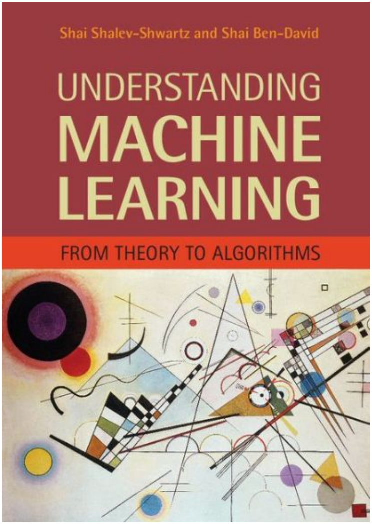
Machine Learning Theory 2025
Lecture 1

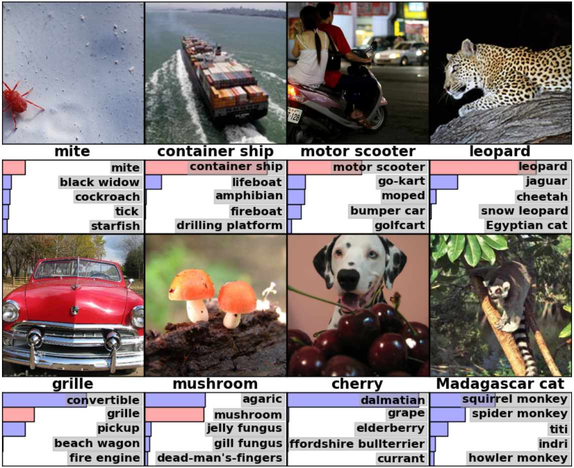
Tim van Erven

- Intro
- Statistical Decision Theory
- Empirical Risk Minimization and Overfitting
- PAC-Learnability for finite classes, realizable case

Book: Shai^2 (for First Half of the Course)



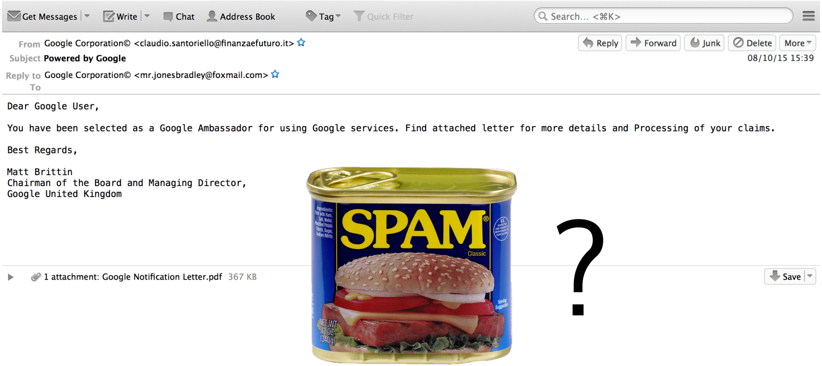
Multiclass Classification Example: Images



Y = image class, $\boldsymbol{X}$ = vector with pixel values

Krizhevsky, Sutskever, Hinton, **ImageNet Classification with Deep Convolutional Neural Networks**, NeurIPS 2012

Binary Classification Example: Spam Detection



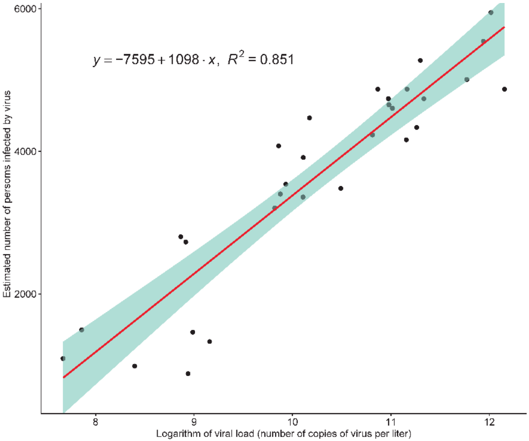
Y = ham/spam

$\boldsymbol{X} = (X_1, \ldots, X_{50\,000})$: X_i is word count for i-th word from dictionary

Spam image by Qwertyxp2000 from https://commons.wikimedia.org/wiki/File:Spam_can.png

Regression Example: Covid Cases from Wastewater

Y = Active number of Covid-19 cases



$\boldsymbol{X} = X_1$ = Log-viral load in wastewater

Vallejo et al., **Highly predictive regression model of active cases of COVID-19 in a population by screening wastewater viral load**, medRxiv preprint, 2020

Regression Example: Prostate Cancer

Goal: Predict level of prostate specific antigen (PSA) for men with prostate cancer

Y = log of PSA
$\boldsymbol{X} = (X_1, \ldots, X_{97})$: 97 clinical measures, including

- log cancer volume
- log prostate weight
- Gleason score
- ...

