Deep Learning for NLP

Lecture 2 – ML principles

Dr. Steffen Eger
Wei Zhao
Niraj Dev Pandey



Natural Language Learning Group (NLLG)
Technische Universität Darmstadt

This lecture:

- Machine Learning Principles
  - Train/dev/test split
  - Evaluation
  - Loss functions

- **Learning goals:**
  - Understand ML & DL foundations

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

ML principles

NLLG

Before we start: Notation

TECHNISCHE UNIVERSITÄT DARMSTADT

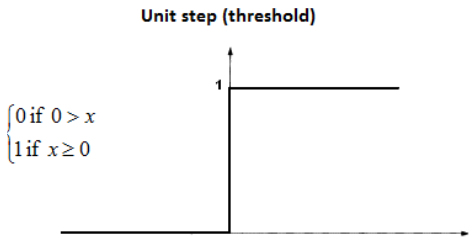
- Throughout, we write
  - Vectors (elements of $\boldsymbol{R}^d$) as bold face letters
    - E.g., $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}, \boldsymbol{x}$
    - Other conventions used in literature:
      1. $\vec{u}, \vec{x}$, ... (adopted from physics)
      2. Nothing: u, v, w, x
         This can be very confusing if in addition you don‘t specify the range of variables („meaningless formulae“)
  - Scalars as ordinary letters: a, b, c, d
  - Matrices are bold and uppercase: $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{X}, \boldsymbol{Y}$

Before we start: Notation

- Often, we assume that vectors are *row vectors*, i.e., lie in $\boldsymbol{R}^{1\times n}$
  - I'll try to be explicit on this

- A dot can mean different things, depending on context:
  - Multiplication: $a \cdot b$
  - Dot product: $\boldsymbol{x} \cdot \boldsymbol{y} = \sum x_i \cdot y_i$
  - Matrix-vector or matrix-matrix multiplication: $\boldsymbol{x} \cdot \boldsymbol{W}$, $\boldsymbol{U} \cdot \boldsymbol{V}$ (note that dimensions must fit here - same is for dot product!)

- A note on differentiability:
  - What is $\sigma'(x)$ for this function?



Before we start: Notation

TECHNISCHE UNIVERSITÄT DARMSTADT

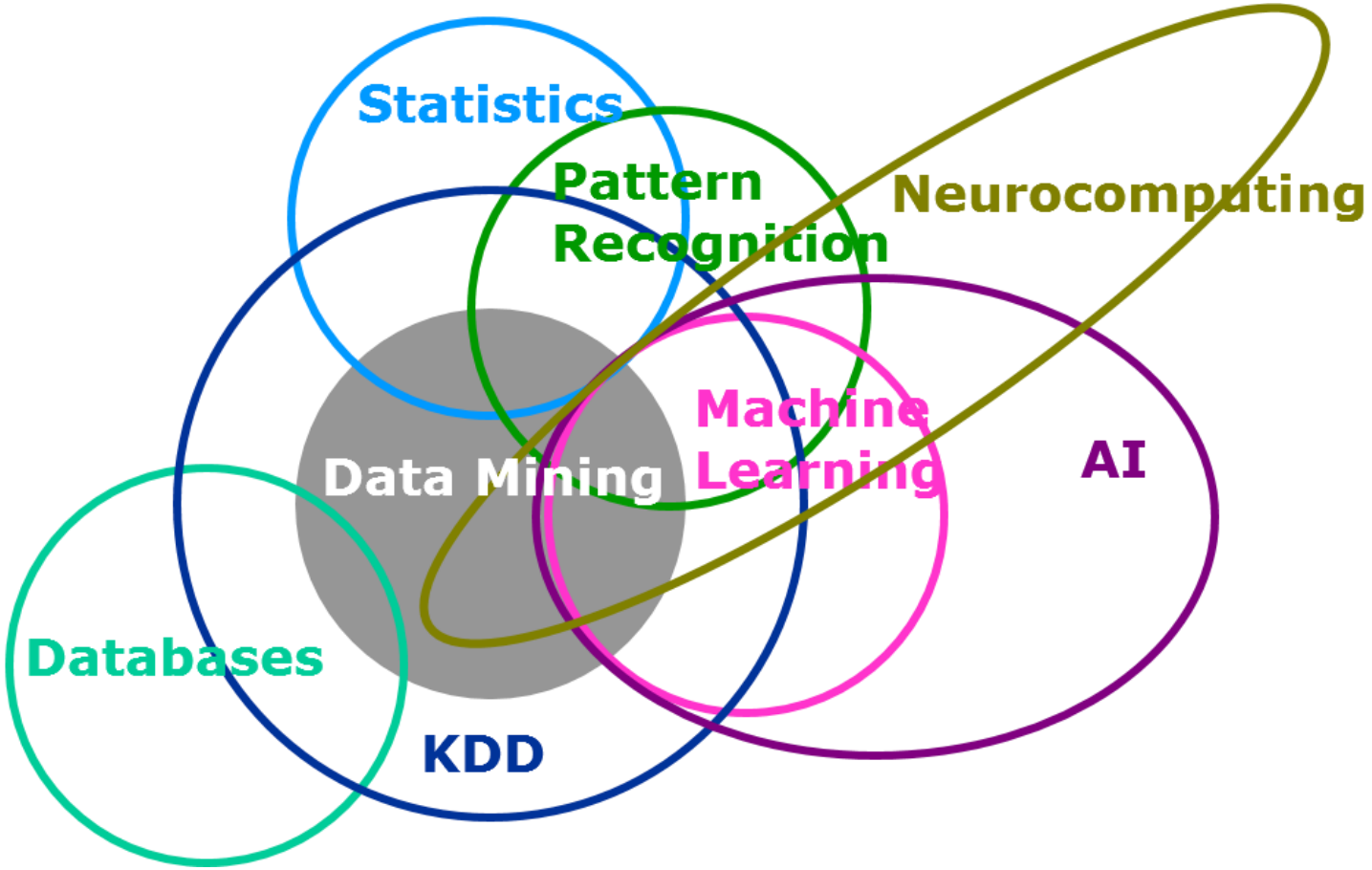
- Elementary vector-matrix multiplication:
  - What is $\boldsymbol{x} \cdot \boldsymbol{E}$, where
    - $\boldsymbol{x} \in \boldsymbol{R}^{1\times n}$ is a 1-hot vector and $\boldsymbol{E} \in \boldsymbol{R}^{n\times d}$?
- Cosine similarity:
  - For two vectors $\boldsymbol{x}, \boldsymbol{y}$, their *cosine similarity* is defined as:
    - $\frac{\boldsymbol{x}\cdot\boldsymbol{y}}{||\boldsymbol{x}||\cdot||\boldsymbol{y}||} \in [-1,1]$

