# Optimisation

6 theoretical lessons — tests in each lecture. (10-15 mins)
3 labs — participation.
1 exam — theory.

---

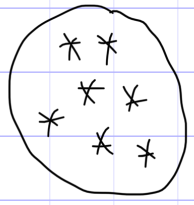
# Machine learning

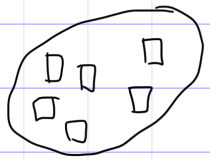
Dataset
$S = \{$

$x$ (input) $\xrightarrow{h}$ $y$
↳ class label
↳ mapping function.
} Classification.

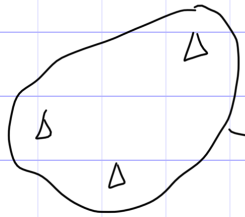
$x$ (input) $\xrightarrow{h}$ $y \in \mathbb{R}$ (↳ value) } Regression.

Regression problem can be transformed into a classif. problem and vice versa.





↳ unsupervised.
} Clustering.
↳ can we assign labels to our data?



→ anomalies.
↳ Strange/not normal

Many dimensions → fewer dimensions. } Dimensionality reduction

# Reinforcement learning.

For this course we will focus on Classif. & Regression.

⇒ Dataset

$$S = \{(x_i, y_i);\ i = 1, 2, \dots, n\}.$$

($x_i$: data, $y_i$: label.)

supervised learning.

$x \xrightarrow{h} y$ } how do we evaluate if $h$ is a good predictor.

$S \longrightarrow$ ML Algorithm $\longrightarrow h$ → hypothesis/regressor/classifier/predictor.

How do we evaluate if $h$ is good?

→ We need to compare $h(x) \longleftrightarrow y$.

For classification we usually use

Class $\quad \mathbb{1}(h(x) \neq y)$ $\quad$ (= 1 if true, 0 otherwise).

→ Loss point of view: −1 for each incorrect * → for optim.

→ Gain point of view: +1 for each correct

Loss function: $\ell(h, (x, y)) = \begin{cases} \mathbb{1}(h(x) \neq y) & \rightarrow \text{class if.} \\ (h(x) - y)^2 & \rightarrow \text{Regres.} \end{cases}$

↳ quadratic loss fn.

↳ there are others.

We want our model to, on average, perform well.

Tomorrow: $(X, Y)$ from distribution $D$.

$$\mathbb{E}_{(x,y)\sim D}[\ell(h,(x,y))]$$

→ expected loss / expected risk. $L_D(h)$ / $R(h)$ or $R_D(h)$.

Our target is to have a low loss in the future, so "find a good predictor" → Minimise the expected loss.

$$\underset{h\in H}{\text{minimise}}\ L_D(h) = \underset{h\in H}{\text{minimise}}\ \mathbb{E}_{(x,y)\sim D}(\ell(h,(x,y))$$

H: class of hypotheses / predictors.

But we do not know D! it can change all the time.
↳ we can't make assumptions

Machine learning is distribution free and why it differs from statistics.

We want a model that works well in the future, but we don't have any info about the future distribution. So we need a model that works well in most cases.

Fundamental assumption of ML:
The dataset that we train on should come from the same distribution that we will test on, otherwise there is no point in learning everything.
↳ Ex. learning to play piano and then having to play violin.
↳ We do not know D, but S is drawn from D.

Solution: Rather than working with expected loss, we work with the empirical loss:

$$L_S(h) = \frac{1}{n}\sum_{i=1}^{n}\ell(h,(x_i,y_i)) \approx \mathbb{E}_{(x,y)\sim D}[\ell(h;(x,y))].$$

↳ more correct for a larger n → Law of Large numbers.

# Empirical Risk Minimisation (ERM)

$$\underset{h \in H}{\text{minimise}} \; L_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h, (x_i, y_i))$$

h can be:

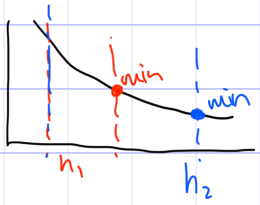
$H_1 = \{ax + b \; ; a, b \in \mathbb{R}\}$

$H_2 = \{ax^2 + bx + c \; ; a, b, c \in \mathbb{R}\}$

$H_1 \subset H_2$. ($H_1$ is in $H_2$.)