Example from Hastie, Tibshirani, Freedman, **Elements of Statistical Learning**, 2nd edition, 2009

Scope of the Course I: Supervised vs Unsupervised

In the Course:
Supervised Machine Learning: Learn to predict response Y for input $\boldsymbol{X}$ based on examples of desired responses. E.g.

- Image classification: $\boldsymbol{X}$ = image, Y = class
- Spam classification: $\boldsymbol{X}$ = e-mail, Y = ham/spam
- Covid regression: $\boldsymbol{X}$ = viral load, Y is nr. of active cases
- Cancer regression: $\boldsymbol{X}$ = clinical measures, Y = antigen amount

Not in the Course:
Unsupervised Machine Learning: Identify structure in inputs $\boldsymbol{X}$. E.g.

- Group data into clusters
- Dimensionality reduction

Scope of the Course II: Batch and Online

We cover two learning models:

Part I, **Batch Learning:**

- Data is obtained as one big batch
- Then learn a predictor
- Deploy predictor once, to be used unchanged on new data

Part II, **Online Learning:**

- Data arrives sequentially over time
- Continuously make predictions for incoming data
- Use new data to keep improving predictor

Scope of this Course III: Foundations vs Practice

What is Missing:

- Not: programming, real data, getting rich and famous quickly…
- By itself this course is **too theoretical**!

…But We Make Up for It:

- Deep understanding via **beautiful concepts and proofs**
- When is learning possible and what are the fundamental limitations?
- Close connections to statistics, game theory, information theory, optimization, …

Supervised Learning

Sample of **training data**: $S = \begin{pmatrix} Y_1 \\ \boldsymbol{X}_1 \end{pmatrix}, \cdots, \begin{pmatrix} Y_m \\ \boldsymbol{X}_m \end{pmatrix}$

(teacher shows us desired response Y_i for input $\boldsymbol{X}_i$)

Y_i: class/response variable
$\boldsymbol{X}_i \in \mathbb{R}^d$: feature vectors

Goal: Learn function $h_S : \mathcal{X} \to \mathcal{Y}$ from **hypothesis class** $\mathcal{H} =$ some set of functions

Supervised Learning

Sample of **training data**: $S = \begin{pmatrix} Y_1 \\ \boldsymbol{X}_1 \end{pmatrix}, \cdots, \begin{pmatrix} Y_m \\ \boldsymbol{X}_m \end{pmatrix}$

(teacher shows us desired response Y_i for input $\boldsymbol{X}_i$)

Y_i: class/response variable
$\boldsymbol{X}_i \in \mathbb{R}^d$: feature vectors

Goal: Learn function $h_S : \mathcal{X} \to \mathcal{Y}$ from **hypothesis class** $\mathcal{H}$ = some set of functions

Evaluate h_S on **test data**:

- New $\boldsymbol{X}$ from same source
- Predict corresponding Y by $\hat{Y} = h_S(\boldsymbol{X})$

Assume $\begin{pmatrix} Y_i \\ \boldsymbol{X}_i \end{pmatrix}$ independent samples from same probability distribution $\mathcal{D}$

Avoid further assumptions on $\mathcal{D}$!
(So $\mathcal{D}$ can be very complicated)

Supervised Learning: Regression

$$S = \begin{pmatrix} Y_1 \\ \boldsymbol{X}_1 \end{pmatrix}, \cdots, \begin{pmatrix} Y_m \\ \boldsymbol{X}_m \end{pmatrix}$$

$Y \in \mathbb{R}$ is a **continuous variable**. E.g.