NLLG

Machine Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

- A fancy (mystifying) name for a rather „ordinary“ field of study



NLLG

Standard Setup (Supervised setting)

TECHNISCHE UNIVERSITÄT DARMSTADT

- We have **data**
  - $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)$

Alternative Notation: $(\boldsymbol{x_i}, t_i)$

$\boldsymbol{x_i}$ or $(\boldsymbol{x}_i, t_i)$ is also called *instance*
t_i is called truth / gold label

- And a statistical model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$

Outputs/predictions of model are denoted as y or $\hat{y}$

- We also specify some *loss* function, e.g.,
  - $\frac{1}{N} \cdot \sum_{(x,y)} \left(y - f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)^2$

NLLG

Standard Setup

TECHNISCHE UNIVERSITÄT DARMSTADT

- We have **data**
  - $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)$
- And a statistical model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$
- We also specify some *loss* function, e.g.,
  - $\frac{1}{N} \cdot \sum_{(x,y)} \left(y - f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)^2$
- **Goal** is then to optimize/minimize our loss over $\boldsymbol{\theta}$:
  - We are looking for parameters that bring our model close to the data

Overfitting

- Loss: $\frac{1}{N} \cdot \sum_{(x,y)} \left(y - f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)^2$

- Goal is then to optimize/minimize our loss over $\boldsymbol{\theta}$

- HOWEVER: our (real) goal is NOT to fit the **given** data well (which is called **overfitting**)

  - Every sufficiently rich model $f_{\boldsymbol{\theta}}$ can do this

  - Moreover, if our data is non-pathological, we'll always find a model $f_{\boldsymbol{\theta}}$ that perfectly fits it

Generalization

TECHNISCHE UNIVERSITÄT DARMSTADT

- Can assume our previous optimization problem is a „proxy“ of our true optimization problem (*); but must keep the latter in mind, too
- Because we‘re interested in generalization performance, **we always split our data**:
  - Test data vs. Train data
  - Test data represents the true underlying distribution
- Your model must perform well on the test data; its performance on the training data is not of interest to us

