $J(\mathbf{w})$?

- First method: *gradient descent* (iterative optimization)

$$\mathbf{w}_{(t+1)} = \mathbf{w}_{(t)} - \alpha \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$

- The parameters $\mathbf{w}_{(t)}$ are updated at each time step (t) with the gradient of the cost $\partial J(\mathbf{w})/\partial \mathbf{w}$ and a *learning rate* α.
- Stop when $\mathbf{w}_{(t)}$ is not changing anymore



Matrix calculus

- We can write our dataset $\mathcal{D}$ as a pair of matrix $\mathbf{X}$ and vector $\mathbf{y}$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_N^\top \end{pmatrix} = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,D} \\ x_{2,1} & x_{2,2} & \dots & x_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \dots & x_{N,D} \end{pmatrix} \in \mathbb{R}^{N \times D}$$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \in \mathbb{R}^N$$

- $x_{n,d}$ is the d^{th} feature of the n^{th} point, y_n is the label of the n^{th} point.

Remember these notations

We will use the notations $\mathbf{X}$ and $\mathbf{y}$ quite frequently in this course.

Matrix calculus

- Since the prediction $\hat{y}_n \in \mathbb{R}$ for one sample $\mathbf{x}_n \in \mathbb{R}^D$ can be written as

$$\hat{y}_n = f(\mathbf{x}_n) = \mathbf{w}^\top \mathbf{x}_n \quad {}^{1}$$

then predictions $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_N)^\top \in \mathbb{R}^D$ for *all* samples can be written

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$$

- And the cost function J can be rewritten as

$$J(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} (y_n - f(\mathbf{x}_n))^2$$
$$J(\mathbf{w}) = \frac{1}{N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

[1] We can discard the $+w_0$ for now, we will explain later why/how we can do that.

Matrix calculus

Objective

$$\underset{\mathbf{w}}{\text{minimize}}\ \frac{1}{N}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|_2^2 \quad \text{given } (\mathbf{X}, \mathbf{y})$$

From the appendix on matrix calculus, we have[2]

Lemma

$$\frac{\partial\|\mathbf{AWB}+\mathbf{C}\|_2^2}{\partial \mathbf{W}} = 2\mathbf{A}^\top(\mathbf{AWB}+\mathbf{C})\mathbf{B}^\top$$

Applying this to $J = \|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2$ and $\mathbf{w}$, we get a closed-form solution to the gradient of J with respect to parameters $\mathbf{w}$:

$$\boxed{\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{2}{N}\mathbf{X}^\top(\mathbf{X}\mathbf{w}-\mathbf{y})}$$

[2]See appendix

Alternative method: analytic solution

Gradient of J with respect to $\mathbf{w}$

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{2}{N}\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y})$$

At a local minima of J, the gradient will be 0, thus we have

$$\frac{2}{N}\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y}) = 0$$

Solving for $\mathbf{w}$, we finally get a closed form solution for the parameters:

$$\boxed{\mathbf{w} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}}$$

(To be more rigorous, we could show that J is convex, and therefore the local minima is the only global minima. *In general, this is not the case for non-convex functions.*)

Recap

Achievement unlocked

We have trained a first *parametric* machine learning algorithm!

- We collected "labeled" training examples $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^\top$, $\mathbf{y} = (y_1, \dots, y_N)^\top$
- We formulated a hypothesis regarding the link between the target and input features: $y = f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$
- We defined a cost function $J(\mathbf{w}) = \frac{1}{N} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$
- We chose parameters $\mathbf{w}$ which minimized the cost on the training dataset $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$
- Now, given a new unlabeled sample $\tilde{\mathbf{x}}$, we can estimate its label with $\hat{y} = f(\tilde{\mathbf{x}}) = \mathbf{w}^\top \tilde{\mathbf{x}}$
  - For instance, we now know that brain response $= 3 \cdot$ stimulus intensity $+ 0.3 \cdot$ age $+ 10$

Outline

Do we always want to minimize the training cost?

Do we always want to minimize the training cost?

