Deep Learning for NLP

Lecture 2 – ML principles

Universität Bielefeld

Dr. Steffen Eger

seminar.dldh@gmail.com

CITEC NLLG

Previous and upcoming lectures

Universität Bielefeld

- Previous lecture:
  - (Historical) Survey of Deep Learning
- Next lecture:
  - Training Networks with Hidden Layers (Backpropagation algorithm)
  - Language Model

This lecture:

Universität Bielefeld

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

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

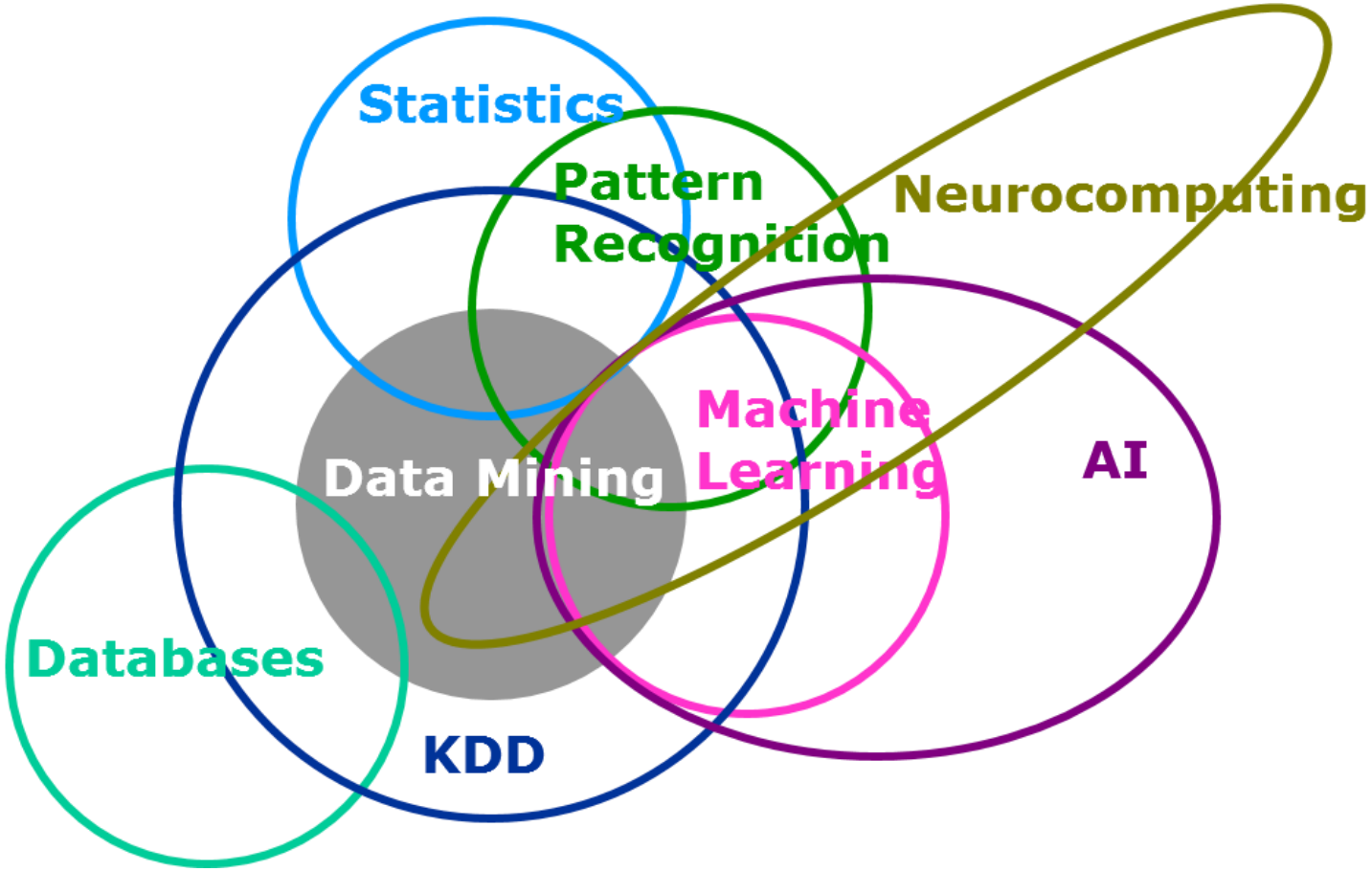
Outline

Universität Bielefeld

ML principles

Machine Learning

Universität Bielefeld



Standard Setup (Supervised setting)