Training data

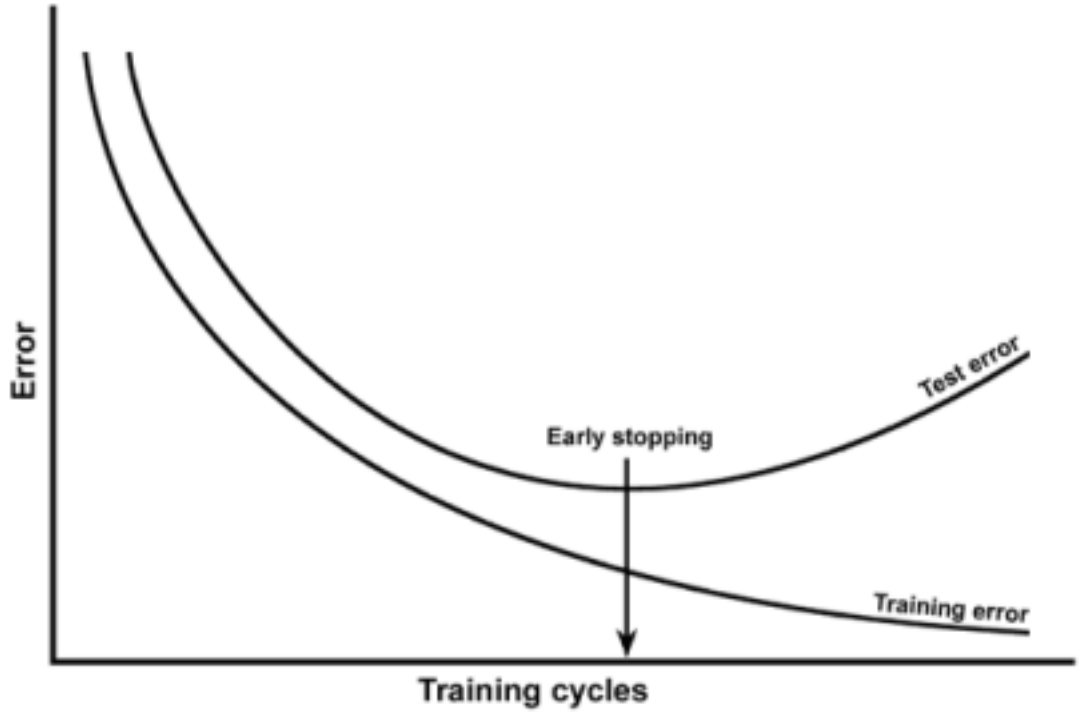
TECHNISCHE UNIVERSITÄT DARMSTADT

- We further split it into development+(proper) train data
- On the proper training data, we optimize the **parameters $\boldsymbol{\theta}$** of our model $f_{\boldsymbol{\theta}}$
- On the dev(elopment) data, we optimize the **hyperparameters** of our model
  - E.g. Learning rate, regularization terms/coefficients, batch size, number of epochs, etc.
  - In particular: can use dev data for **early stopping**, etc. (extremely common and popular)

Training data

TECHNISCHE UNIVERSITÄT DARMSTADT

In particular: can use dev data for **early stopping**, etc. (extremely common and popular)



NLLG

Test data

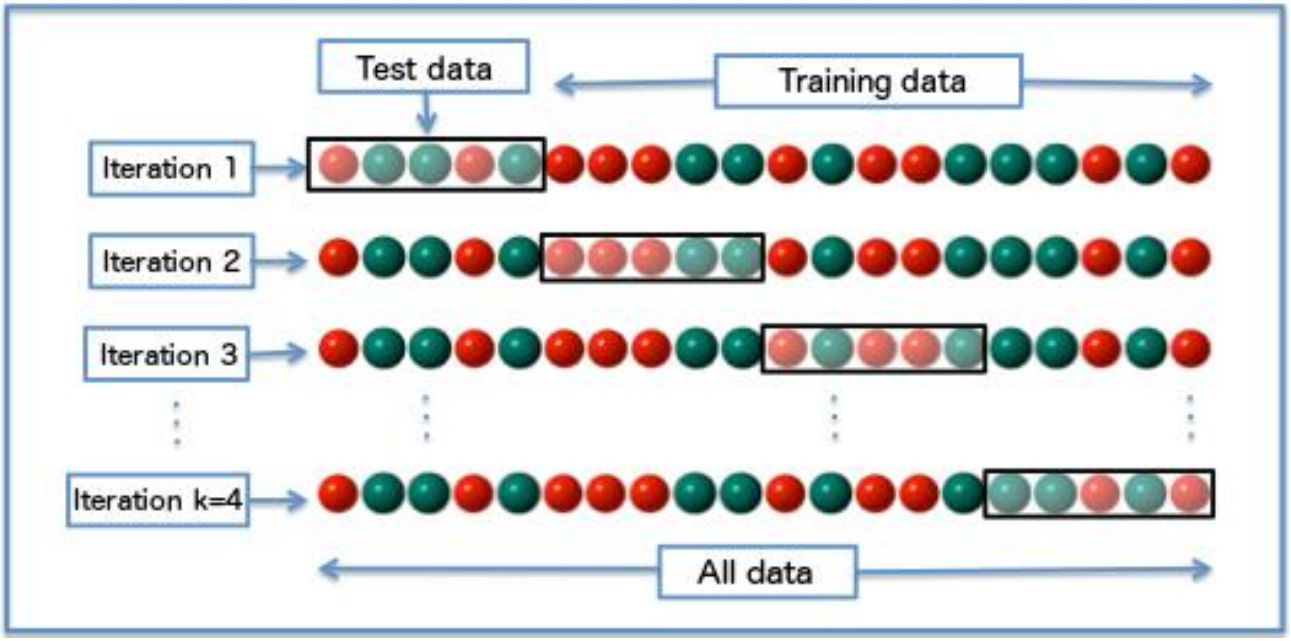
TECHNISCHE UNIVERSITÄT DARMSTADT

- Could choose some fraction of your overall data for testing, e.g., 75%/25% split (train/test) or 90%/10%
- That‘s not always a good idea: by chance, you could pick 10% that are not representative of the overall problem
- Better is k-fold cross-validation:
  - Train on (k-1) equally sized folds, test on the remaining
  - Repeat k times