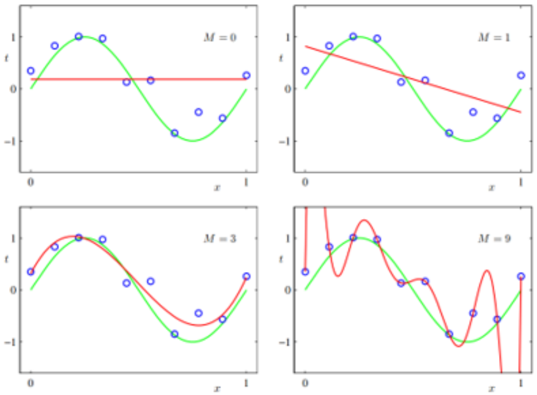


Figure: Fitting a set of points with a polynomial of degree M ranging from 0 to 9. Red line is the fitted polynome, green line is the ground truth distribution.

Do we always want to minimize the training cost?

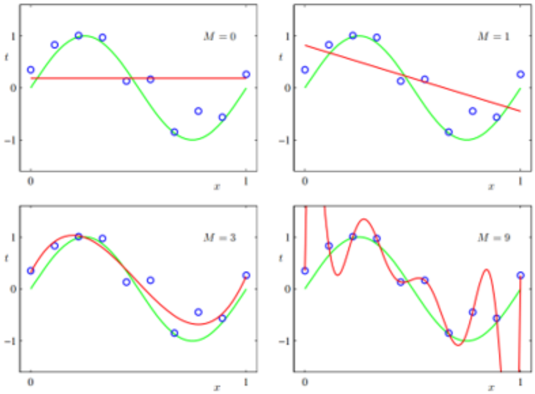


Figure: Fitting a set of points with a polynomial of degree M ranging from 0 to 9. Red line is the fitted polynome, green line is the ground truth distribution.

- In general: *no*.
- Polynomial[a] with degree $M = 9$ has the lowest training error, and yet it is clearly not the best model.
- Sometimes, a low error on the training set can result in a high error in new data.
- This is called $\boxed{\text{OVERFITTING}}$

[a] Be patient, we will see how to fit a polynomial in a few slides.

Is it bad?

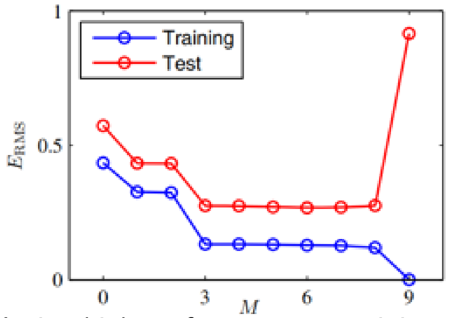


Figure: Root mean square error on training and test sets for polynomials of different degrees.

- Having high performance on training data only is useless.
- We are actually only interested in the performance on *new* data.
- It is thus critical to *always* split a dataset into a training set and a test set, to check that our model produces good predictions on *new* data.

How do we know when we overfit / underfit?

- We evaluate the performance of our predictions on data that was NOT used to train the model
- If there is high error on test data and low error on training data, we are probably overfitting

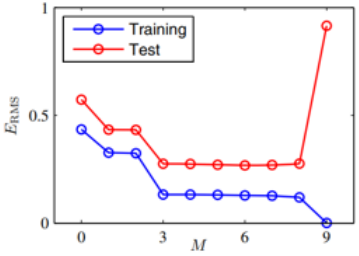
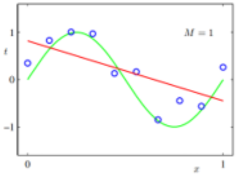
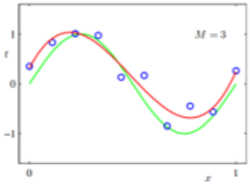


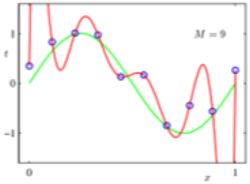
Figure: For degree $M = 9$, there is low error on training data but high error on test data: we are overfitting

14

This is REALLY a critical concept





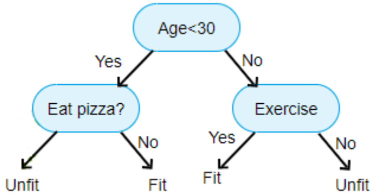


- Underfitting:
  - High error on training data
  - High error on test data
- Just right:
  - Low error on training data
  - Low error on test data

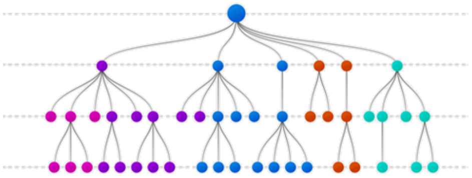
- Overfitting:
  - Low error on training data
  - High error on test data

Under/over-fitting and model complexity

- Under/over-fitting is related to the "complexity" of our model. Example with another model family, decision trees[3]:



(a) A simple decision tree



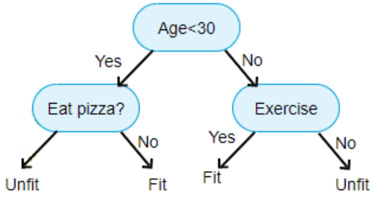
(b) A more complex decision tree

- Is very complex always bad?