- Y = Covid-19 cases
  $\boldsymbol{X} = (X_1, X_2)$: X_1 = viral load, X_2 = population size

Linear Regression ($\mathcal{H}$ = affine functions):

$$h_{\boldsymbol{w},b}(\boldsymbol{X}) = b + \langle \boldsymbol{w}, \boldsymbol{X} \rangle = b + \sum_{i=1}^{d} w_i X_i$$

Can assume $b = 0$ w.l.o.g. to simplify notation, because:

$$\boldsymbol{w}' = (b, w_1, \ldots, w_d) \qquad \boldsymbol{X}' = (1, X_1, \ldots, X_d)$$

$$h_{\boldsymbol{w}'}(\boldsymbol{X}') = \langle \boldsymbol{w}', \boldsymbol{X}' \rangle = h_{\boldsymbol{w},b}(\boldsymbol{X})$$

Supervised Learning: Classification

$$S = \begin{pmatrix} Y_1 \\ \boldsymbol{X}_1 \end{pmatrix}, \cdots, \begin{pmatrix} Y_m \\ \boldsymbol{X}_m \end{pmatrix}$$

Y is a **categorical variable**

- E.g. $Y \in \{\text{Ham}, \text{Spam}\}$ or $Y \in \{\text{Mite}, \text{Leopard}, \text{Mushroom}\}$

Binary Classification (with two classes):

- Can e.g. map "Ham" $\mapsto -1$, "Spam" $\mapsto +1$
- So assume $Y \in \{-1, +1\}$ or sometimes $Y \in \{0, 1\}$ without loss of generality (w.l.o.g.)

Halfspaces ($\mathcal{H}$ = Linear Predictors):

$$h_{\boldsymbol{w},b}(\boldsymbol{X}) = \text{sign}(b + \langle \boldsymbol{w}, \boldsymbol{X} \rangle) \in \{-1, +1\}$$

Overfitting
(why machine learning is non-trivial)

The #1 Beginner's Mistake:

- Try many machine learning methods and fine-tune their settings until the number of mistakes on the training data S is small
- What can go wrong?

Poll:

1. Trying many methods and settings can take a very long time.
2. Few mistakes on S does not guarantee good learning.
3. You should only use methods taught in this course.

Overfitting
(why machine learning is non-trivial)

The #1 Beginner's Mistake:

- Try many machine learning methods and fine-tune their settings until the number of mistakes on the training data S is small
- What can go wrong?

Poll:

1. Trying many methods and settings can take a very long time.
2. **Few mistakes on S does not guarantee good learning!**
3. ~~You should only use methods taught in this course.~~

Overfitting
(why machine learning is non-trivial)

The #1 Beginner's Mistake:

- Try many machine learning methods and fine-tune their settings until the number of mistakes on the training data S is small
- What can go wrong?

- Suppose $\boldsymbol{X}$ is uniformly distributed in $[-1, +1]^2$
- $Y = +1$ if $X_1 \geq 0$; $Y = -1$ otherwise.

$$h_S(\boldsymbol{X}) = \begin{cases} Y_i & \text{for smallest } i \in \{1, \ldots, m\} \text{ such that } \boldsymbol{X} = \boldsymbol{X}_i \\ -1 & \text{if no such } i \text{ exists} \end{cases}$$

Perfect on training data S,
but probability of mistake $= 1/2$ on new $(\boldsymbol{X}, Y)$ from $\mathcal{D}$!
No better than random guessing!

Statistical Decision Theory I: Loss

Measure error by **loss function**: $\ell(h, \boldsymbol{X}, Y)$

Classification (0/1-loss counts mistakes):

$$\ell(h, \boldsymbol{X}, Y) = \begin{cases} 0 & \text{if } h(\boldsymbol{X}) = Y \\ 1 & \text{if } h(\boldsymbol{X}) \neq Y \end{cases}$$

Regression (Squared Error):

$$\ell(h, \boldsymbol{X}, Y) = (Y - h(\boldsymbol{X}))^2$$

Other choices possible!
(Depends on what is important in your application)

Statistical Decision Theory II: Risk

Risk: $L_{\mathcal{D}}(h) = \mathbb{E}[\ell(h, \boldsymbol{X}, Y)]$ for $(\boldsymbol{X}, Y) \sim \mathcal{D}$

Empirical Risk: $$L_S(h) = \frac{1}{m}\sum_{i=1}^{m} \ell(h, \boldsymbol{X}_i, Y_i)$$

Bayes Optimal Predictor: $f_{\mathcal{D}} \in \arg\min_f L_{\mathcal{D}}(f)$

- Unknown, because risk depends on $\mathcal{D}$
- No learning alg can do better (by definition)