Test data

TECHNISCHE UNIVERSITÄT DARMSTADT

- Better is k-fold cross-validation:
  - Train on (k-1) equally sized folds, test on the remaining
  - Repeat k times



NLLG

Hyperparameter search

- How to search for good hyperparameters?
  - Grid search
  - Random search (Bergstra and Bengio 2012)
  - Bayesian Methods

Hyperparameter search

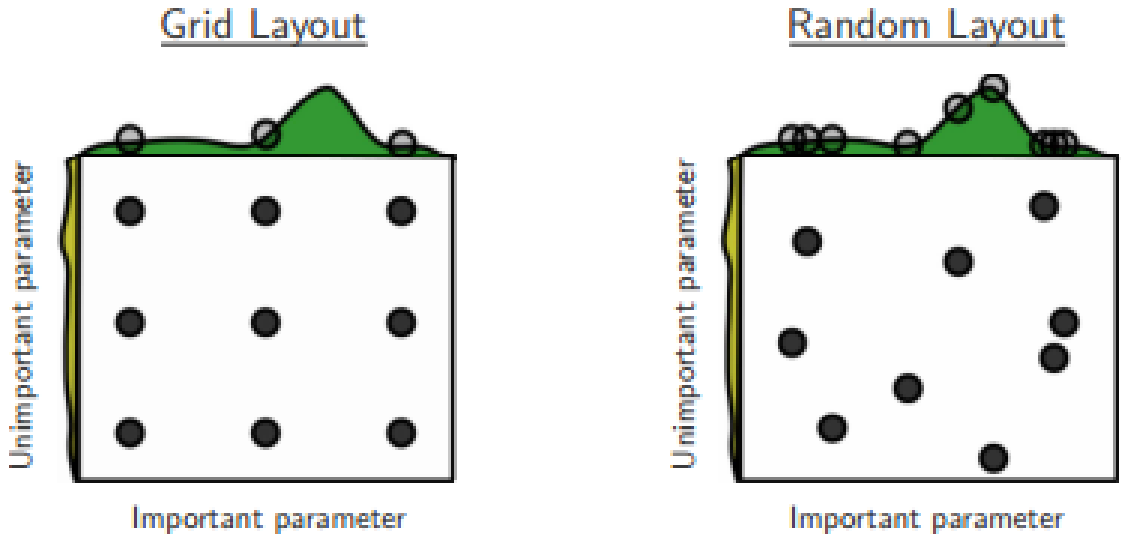


Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

Evaluation

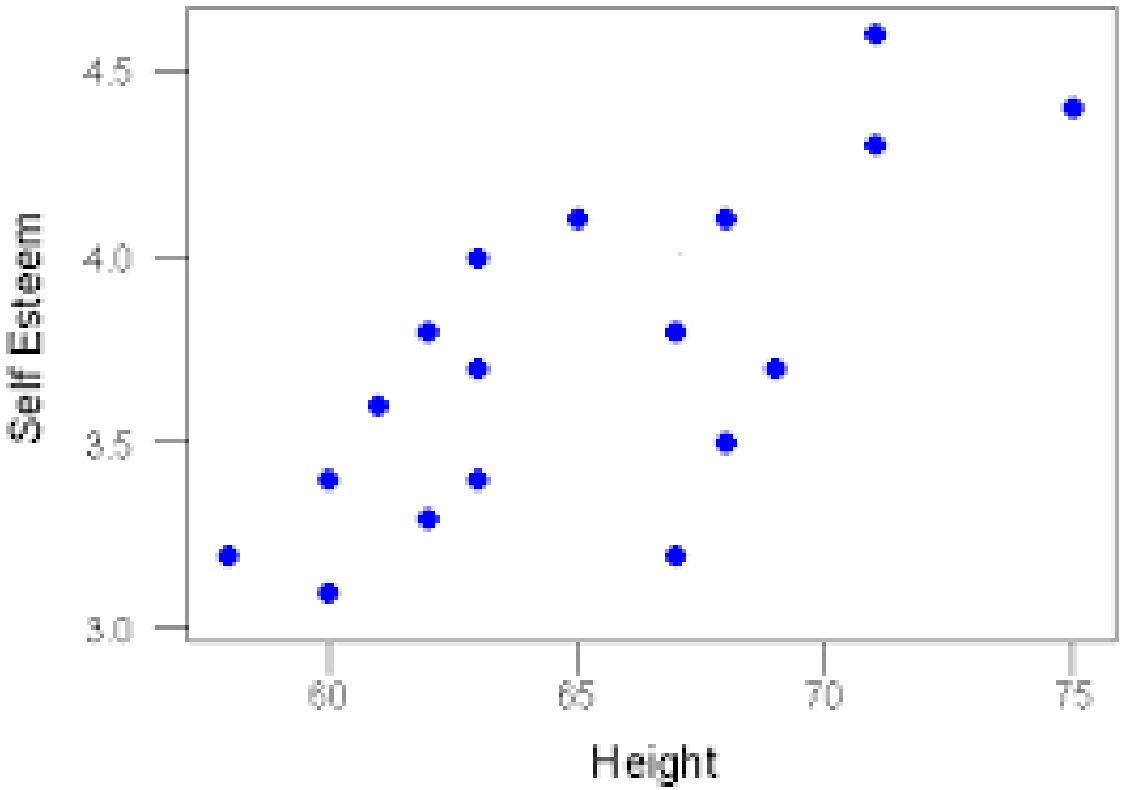
TECHNISCHE UNIVERSITÄT DARMSTADT

- We often use **accuracy** on the test data to evaluate our model performance:
  - How many instances are correctly classified divided by the number of instances in the test set