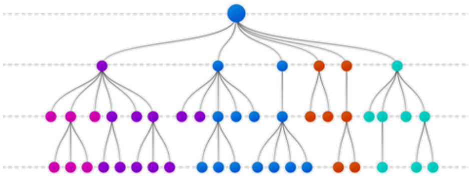
[3]We will see how to build decision trees in a few lectures.

Under/over-fitting and model complexity

- Under/over-fitting is related to the "complexity" of our model. Example with another model family, decision trees[3]:



(a) A simple decision tree



(b) A more complex decision tree

- Is very complex always bad? No, sometimes we want to model complex things.

[3]We will see how to build decision trees in a few lectures.

One last time

- This is probably the most important idea from this class. So let's insist a little bit.

Definition of overfitting

Overfitting occurs when a model learns the training data too well, capturing noise and outliers, which leads to poor generalization on unseen data.

- Remember, a model should perform well not just on training data but also on new, unseen data.
- Always validate your model to ensure it is not overfitting.

Outline

Can we overfit with a linear regression?

Can we overfit with a linear regression?

Short answer

Yes, linear regression models can overfit!

- Overfitting typically occurs when:
  - Number of features $D \approx N$ number of samples
  - Number of features $D > N$ number of samples
- In these cases, the model has too many "degrees of freedom"
- It can fit the training data perfectly, but fail to generalize

Example: polynomial features

While this may not be obvious for now, this is what happened earlier with our polynomial fitting. Cf. explanation in a few slides, on polynomial features.

- What can we do to prevent that?

Regularization

- We can use *regularization*: for instance, we can penalize weights with large values in $\mathbf{w}$
- To achieve this, we add a penalty to the training loss:

$$J(\mathbf{w}) = \frac{1}{N}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

- λ controls the strenght of the regularization: the higher the λ, the more we penalize large weights in $\mathbf{w}$
- λ is called a *hyper-parameter*

Definition: hyper-parameter

A hyper-parameter is a configuration external to the model that cannot be learned from the data directly. It is typically set before the learning process begins and controls aspects of the model's behavior or training process.

Ridge regression

- A linear regression regularizated with an $\ell 2$ norm is sometimes called a *ridge* regression.

Objective of the ridge regression

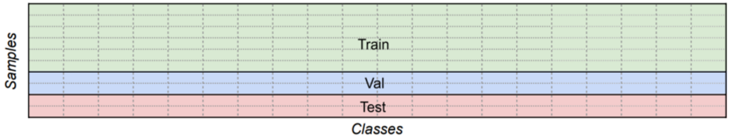
$$J(\mathbf{w}) = \frac{1}{N}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

- Similarly to previously, we can use the lemma to compute the gradient of the loss $J(\mathbf{w})$ w.r.t. $\mathbf{w}$, then set it to 0 to obtain an analytical solution to this problem

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = 0 \implies \mathbf{w} = (\mathbf{X}^\top\mathbf{X} + \lambda N \mathbf{I}_D)^{-1}\mathbf{X}^\top\mathbf{y}$$

Hyper-parameters

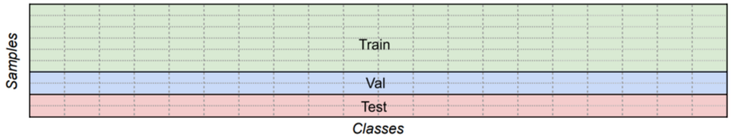
- Importantly, the value of hyper-parameters should be selected using a *validation set*.
- We thus typically have three distinct sets:



- Why not on the testing set?