Examples for Classification:

- $L_{\mathcal{D}}(h) = \Pr(h(\boldsymbol{X}) \neq Y)$
- $L_S(h) =$ proportion of mistakes on the training data S
- $f_{\mathcal{D}}(\boldsymbol{X}) = \arg\max_y \Pr(Y = y \mid \boldsymbol{X})$ is most likely class

Statistical Decision Theory II: Risk

Risk: $L_{\mathcal{D}}(h) = \mathbb{E}[\ell(h, \boldsymbol{X}, Y)]$ for $(\boldsymbol{X}, Y) \sim \mathcal{D}$

Empirical Risk: $L_S(h) = \frac{1}{m} \sum_{i=1}^{m} \ell(h, \boldsymbol{X}_i, Y_i)$

Bayes Optimal Predictor: $f_{\mathcal{D}} \in \arg\min_f L_{\mathcal{D}}(f)$

- Unknown, because risk depends on $\mathcal{D}$
- No learning alg can do better (by definition)

Empirical Risk Minimization (ERM): $f_s \in \arg\min_{h \in \mathcal{H}} L_S(h)$

- Minimize **empirical risk** (known) instead of risk (unknown)
- Restrict to **hypothesis class** $\mathcal{H}$ to prevent overfitting

Choice of $\mathcal{H}$ is a **modeling decision**,
made before seeing the data!

No Overfitting for (Multiclass) Classification

Definition (Realizability assumption)

Exists $h^* \in \mathcal{H}$ that perfectly predicts Y (with probability 1): $\Pr(h^*(\boldsymbol{X}) = Y) = 1$.

Huge simplification:

- $Y = h^*(\boldsymbol{X})$ without any noise
- We were lucky enough to include h^* in $\mathcal{H}$

No Overfitting for (Multiclass) Classification

Definition (**Realizability assumption**)

Exists $h^* \in \mathcal{H}$ that perfectly predicts Y (with probability 1): $\Pr(h^*(\boldsymbol{X}) = Y) = 1$.

Huge simplification:

- $Y = h^*(\boldsymbol{X})$ without any noise
- We were lucky enough to include h^* in $\mathcal{H}$

Theorem (First Example of PAC-Learning)

Assume $\mathcal{H}$ is **finite**, **realizability** *holds. Choose any $\delta \in (0, 1)$, $\epsilon > 0$. Then, for all $m \geq \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon}$, ERM over $\mathcal{H}$ guarantees*

$$L_{\mathcal{D}}(h_S) \leq \epsilon$$

with probability at least $1 - \delta$.

NB Lower bound on m does not depend on $\mathcal{D}$ or on h^*!

PAC learning: probably approximately correct

Proof (handwritten)

Recall that $L_D(h) = \Pr(h(\boldsymbol{X}) \neq Y)$
'Bad" hypotheses: $\mathcal{H}_B = \{h \in \mathcal{H} : \Pr(h(\boldsymbol{X}) \neq Y) > \epsilon\}$
ERM only selects a bad hypothesis h if $L_S(h) = 0$.
So sufficient to show that

$$\Pr(\text{exists } h \in \mathcal{H}_B : L_S(h) = 0) \leq \delta.$$

Lemma (Union Bound)

For any two events A and B, $\Pr(A \text{ or } B) \leq \Pr(A) + \Pr(B)$.

Hence

$$\Pr(\text{exists } h \in \mathcal{H}_B : L_S(h) = 0) \leq \sum_{h \in \mathcal{H}_B} \Pr(L_S(h) = 0)$$
$$\leq \sum_{h \in \mathcal{H}_B} (1-\epsilon)^m \leq |\mathcal{H}|(1-\epsilon)^m \leq |\mathcal{H}|e^{-\epsilon m}$$

This is guaranteed to be at most δ if $m \geq \frac{\ln(|\mathcal{H}|/\delta)}{\epsilon}$.

Close Relation to Statistics, But...

Stats:

- Estimate **true parameters**, with uncertainty quantification
- Follow rigorous procedures or results are nonsense

Machine Learning:

- Estimate parameters that **predict well**
  - Possible under weaker assumptions/more complicated models!
- Can always estimate risk on a test set, even for crazy learning algorithm $\rightarrow$ **cowboy mentality can work!**
- (Fast!) algorithms

ML vs Stats (Handwritten)