- However, there are other plausible evaluation measures:

  - E.g., when your outputs are continuous:
    - **Squared distance (MSE)**, **cosine**, **correlation**, etc.
  - Say, your output is a sequence:
    - Could use **edit distance**, for instance

Evaluation - Correlation

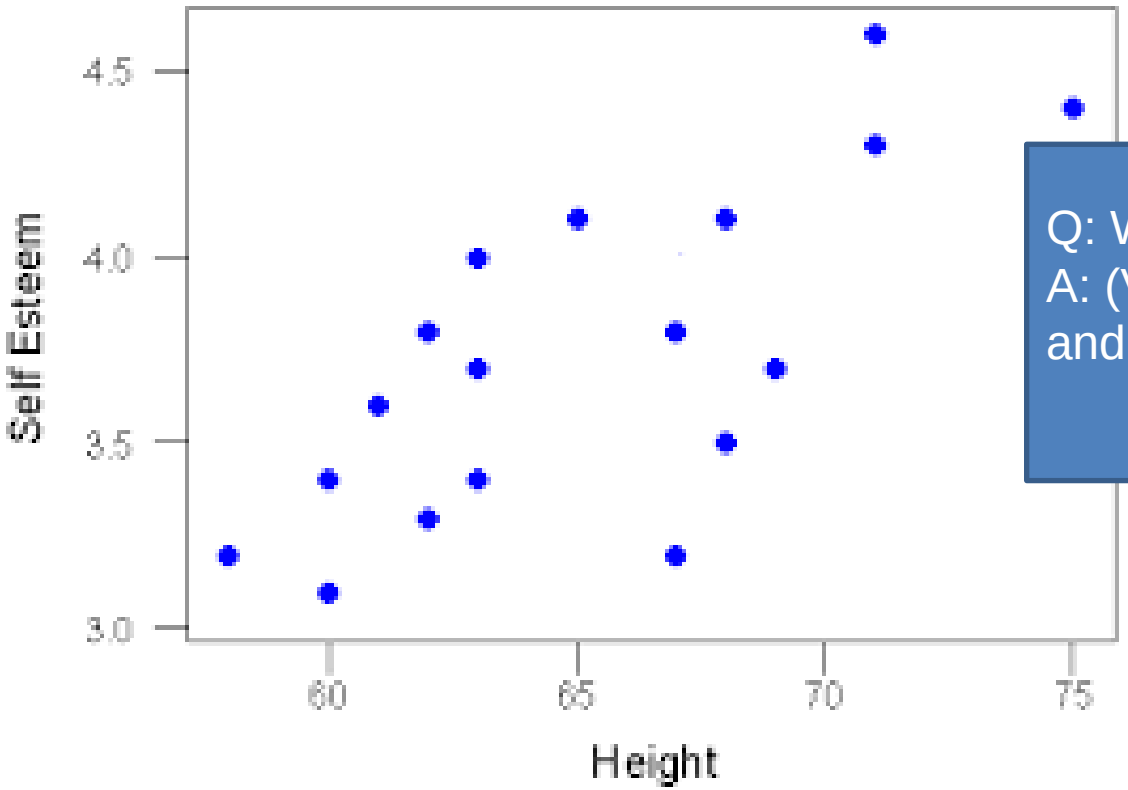



```
>>> from scipy.stats import pearsonr
>>> # try also spearmanr
>>> x=[1,2,3]; y=[-1,-4,6]
>>> pearsonr(x,y)
0.682
```

Measures linear relationship between two variables:
if one increases, does the other also increase (in expectation)?