Hyper-parameters

- Importantly, the value of hyper-parameters should be selected using a *validation set*.
- We thus typically have three distinct sets:

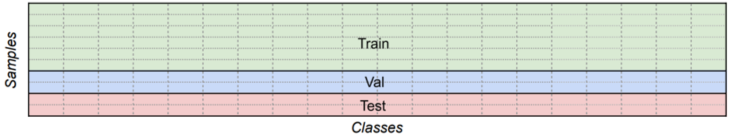


- Why not on the testing set?
  - We would risk overestimating the performance of our model. [4]

[4]

Hyper-parameters

- Importantly, the value of hyper-parameters should be selected using a *validation set*.
- We thus typically have three distinct sets:



- Why not on the testing set?
  - ▸ We would risk overestimating the performance of our model. [4]

[4]For instance, if we had a set of 1000 candidate hyperparameters with little to no direct effect (*e.g.* a seed) on performance, choosing the best using the testing set and reporting this performance would be equivalent to reporting our best score ever on a game as the score we may expect. This is obviously not a really honest estimate.

$\ell 1$-norm and $\ell 2$-norm

- We can also use an $\ell 1$-norm:

$$\frac{1}{N}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1$$

- This is sometimes called *Lasso* regression
- Advantage: sparsity of $\mathbf{w}$ (sets some values to 0)
    - Useful for feature selection
- But no closed-form solution...

- Other variants: combination of $\ell 1$ and $\ell 2$ (elasticnet), ...

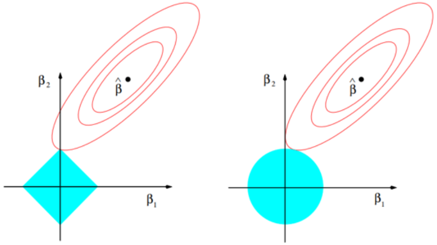


Figure: [a] Selection of parameters $\mathbf{w}$ with constraints on $\|\mathbf{w}\|$. $\hat{\mathbf{w}}$ correspond to solution without constraints. With regularization, $\mathbf{w}$ must be in the blue area. [b] Left: $\ell 1$ norm. Right: $\ell 2$ norm.

[a]Figure from Tibshirani.

[b]There are some links with constrained optimization and Lagrange multipliers. This will be covered later.

Outline

The bias-variance decomposition

- Assumptions:

$$y = f(\mathbf{x}) + \epsilon, \quad \text{where } \epsilon \sim \mathcal{N}(0, \sigma^2)$$
$$\hat{y} = \hat{f}_{\mathcal{D}}(\mathbf{x})$$

$f(\mathbf{x})$: *"Ground truth"* function (generally unknown)

y: Available target(s): ground truth + some noise *e.g.* measurement error

$\hat{f}_{\mathcal{D}}(\mathbf{x})$: Estimated function from dataset $\mathcal{D}$

- We can then write the expected error as:

$$\mathbb{E}[(y - \hat{y})^2] = \mathbb{E}\left[((y - f) + (f - \hat{y}))^2\right]$$
$$\vdots$$
$$= \underbrace{\sigma^2}_{\text{Intrinsic noise}} + \underbrace{\mathbb{E}\left[(f - \mathbb{E}[\hat{f}])^2\right]}_{\text{Bias}^2} + \underbrace{\text{Var}[\hat{f}]}_{\text{Variance}}$$

What does it mean in practice?

- There is a trade-off between bias and variance:
  - Either we have an estimated $\hat{f}$ with low bias, but there will be high variance in the estimated $\hat{f}$
  - Or we have low variance in $\hat{f}$, but we will have higher bias

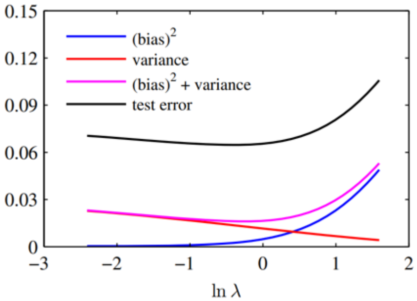
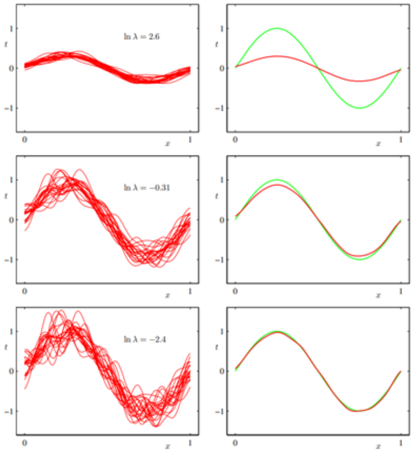


Figure: [5] Increasing the regularization decreases the variance but increases the bias. Total error is the sum of bias, variance and noise.