Universität Bielefeld

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

Standard Setup

Universität Bielefeld

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

Universität Bielefeld

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**)

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

Generalization

Universität Bielefeld

- The (real) goal in ML is to **generalize** particularly well to unseen data

- Hence, our (real) goal is more to minimize the expected loss rather than the actually observed loss:

$$E\left[\sum\left(y - f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)^{\mathbf{2}}\right] \qquad (*)$$

where the expectation is over the distribution of datapoints: $(\boldsymbol{x}, y) \sim D$

Generalization

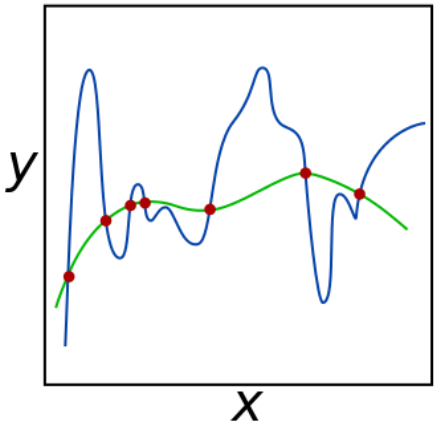
Universität Bielefeld

- Because we‘re interested in generalization performance, **we always split our data**:
  - Test data vs. Train data
  - Test data represents the true underlying distribution
- Model must perform well on the test data; performance on the training data is not of interest

Regularization

Universität Bielefeld

- To achieve generalization, one usually introduces *regularization*
  - a term added to the loss function that constrains the parameters to be small, for example
  - Intuition: use a smaller polynomial to fit your data (less expressive function) if possible



Training data

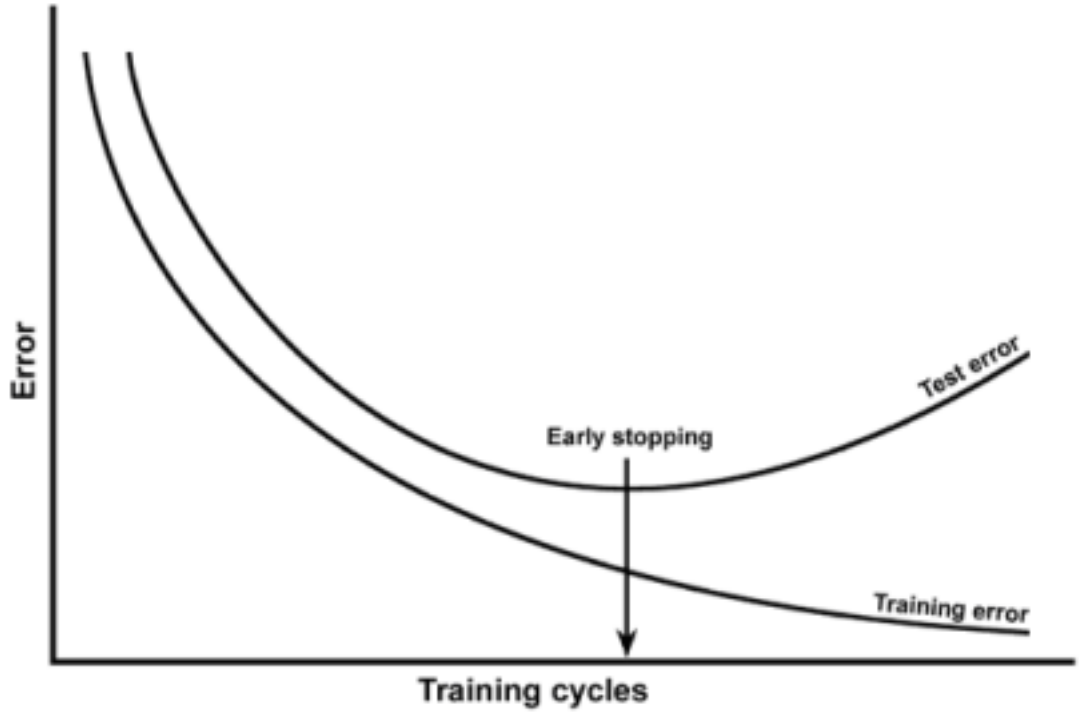
Universität Bielefeld

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

Training data

Universität Bielefeld

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



Test data

Universität Bielefeld

- Could choose some fraction of your overall data for testing, e.g., 75%/25% split (train/test) or 90%/10%

Quizz

Universität Bielefeld

Why can a 90%-10% train/test split be dangerous?