Can use when your output is **continuous** (real numbers) rather than discrete

Evaluation - Correlation



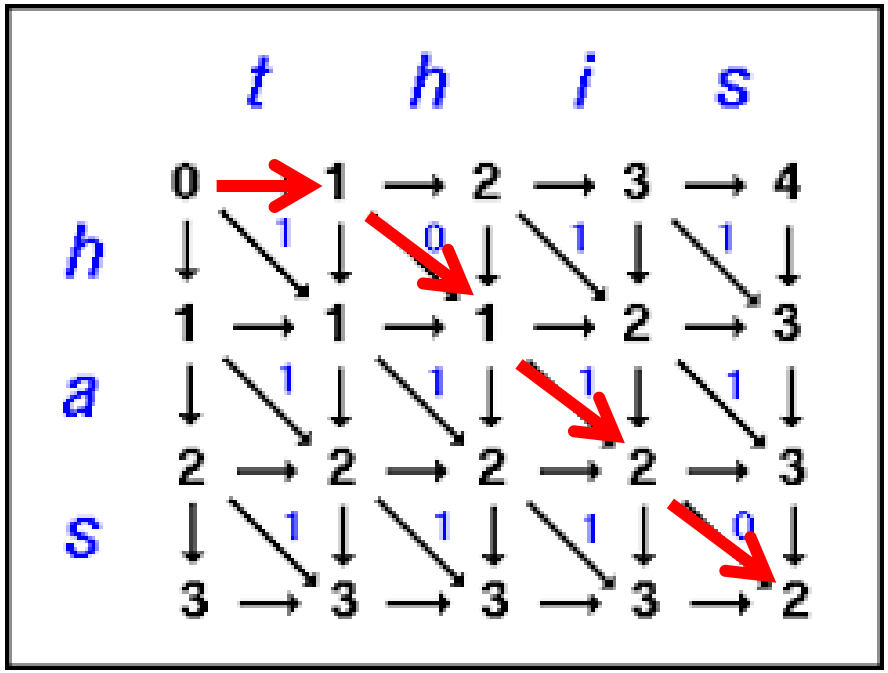
Q: What are we correlating?
A: (Vector of) model predictions and truth labels

Measures linear relationship between two variables:
if one increases, does the other also increase (in expectation)?

Can use when your output is **continuous** (real numbers) rather than discrete

Evaluation – Edit distance

- Minimal number of insertions, deletions and substitutions to transform one sequence into another




```
>>> import editdistance as ed
>>> ed.eval("this","has")
2
```

Can use when your output are **strings** (words, text, documents)

Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Even when your outputs are discrete classes, accuracy is sometimes not a good evaluation measure

- Say, you want to predict whether a patient has a rare disease Q

- Which system is better?

Terminology (precision/recall)

TECHNISCHE UNIVERSITÄT DARMSTADT

	Prediction is Q	Prediction is not Q
Patient has Q	True positive	False negative
Patient has not Q	False positive	True negative

Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- System A

	Prediction is Q	Prediction is not Q
Patient has Q	0	10
Patient has not Q	1	1004

- System B

	Prediction is Q	Prediction is not Q
Patient has Q	4	6
Patient has not Q	5	1000

- Both systems have an acc. of 1004/1015 = 99%

Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

For Class „Disease“

- System B has a *precision* of
  4/9 = 44%
- and a *recall* of
  4/10 = 40%

- System A has precision of 0% and recall of 0%

For Class „No Disease“, both systems are very close:

- Precision A: 1004/1014, Pr B: 1000/1006
- Recall A: 1004/1005, Rec B: 1000/1005

NLLG

Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- When there are more than two classes, precision and recall for class i are defined as
    - Precision_i = C(i,i)/sum(C(:,i))
    - Recall_i = C(i,i)/sum(C(i,:))

  where C is a *confusion matrix* as above

- From precision and recall, one can compute the „*F1 score*“:
    - The harmonic mean of P and R, defined as $2PR/(P+R)$

NLLG

Evaluation – F1 measure

TECHNISCHE UNIVERSITÄT DARMSTADT

- For two or more classes, one typically computes the F1-score of each class and then combines this in an overall score:
  - For example, averaging all the F1 scores
- There are different manners in which this can be done with different names
  - e.g. *micro F1* vs. *macro F1*

Can use when your output are **discrete classes** which are **imbalanced**

Evaluation – Others

TECHNISCHE UNIVERSITÄT DARMSTADT

- There are many more evaluation measures
  - For Machine Translation (MT): BLEU scores,…
  - For Summarization: ROUGE (n-gram overlap),….
  - Etc.
- Choosing which evaluation measure is an import field of research
- Importantly: For higher level NLP tasks, automatic measures may correlate poorly with human evaluation
  → need for new measures

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Loss functions

Preliminary

TECHNISCHE UNIVERSITÄT DARMSTADT

- In the remainder, we'll call
  - The true labels $\boldsymbol{t} = (t_1, \ldots, t_m)$ ("truth")
  - Our network's predictions $\boldsymbol{y} = (y_1, \ldots, y_m)$
  - Note that we have m output units – indexed by j
  - As before the number of samples is N:

$$\{(\boldsymbol{x}_1, \boldsymbol{t}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{t}_N)\}$$

    - We index them by i

Square loss

TECHNISCHE UNIVERSITÄT DARMSTADT

- We had said before that our goal in ML is to solve (a variant of)

$$\sum_{(\boldsymbol{x},t)\in T} (f_{\boldsymbol{\theta}}(\boldsymbol{x}) - t)^2$$

where $f_{\boldsymbol{\theta}}$ was our model, parametrized by $\boldsymbol{\theta}$

- What we optimized there was the so-called **square loss**

$$\ell(t, y) = (y - t)^2$$

- Multi-dimensional square loss would look as follows:

$$\ell(\boldsymbol{t}, \boldsymbol{y}) = \sum_j \left(y_j - t_j\right)^2 = \left|\left|\boldsymbol{y} - \boldsymbol{t}\right|\right|^2$$