[5] From Bishop.

Illustration of the trade-off

- We sample different datasets, and estimate $\hat{f}$



High bias
Low variance

Medium bias
Medium variance

Low bias
High variance

Outline

What if the relation is clearly not linear?

- Assuming we have a single input variable x, the graph of the function can only be a line:

$$f(x) = w_1 \cdot x + w_0$$

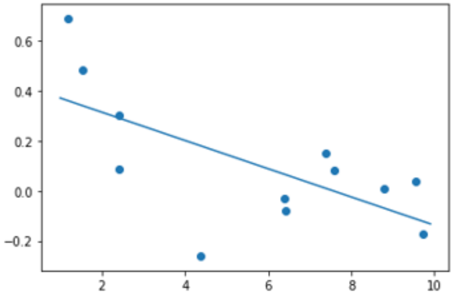


Figure: Example of a linear regression applied on a non linear relation.

- This is not a great fit...

Introduce non-linearity as new features

- We can enhance our original features with non-linear transformations:

Original features:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{pmatrix} = \begin{pmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ \vdots & \vdots \\ x_{N,1} & x_{N,2} \end{pmatrix}$$

"Enhanced" polynomial features:

$$\mathbf{X}^* = \begin{pmatrix} \mathbf{x}_1^* \\ \mathbf{x}_2^* \\ \vdots \\ \mathbf{x}_N^* \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & (x_{1,1})^2 & x_{1,2} & (x_{1,2})^2 & x_{1,1}x_{1,2} & \cdots \\ 1 & x_{2,1} & (x_{2,1})^2 & x_{2,2} & (x_{2,2})^2 & x_{2,1}x_{2,2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 1 & x_{N,1} & (x_{N,1})^2 & x_{N,2} & (x_{N,2})^2 & x_{N,1}x_{N,2} & \cdots \end{pmatrix}$$

- This allows us to capture non-linear relationships using linear regression techniques

Result

- Although we are still technically doing linear regression, adding features via non-linear transformations of x enables us to model non linear relations w.r.t. the initial variable x.

$$f(x) = w_0 \cdot 1 + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3$$

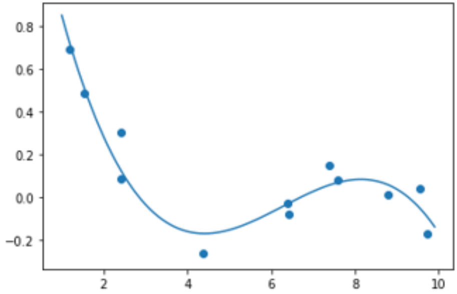


Figure: Non-linear relation between the output and a unique input variable x.

Other possibilities

- There are many possibilities of nonlinear features

Example of feature transformations

$$x^2 \quad x^3 \quad \sqrt{x} \quad \sqrt[3]{x} \quad \sin(\omega x) \quad \log(x) \quad \sigma\left(\frac{x-\mu}{s}\right) \ldots$$

Gaussian basis function: $\exp\left(-\frac{(x-\mu)^2}{2s^2}\right)$

(we will meet this function again later)

But be careful

Adding too many features increases the risk of overfitting.

“With great (polynomial) power comes great responsibility”

– Spiderman?

Bonus: how to deal with w_0

- Sometimes we considered that f is strictly linear:
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$
- And sometimes we added a "bias" or an "intercept" w_0:
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$$
- The 1st situation is actually identical to the 2nd one if we add a column of 1s in the matrix $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^\top$:

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{pmatrix} = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots \\ x_{2,1} & x_{2,2} & \dots \\ \vdots & \vdots & \\ x_{N,1} & x_{N,2} & \dots \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \dots \\ 1 & x_{2,1} & x_{2,2} & \dots \\ \vdots & \vdots & \vdots & \\ 1 & x_{N,1} & x_{N,2} & \dots \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1^* \\ \mathbf{x}_2^* \\ \vdots \\ \mathbf{x}_N^* \end{pmatrix}$$

Then, $$\begin{aligned} f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}^* \quad &= w_0 \cdot 1 + w_1 \cdot x_1 + \dots + w_D \cdot x_D \\ &= \mathbf{w}^\top \mathbf{x} + w_0 \end{aligned}$$

Side node: why least squares?