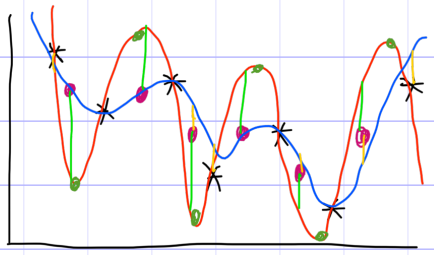
$h_1^* \in \underset{h \in H_1}{\text{argmin}} \; L_S(h)$ $\qquad$ $h_2^* \in \underset{h \in H}{\text{argmin}} \; L_S(h)$

$L_S(h_1^*) \geq L_S(h_2^*)$




$h_2^*$ can be in $h_1^*$ but in general $h_2^*$ is better than $h_1^*$

If we have n data points we can find a polinomial of degree n-1 that will have error = 0.



↳ overfitting.

new data

better model

errors with overfit.

errors with better model.

more new data

now underfits

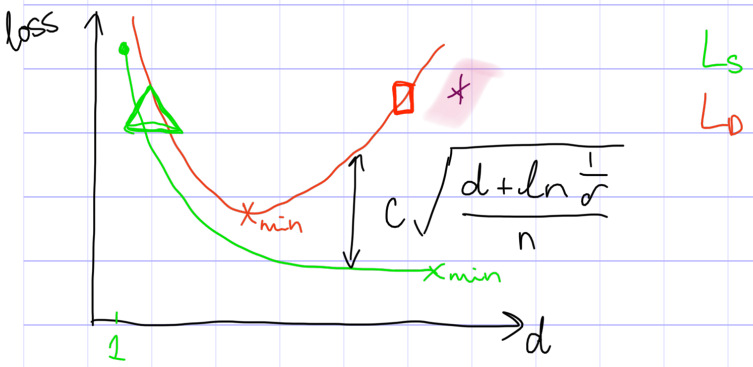
good.

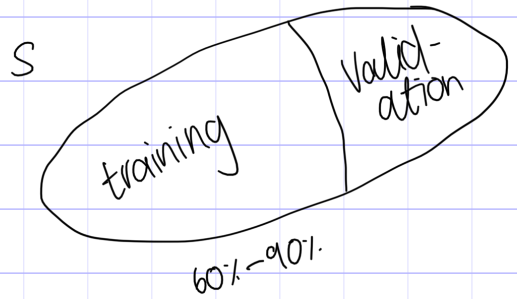
# Statistical learning

$|L_S(h) - L_S(h)| > c\sqrt{\frac{d + \ln\frac{1}{\delta}}{n}}$ with probability $\delta$

$\delta$ → how close you want the values to be.
$d$ → VC dimension → measure of complexity of class H.
Vapnik-Chervonenkis.

for the class of one variable with polinomial degree m:
$d = m+1$.




$S = S_T \cup S_V$

$h_1^* \in \underset{h \in H_1}{\operatorname{argmin}} L_{S_T}(h)$

$h_2^* \in \underset{h \in H_2}{\operatorname{argmin}} L_{S_T}(h)$

$L_{S_T}(h_1^*) \geq h_{S_T}(h_2^*)$ → Validation data will help us decide between $h_1^*$ and $h_2^*$.

$L_{S_V}(h_1^*) < L_{S_V}(h_2^*)$

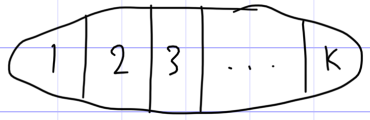
□ pick $h_1^*$ and say: $h_2^*$ was overfitting

$L_{S_V}(h_1^*) > L_{S_V}(h_2^*)$

△ pick $h_2^*$ and say: $h_1^*$ was underfitting.

But sometimes we do not have enough data to split it into train and validation.

We can then use K-fold cross-validation.




before: $h_1^* \in \underset{h \in H_1}{\operatorname{argmin}} L_{S_1}(h)$

now: $h_1^{*(1)} \in \underset{h \in H_1}{\operatorname{argmin}} L_{S_2 \cup \dots S_K}(h)$

$\vdots$

$h_1^{*(2)} \in \underset{h \in H_1}{\operatorname{argmin}} L_{S_1 \cup S_3 \dots S_K}(h)$

$h_1^{*(n)} \in \underset{h \in H_1}{\operatorname{argmin}} L_{S_1 \cup S_2 \dots S_{K-1}}(h)$