Types of loss functions

TECHNISCHE UNIVERSITÄT DARMSTADT

- There are other loss functions commonly used in machine learning such as
  - **0-1 loss**: $\ell(\boldsymbol{t}, \boldsymbol{y}) = \begin{cases} 1, & \text{if } \boldsymbol{t} \neq \boldsymbol{y} \\ 0, & \text{if } \boldsymbol{t} = \boldsymbol{y} \end{cases}$
  - **Multi-dim Hinge loss:** $\ell(\boldsymbol{t}, \boldsymbol{y}) = \sum_j \max(0, y_j - y_t + 1)$
    - where t is the index where $\boldsymbol{t}_t = 1$
  - **Cross-entropy loss**
    - $\ell(\boldsymbol{t}, \boldsymbol{y}) = -\sum_j t_j \log(y_j)$ (Minimum value is achieved when $\mathbf{t} = \boldsymbol{y}$. In this case $\ell(\boldsymbol{t}, \boldsymbol{y}) = H(\boldsymbol{y})$, the entropy of **y**)

NLLG

Examples

TECHNISCHE UNIVERSITÄT DARMSTADT

- Suppose that for an input $\boldsymbol{x}$
  - $\boldsymbol{t} = (0,1,0,0)$
  - $\boldsymbol{y} = (0.25,0.3,0.4,0.05)$

NLLG

Examples

TECHNISCHE UNIVERSITÄT DARMSTADT

- Square loss is

$$\boldsymbol{t} = (0,1,0,0)$$
$$\boldsymbol{y} = (0.25,0.3,0.4,0.05)$$

Examples

- Square loss is
  - $0.25^2 + 0.7^2 + 0.4^2 + 0.05^2$

$$\boldsymbol{t} = (0,1,0,0)$$
$$\boldsymbol{y} = (0.25,0.3,0.4,0.05)$$

Examples

- 0-1 loss is

$$\boldsymbol{t} = (0,1,0,0)$$
$$\boldsymbol{y} = (0.25,0.3,0.4,0.05)$$

Examples

- 0-1 loss is
  - 1

$$\boldsymbol{t} = (0,1,0,0)$$
$$\boldsymbol{y} = (0.25,0.3,0.4,0.05)$$

Examples

- Multi-dim Hinge Loss is

$$\boldsymbol{t} = (0,1,0,0)$$
$$\boldsymbol{y} = (0.25,0.3,0.4,0.05)$$

$$\ell(\boldsymbol{t}, \boldsymbol{y}) = \sum_j \max(0, y_j - y_t + 1)$$

Examples

- Multi-dim Hinge Loss is

$$\boldsymbol{t} = (0,1,0,0)$$
$$\boldsymbol{y} = (0.25,0.3,0.4,0.05)$$

  - $(0.25 - 0.3 + 1) + (0.3 - 0.3 + 1)$
    $+(0.4 - 0.3 + 1) + (0.05 - 0.3 + 1)$

$$\ell(\boldsymbol{t}, \boldsymbol{y}) = \sum_j \max(0, y_j - y_t + 1)$$

Whenever $y_j - y_t + 1 \leq 0$
$\leftrightarrow \; y_j \leq y_t - 1$
we occur no loss for class j. The constant 1 is the „margin“

Examples

TECHNISCHE UNIVERSITÄT DARMSTADT

- Cross-Entropy loss is

$$\boldsymbol{t} = (0,1,0,0)$$
$$\boldsymbol{y} = (0.25,0.3,0.4,0.05)$$

$$\ell(\boldsymbol{t}, \boldsymbol{y}) = -\sum_j t_j \log(y_j)$$

Examples

TECHNISCHE UNIVERSITÄT DARMSTADT

- Cross-Entropy loss is

$$\boldsymbol{t} = (0,1,0,0)$$
$$\boldsymbol{y} = (0.25,0.3,0.4,0.05)$$

- $-\log(0.3)$

$$\ell(\boldsymbol{t}, \boldsymbol{y}) = -\sum_j t_j \log(y_j)$$

Training data loss

- Loss over whole training data is the sum over loss for each example
  - $L = \sum \ell_i$
    - where $\ell_i = \ell(\boldsymbol{t}_i, \boldsymbol{y}_i)$

Cross-entropy loss

TECHNISCHE UNIVERSITÄT DARMSTADT

- The 'natural' loss for softmax is cross-entropy
  - $\ell(\boldsymbol{t}, \boldsymbol{y}) = -\sum_j t_j \log(y_j)$
    - $\boldsymbol{t} = (t_1, \ldots, t_m)$ is the target output (distribution)
    - $\boldsymbol{y} = (y_1, \ldots, y_m)$ is the network prediction (distribution)

- Cross-entropy $H(\boldsymbol{p}, \boldsymbol{q}) = -\sum_x p(x) \log q(x)$
  - is (related to) a measure of distance between two (discrete) probability distributions
    - CE is minimized when $\boldsymbol{p} = \boldsymbol{q}$
      - But it's not zero then, but $H(\boldsymbol{p})$, the *entropy* of distribution $\boldsymbol{p}$
  - It's not symmetric, $H(\boldsymbol{p}, \boldsymbol{q}) \neq H(\boldsymbol{q}, \boldsymbol{p})$
  - $H(\boldsymbol{p}, \boldsymbol{q}) = H(\boldsymbol{p}) + \mathrm{KL}(\boldsymbol{p}, \boldsymbol{q})$
    - where KL is the Kullback-Leibler divergence

Square loss vs. CE loss

TECHNISCHE UNIVERSITÄT DARMSTADT

- Is square loss a good loss function for multi-class prediction?