A: Too little test data
B: Too much train data
C: May accidentally pick hard test cases
D: May accidentally pick easy test cases
E: Performance estimate is unreliable

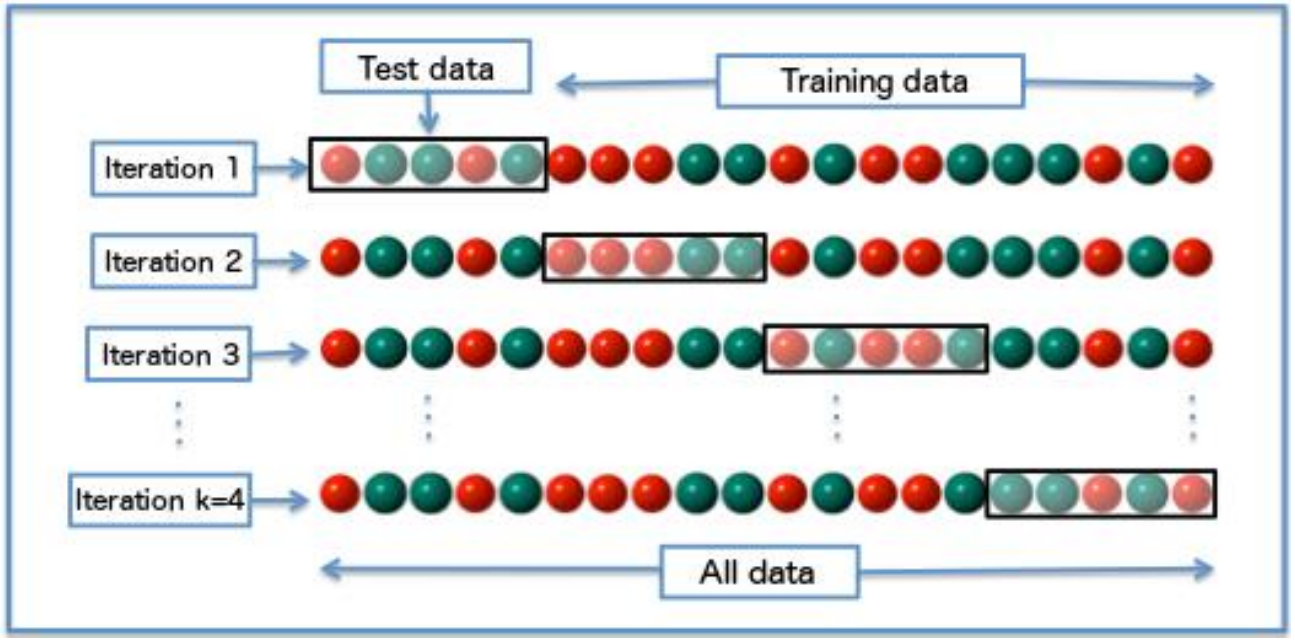
Test data

Universität Bielefeld

- Could choose some fraction of your overall data for testing, e.g., 75%/25% split (train/test) or 90%/10%

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

Test data


Test data
Training data
Iteration 1
Iteration 2
Iteration 3
Iteration k=4
All data

Test data

Universität Bielefeld

- This way, estimate of true performance is much more unbiased
- However, also computationally more costly

- Commonly used methods: **10-fold cross-validation, leave-one-out-validation**
- Other strategies:
  - e.g. *random subsample validation* in which test folds are randomly selected

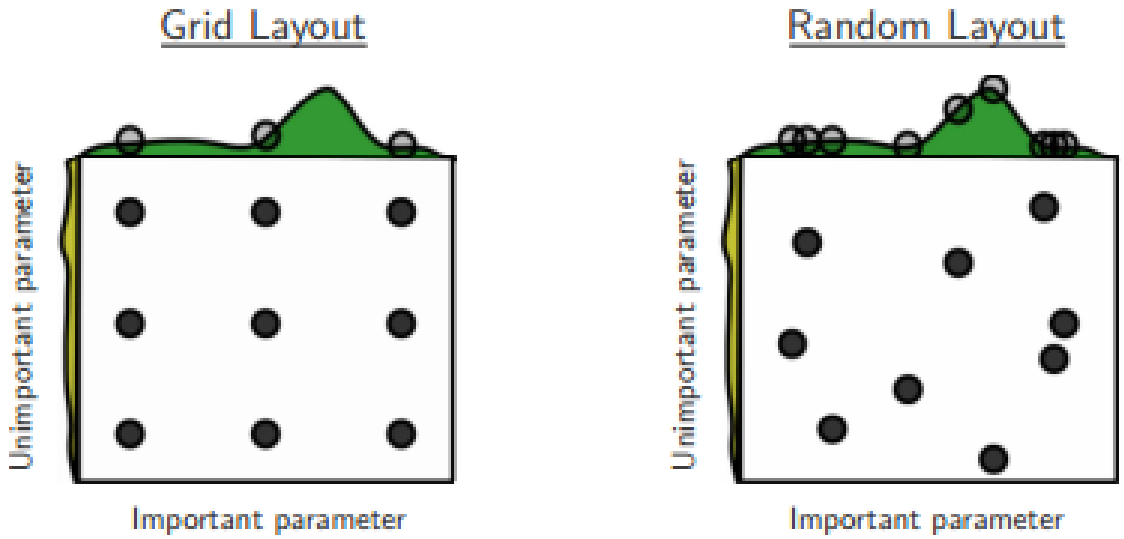
Hyperparameter search

Universität Bielefeld

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

Hyperparameter search

Universität Bielefeld



CITEC NLLG

Hyperparameter search

Universität Bielefeld

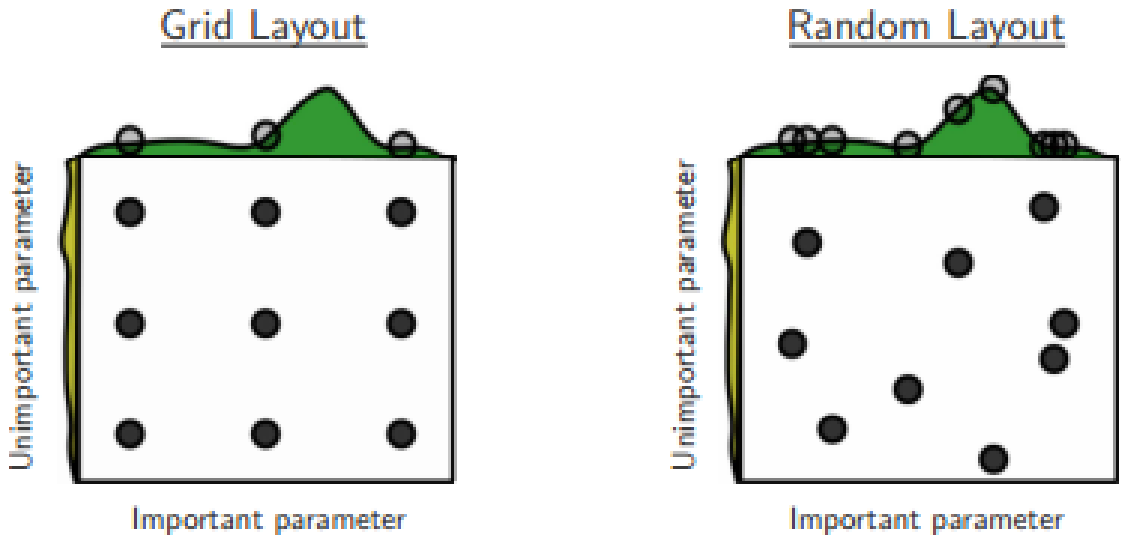


Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct 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.

Outline

Universität Bielefeld

Loss functions

Preliminary

Universität Bielefeld

In the remainder, we'll call

- The true labels $\boldsymbol{t} = (t_1, \ldots, t_m)$ ("truth")
- Our network's predictions $\boldsymbol{y} = (y_1, \ldots, y_m)$

Square loss

Universität Bielefeld

- 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

Universität Bielefeld

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 $\boldsymbol{t}_t = 1$

Let's assume t is a **one-hot vector** i.e., t = [0,0,…,1,0,..0,0]

- **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**)

Examples

Universität Bielefeld

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

Let's assume t is a **one-hot vector**
i.e., t = [0,0,…,1,0,..0,0]
→ multi-label classification