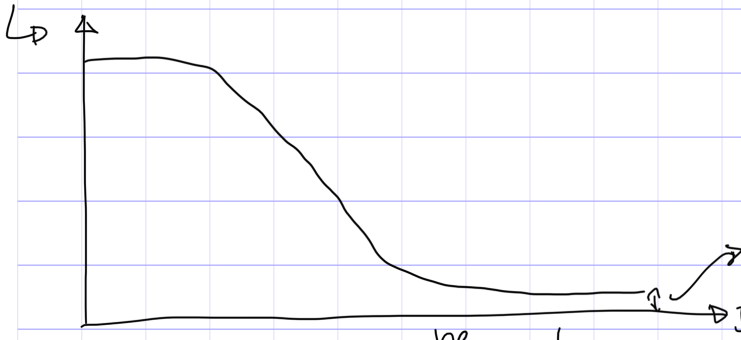
Validation error $= \frac{1}{K} \sum_{i=1}^{K} L_{S_i}(h_1^{*(i)})$

Other algorithm parameters:

- learning rate / step.
- batch size.
- regularisation coefficient.

HYPERPARAMETERS.

what values to choose?
ex. if we have 3 values for each → 27 tests, + 5 fold CV + 3 models
→ Lots of problems.
→ Solve optimisation problems.
↳ to improve efficiency.




minimise $_{h \in H} L_S(h)$ (S can be train / fold / valid etc.)

h is parametrised by a vector of parameters $w \in \mathbb{R}^d$.

$h(x) = \sum_i w_i x_i$

$$L_S(h) = \sum_{i=1}^{n} \ell(h, (x_i, y_i)) =$$

a quadratic loss $= \sum_{i=1}^{n} (h(w, x_i) - y_i)^2$.

$\ell(h(x_i, y_i)) = f(w, i)$

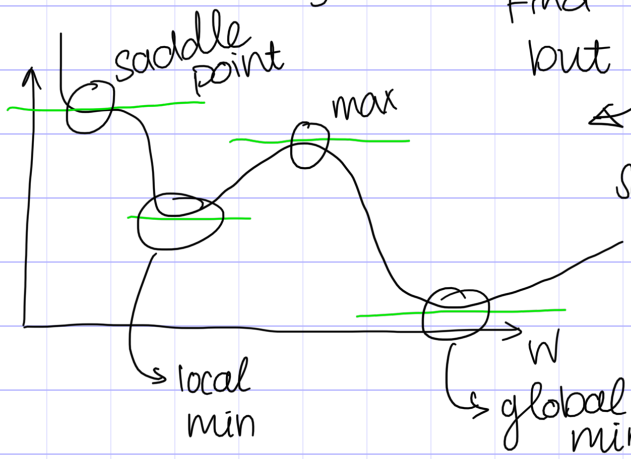
$L_S(h) \rightarrow F(w)$

$\min_{w \in W} F(w) \rightarrow \min_{w \in \mathbb{R}^d} F(w)$

$\min_{w} F(w)$ } How do we do this?

Find where derivative $= 0$

but problems.




so we need second derivative as well.

if $< 0 \rightarrow$ min.

This is for just one variable... We have many variables/parameters...

So we need the Hessian Matrix.

$F'(w)$

$\nabla F(w) = \begin{bmatrix} \frac{\partial F}{\partial w_1} \\ \vdots \\ \frac{\partial F}{\partial w_d} \end{bmatrix} \in \mathbb{R}^d.$

$F(w) = w_1^2 + w_2^2$

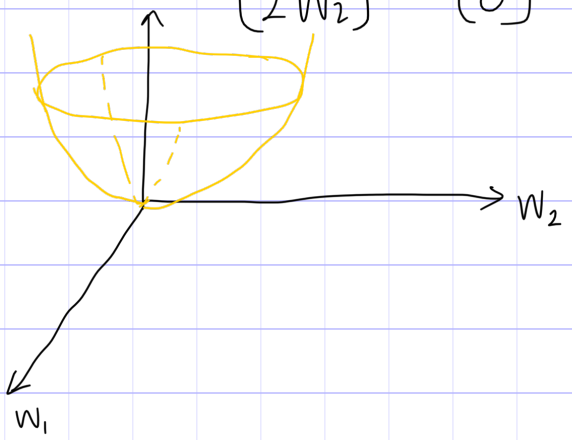
$$\frac{\partial F}{\partial w_1} = 2w_1 + 0 = 2w_1$$

$$\frac{\partial F}{\partial w_2} = 2w_2$$

$$\nabla F = \begin{bmatrix} 2w_1 \\ 2w_2 \end{bmatrix}$$

At a critical point $\longleftrightarrow$ $\nabla F =$ 0 (0 vector.)

$$\therefore \begin{bmatrix} 2w_1 \\ 2w_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \longrightarrow w_1 = w_2 = 0.$$




$$F(w_1, w_2) = K$$

$w_1^2 + w_2^2 = K$ → eqn of circle.