- Consider
  - $\boldsymbol{t} = (1,0)$
  - $\boldsymbol{y} = (\epsilon, 1-\epsilon)$

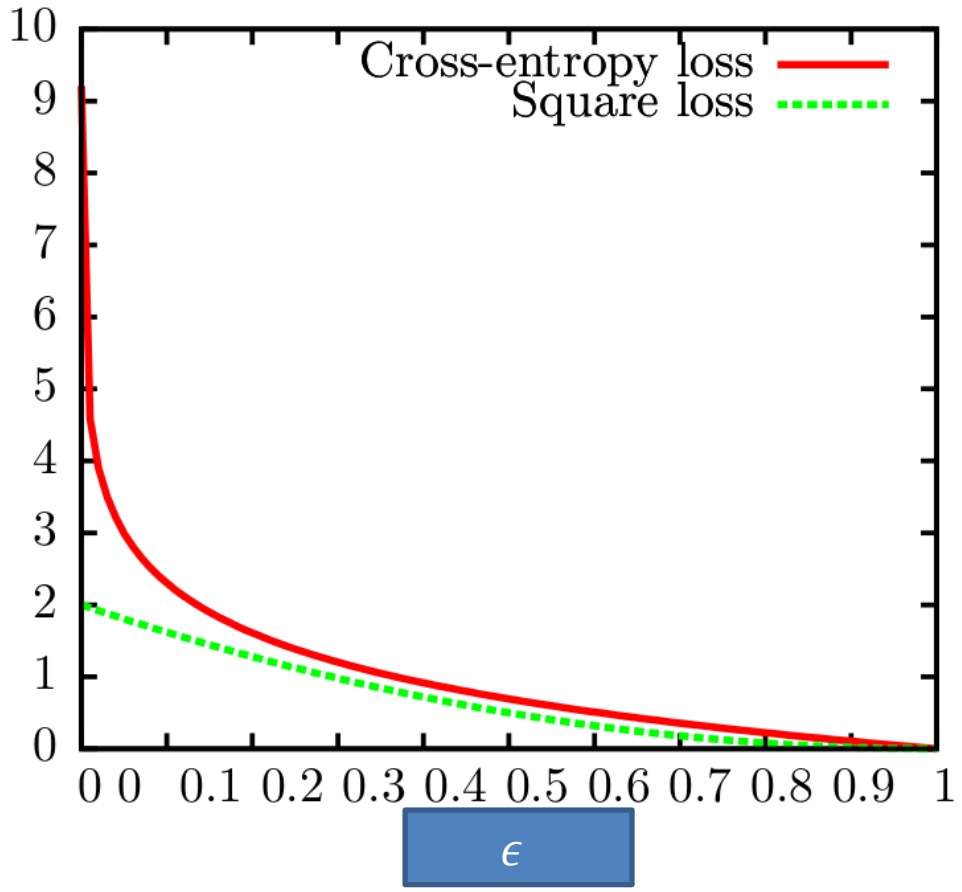
- Square loss $2(1-\epsilon)^2$
- Cross entropy loss $-\log \epsilon$

Square loss vs. CE loss

- Is square loss a good loss function for multi-class prediction?

- Consider
  - $\boldsymbol{t} = (1,0)$
  - $\boldsymbol{y} = (\epsilon, 1-\epsilon)$

- Square loss $2(1-\epsilon)^2$
- Cross entropy loss $-\log \epsilon$



Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Foundations of ML
  - Train vs. Dev vs. Test Set
  - (Expected) Loss function optimization
  - Evaluation Measures

- Cross-Entropy vs Square Loss for Neural Nets

- General/Advice:
  - In class, we use square loss or (more often) cross-entropy loss
  - Output layer typically has softmax activation function (multi-class classification) → in this case, always choose CE loss
  - Loss function and evaluation measure go together, i.e., one may change loss when another evaluation metric is used

References

- Maas et al. 2013, Rectifier non-linearities improve neural network acoustic models. In ICML, vol. 30
- He et al., 2015, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification
- Clevert et al., 2015, Fast and Accurate Deep Network Learning by Exponential Linear Units
- Goodfellow et al., 2013, Maxout Networks
- Glorot and Bengio, 2010, Understanding the difficulty of training deep feedforward neural networks
- Ioffe and Szegedy, 2015, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift