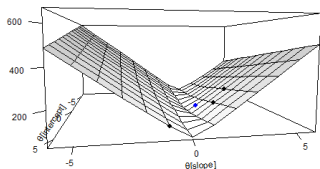


Introduction to Machine Learning

ML-Basics

Losses & Risk Minimization



Learning goals

- Know the concept of loss
- Understand the relationship between loss and risk
- Understand the relationship between risk minimization and finding the best model

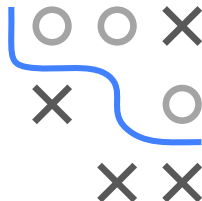
HOW TO EVALUATE MODELS

- When training a learner, we optimize over our hypothesis space, to find the function which matches our training data best.
- This means, we are looking for a function, where the predicted output per training point is as close as possible to the observed label.

Features x		Target y	?	Prediction \hat{y}
People in Office (Feature 1) x_1	Salary (Feature 2) x_2	Worked Minutes Week (Target Variable)		Worked Minutes Week (Target Variable)
4	4300 €	2220	\approx	2588
12	2700 €	1800		1644
5	3100 €	1920		1870

$\mathcal{D}_{\text{train}}$

- To make this precise, we need to define now how we measure the difference between a prediction and a ground truth label pointwise.

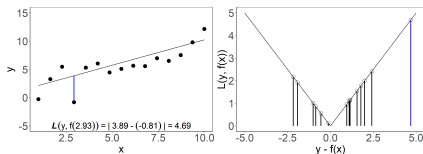


LOSS

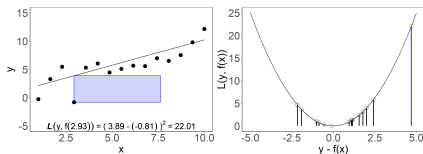
The **loss function** $L(y, f(\mathbf{x}))$ quantifies the "quality" of the prediction $f(\mathbf{x})$ of a single observation \mathbf{x} :

$$L : \mathcal{Y} \times \mathbb{R}^g \rightarrow \mathbb{R}.$$

In regression, we could use the absolute loss $L(y, f(\mathbf{x})) = |f(\mathbf{x}) - y|$;



or the L2-loss $L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$:

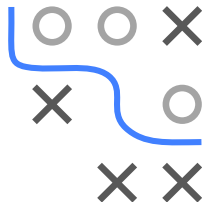


RISK OF A MODEL

- The (theoretical) **risk** associated with a certain hypothesis $f(\mathbf{x})$ measured by a loss function $L(y, f(\mathbf{x}))$ is the **expected loss**

$$\mathcal{R}(f) := \mathbb{E}_{xy}[L(y, f(\mathbf{x}))] = \int L(y, f(\mathbf{x})) d\mathbb{P}_{xy}.$$

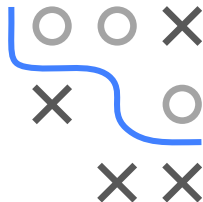
- This is the average error we incur when we use f on data from \mathbb{P}_{xy} .
- Goal in ML: Find a hypothesis $f(\mathbf{x}) \in \mathcal{H}$ that **minimizes** risk.



RISK OF A MODEL / 2

Problem: Minimizing $\mathcal{R}(f)$ over f is not feasible:

- \mathbb{P}_{xy} is unknown (otherwise we could use it to construct optimal predictions).
- We could estimate \mathbb{P}_{xy} in non-parametric fashion from the data \mathcal{D} , e.g., by kernel density estimation, but this really does not scale to higher dimensions (see “curse of dimensionality”).
- We can efficiently estimate \mathbb{P}_{xy} , if we place rigorous assumptions on its distributional form, and methods like discriminant analysis work exactly this way.



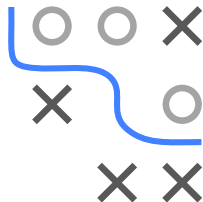
But as we have n i.i.d. data points from \mathbb{P}_{xy} available we can simply approximate the expected risk by computing it on \mathcal{D} .

EMPIRICAL RISK

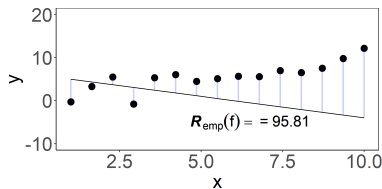
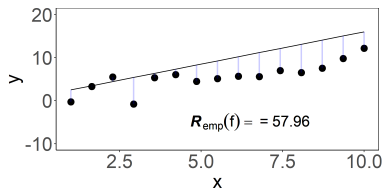
To evaluate, how well a given function f matches our training data, we now simply sum-up all f 's pointwise losses.

$$\mathcal{R}_{\text{emp}}(f) = \sum_{i=1}^n L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right)$$

This gives rise to the **empirical risk function** which allows us to associate one quality score with each of our models, which encodes how well our model fits our training data.



$$\mathcal{R}_{\text{emp}} : \mathcal{H} \rightarrow \mathbb{R}$$



EMPIRICAL RISK MINIMIZATION

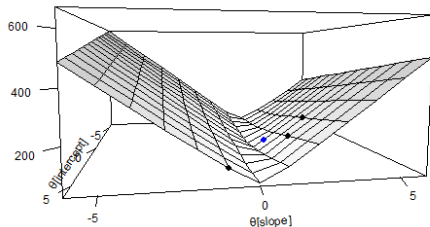
But usually \mathcal{H} is infinitely large.

Instead we can consider the risk surface w.r.t. the parameters θ .
(By this I simply mean the visualization of $\mathcal{R}_{\text{emp}}(\theta)$)



$$\mathcal{R}_{\text{emp}}(\theta) : \mathbb{R}^d \rightarrow \mathbb{R}.$$

Model	$\theta_{\text{intercept}}$	θ_{slope}	$\mathcal{R}_{\text{emp}}(\theta)$
f_1	2	3	194.62
f_2	3	2	127.12
f_3	6	-1	95.81
f_4	1	1.5	57.96



EMPIRICAL RISK MINIMIZATION / 2

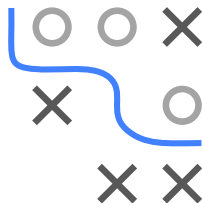
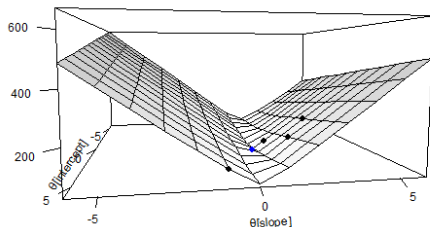
Minimizing this surface is called **empirical risk minimization** (ERM).

$$\hat{\theta} = \arg \min_{\theta \in \Theta} \mathcal{R}_{\text{emp}}(\theta).$$

Usually we do this by numerical optimization.

$$\mathcal{R} : \mathbb{R}^d \rightarrow \mathbb{R}.$$

Model	$\theta_{\text{intercept}}$	θ_{slope}	$\mathcal{R}_{\text{emp}}(\theta)$
f_1	2	3	194.62
f_2	3	2	127.12
f_3	6	-1	95.81
f_4	1	1.5	57.96
f_5	1.25	0.90	23.40



In a certain sense, we have now reduced the problem of learning to **numerical parameter optimization**.