- Why use least squares in the loss? $\frac{1}{N}\sum_{n=1}^{N}(y_n - f(\mathbf{x}_n))^2$
- Why not the absolute value of the difference? $|y_n - f(\mathbf{x}_n)|$

Answer: under the assumption

$$y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

- the maximum likelihood estimator of $\mathbf{w}$ minimizes the sum of the squared differences

Interpretability

- Can we get insight into how the model "makes decisions"?
  - Mostly yes, the formula is pretty straightforward
- Let's say we trained a linear regression model to predict the price of an appartment based on its surface area and distance to the city center:

$$\text{price}(\text{area}, \text{distance}) = 11,000 \cdot \text{area} - 5,000 \cdot \text{distance} + 30,000$$

- Interpretation:
  - $w_{\text{area}} > 0$: "positive" influence of area: area $\nearrow$, price $\nearrow$
  - $w_{\text{distance}} < 0$: "negative" influence of distance: distance $\nearrow$, price $\searrow$
- Is one variable "more important" than the other?

Interpretability

- What about this one?

$$\text{networth}(\text{salary}, \text{age}) = 2 \cdot \text{salary} \ + \ 2{,}000 \cdot \text{age}$$

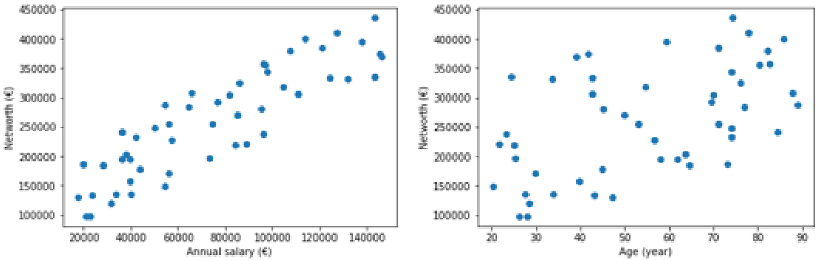


Figure: Plots of networth with respect to salary and age.

- Salary seems to have more importance than age

Interpretability

Variables can (should) be "standardized" for interpretation

$$x_{\text{stand}} = \frac{x - \mu}{\sigma}$$

where μ is the mean and σ the standard deviation of all x.

$$\text{networth}(\text{salary}, \text{age}) = 77,000 \cdot \text{salary}_{\text{stand}} \; + \; 43,000 \cdot \text{age}_{\text{stand}}$$

- Now salary seems to be more important than age

- **Other methods**: we could also use correlation:

$$r_{X,Y} = \frac{\text{Cov}[X, Y]}{\sigma_X \sigma_Y}$$

- But always keep in mind this only works with the assumption that relations are linear

Outline

Linear models for classification

- Let's consider a binary classification problem
  - *e.g.* estimate the 5-year outcome given the size of a tumor
- We have N 1-dimensional
- We can use 1=death, 0=alive

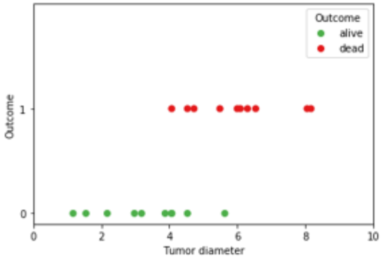


Figure: Plot of data

Fitting a linear regression

- We can fit a linear regression, and predict 1 if our prediction is > 0.5, and 0 otherwise

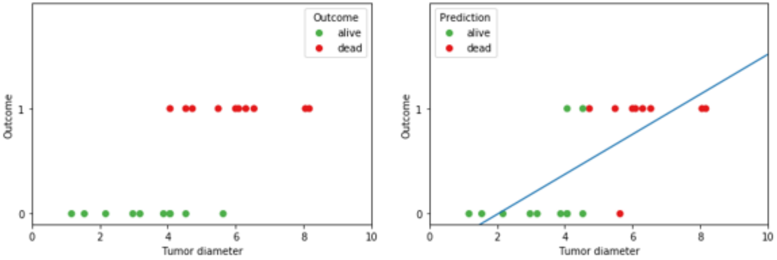


Figure: Plot of data

Robustness to outliers

- What happens if we add some data points?

