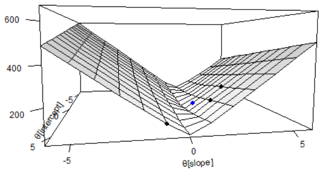
Introduction to Machine Learning

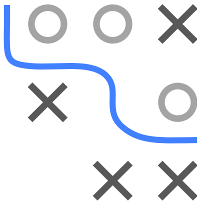
ML-Basics
Losses & Risk Minimization



Learning goals

- Know concept of loss function
- Understand concept of theoretical and empirical risk
- Understand relationship between risk minimization and finding best model

HOW TO EVALUATE MODELS



- Training a learner = optimize over hypothesis space
- Find function that matches training data best
- We compare point-wise predicted outputs to observed labels

Features x	
People in Office (Feature 1) x_1	Salary (Feature 2) x_2
4	4300 €
12	2700 €
5	3100 €

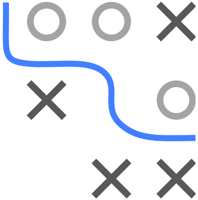
Target y
Worked Minutes Week (Target Variable)
2220
1800
1920

$\overset{?}{\approx}$

Prediction $\hat{y}$
Worked Minutes Week (Target Variable)
2588
1644
1870

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

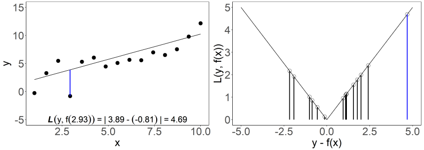
LOSS



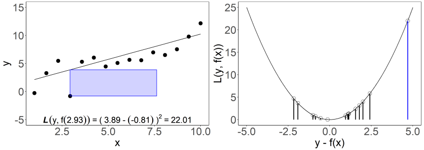
Loss function $L(y, f(\mathbf{x}))$ quantifies point-wise how we measure errors in predictions for a single $\mathbf{x}$:

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

Regression: Could use absolute L1 loss $L(y, f(\mathbf{x})) = |y - f(\mathbf{x})|$



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



RISK OF A MODEL

- Theoretical **risk** of a candidate model f is the **expected loss**

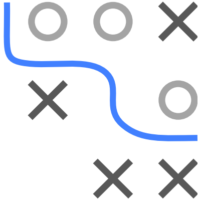
$$\mathcal{R}(f) := \mathbb{E}_{xy}[L(y, f(\mathbf{x}))] = \int L(y, f(\mathbf{x}))\, d\mathbb{P}_{xy}$$

- Average error we incur when we use f on data from $\mathbb{P}_{xy}$
- Goal in ML: Find a hypothesis $f \in \mathcal{H}$ that **minimizes** this

Problems:

- $\mathbb{P}_{xy}$ is unknown
- Could estimate $\mathbb{P}_{xy}$ non-parametrically, e.g., by kernel density estimation, doesn't scale to higher dimensions
- Could efficiently estimate $\mathbb{P}_{xy}$, if we place assumptions on its form, e.g. cf. discriminant analysis

EMPIRICAL RISK

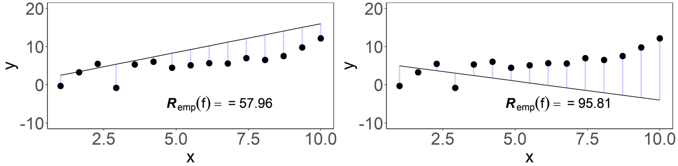


- Have n i.i.d. data from $\mathbb{P}_{xy}$, approximate expected risk empirically
- Just sum up all losses over training data

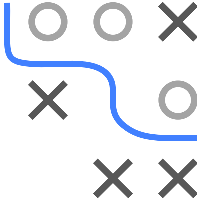
$$\mathcal{R}_{\text{emp}}(f) = \sum_{i=1}^{n} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right)$$

- Associates one quality score with each $f \in \mathcal{H}$
- Encodes: How well does f fits training data
- Now we get very close to solve this by optimization

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



EMPIRICAL RISK



- Can also define as average loss

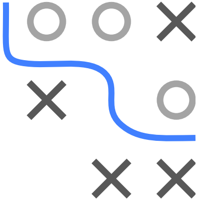
$$\bar{\mathcal{R}}_{\text{emp}}(f) = \frac{1}{n}\sum_{i=1}^{n} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right)$$

- Constant factor $\frac{1}{n}$ doesn't make a difference in optimization
- We usually use $\mathcal{R}_{\text{emp}}(f)$
- Since f is usually defined by **parameters** $\boldsymbol{\theta}$, this becomes:

$$\mathcal{R}_{\text{emp}} : \mathbb{R}^d \to \mathbb{R}$$

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

EMPIRICAL RISK MINIMIZATION



- Best model = smallest risk
- For finite $\mathcal{H}$: we could tabulate exhaustively

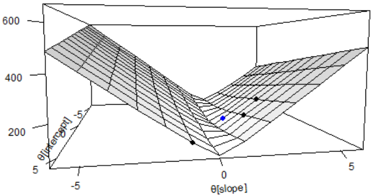
Model	$\theta_{intercept}$	θ_{slope}	$\mathcal{R}_{\text{emp}}(\theta)$
f_1	2	3	194.62
f_2	3	2	127.12
f_3	6	-1	95.81
f_4	1	1.5	57.96

EMPIRICAL RISK MINIMIZATION

- But usually $\mathcal{H}$ is infinitely large
- Instead: Simply consider risk surface w.r.t. the parameters $\boldsymbol{\theta}$

$$\mathcal{R}_{\text{emp}} : \mathbb{R}^d \to \mathbb{R}$$

Model	$\boldsymbol{\theta}_{intercept}$	$\boldsymbol{\theta}_{slope}$	$\mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$
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



EMPRICAL RISK MINIMIZATION

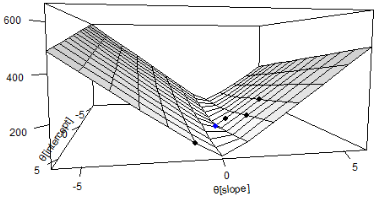
Minimizing this surface is called **empirical risk minimization** (ERM)

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

Usually we do this by numerical optimization

$$\mathcal{R}_{\text{emp}} : \mathbb{R}^d \to \mathbb{R}$$

Model	$\boldsymbol{\theta}_{intercept}$	$\boldsymbol{\theta}_{slope}$	$\mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$
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



Kind of: Reduced "learning" to **numerical parameter optimization**
(Later we will learn that this is only part of the complete picture!)