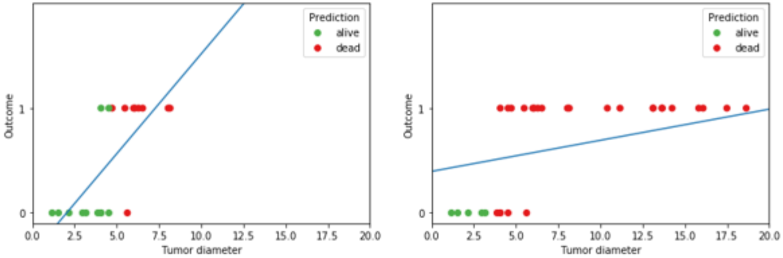


Figure: Plot of data

- These new points do not really bring any new information
- Yet predictions on old points change significantly

The sigmoid function

- It would be more relevant to only predict values between 0 and 1
- We can use the sigmoid function $\sigma(\cdot)$:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

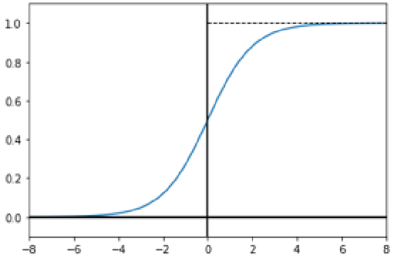


Figure: Plot of data

Predicting probabilities

- Now we can ensure that we only predict "probabilities" between 0 and 1 by apply the sigmoid function $\sigma(\cdot)$ to our initial predictions $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$:

$$\hat{y} = \sigma(f(\mathbf{x})) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}$$

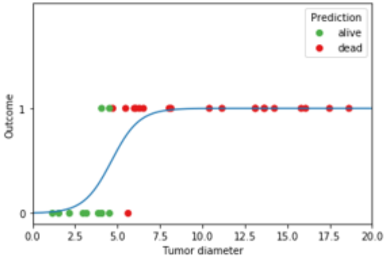


Figure: Plot of data

- This is sometimes called a logistic model.

The logistic loss

- Is using the least square loss still relevant?

The logistic loss

- Is using the least square loss still relevant?
  - Not so much
- Instead, we use the negative log-likelihood:

$$-\log P(y|\hat{y}) = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y}))$$

y: "True" label, $\in \{0,1\}$

$\hat{y}$: Estimated probability, $\in]0,1[$

$$\hat{y} = \frac{1}{1+e^{-\mathbf{w}^\top \mathbf{x}}}$$

- Total training loss: $J(\mathbf{w}) = -\sum_{n=1}^{N} \log P(y_n|\hat{y}_n)$
- This is called the logistic regression loss or log loss

Generalized linear models

- This is a “generalized” linear model, but it is still a linear model: the decision “boundary” is linear

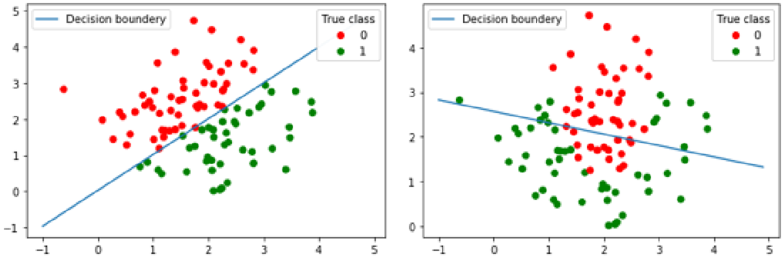


Figure: Plot of data

- It is therefore not (directly) capable of approximating nonlinear decision functions

Further info on logistic regression

- It is of course also possible to introduce nonlinearity in the features
- But it is also possible to overfit
- We can use the same idea as with linear regression and introduce $\ell 2/\ell 1$ regularization
- Everything we said about interpretation is still relevant

Outline

Class summary

To train a supervised learning model, we should:[6]

1. Apply transformations on the data
   - *e.g.* polynomial features, standardization...
2. Select a set of candidate hyper-parameters
   - *e.g.* $\lambda \in \{0.1, 1, 10, 100\}$

For each hyper-parameter(s):

3. Train the chosen model on the training set
   - *e.g.* minimize $J_\lambda(\mathbf{w})$
4. Measure the performance on the validation set

And finally:

5. Select the hyper-parameter(s) with the lowest error
6. Measure the final error on the test set

[6] We are obviously skipping some steps compared to a real use case, such as data collection, data exploration, *asking questions* etc. This here is only the part that is directly related to the machine learning steps that we covered in class.