Examples

Universität Bielefeld

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

Examples

Universität Bielefeld

- Square loss is

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

Examples

Universität Bielefeld

- 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

Universität Bielefeld

- 0-1 loss is

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

Examples

Universität Bielefeld

- 0-1 loss is
  - 1

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

Examples

Universität Bielefeld

- 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

Universität Bielefeld

- 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

Universität Bielefeld

- 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

Universität Bielefeld

- 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

Universität Bielefeld

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

Universität Bielefeld

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 loss

Universität Bielefeld

- 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

Universität Bielefeld

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

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

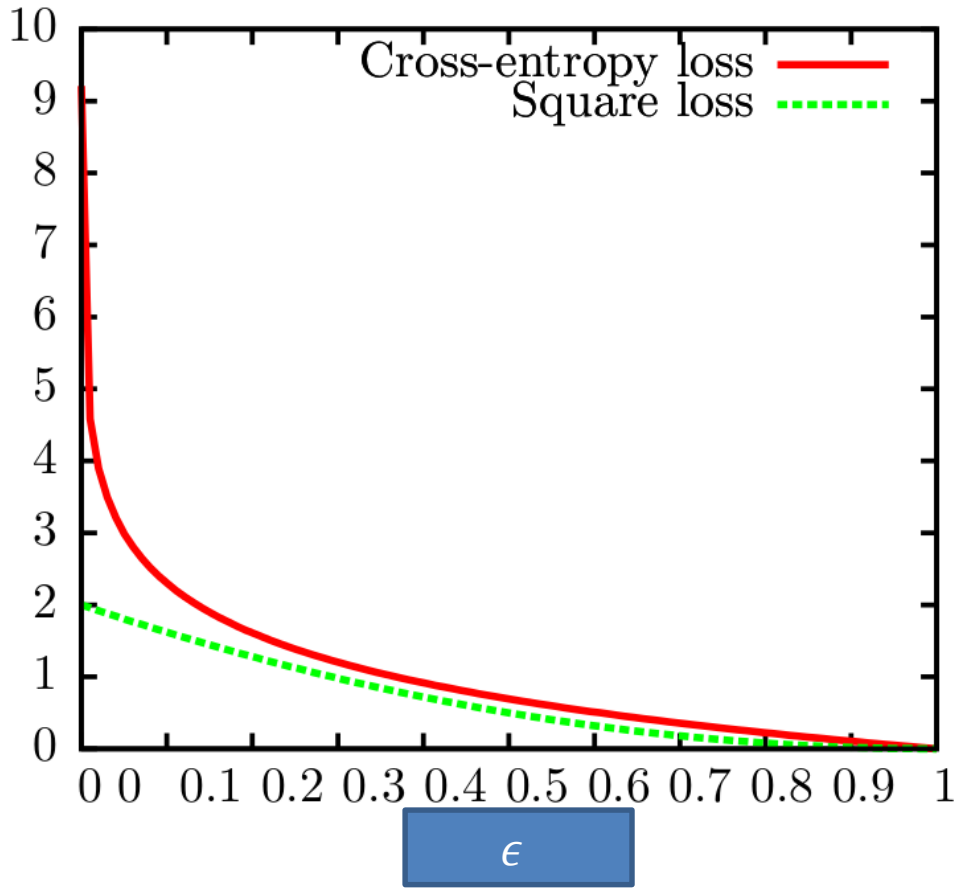
- Square loss $2(1-\epsilon)^2$
- Cross entropy loss $-\log \epsilon$

Square loss vs. CE loss

Universität Bielefeld

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 $-\log \epsilon$



CITEC NLLG

Outline

Universität Bielefeld

Evaluation

Evaluation

Universität Bielefeld

- We often use **accuracy**:
  - How many instances are correctly classified divided by the number of instances in the test set

- Btw. Assume our classifier is a DL model with softmax in the last layer to perform multi-class classification
  - How to compute the accuracy? i.e.,
  - How do we get a decision from the probability distribution?

DISCUSS

Universität Bielefeld

Why are we not evaluating models using loss functions directly?

Evaluation – Beyond accuracy

Universität Bielefeld

- When data is imbalanced:
  - F1
- When outputs are continuous:
  - E.g., **Squared distance (MSE)**, **correlation**, etc.
- When output is a sequence:
  - E.g., **edit distance**, **BLEU**, etc.

Evaluation – F1

Universität Bielefeld

- 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?

Evaluation

Universität Bielefeld

- 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

Universität Bielefeld

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

Evaluation

Universität Bielefeld

- 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:

  $$2PR/(P+R)$$

Quizz

Universität Bielefeld

Pred

True

	Politics	Sports	Religion
Politics	2	4	3
Sports	1	9	5
Religion	6	8	9

What is the recall of class „Sports“?

- *A: 9/17*
- *B: 9/21*
- *C: 9/15*
- *D: 20/42*

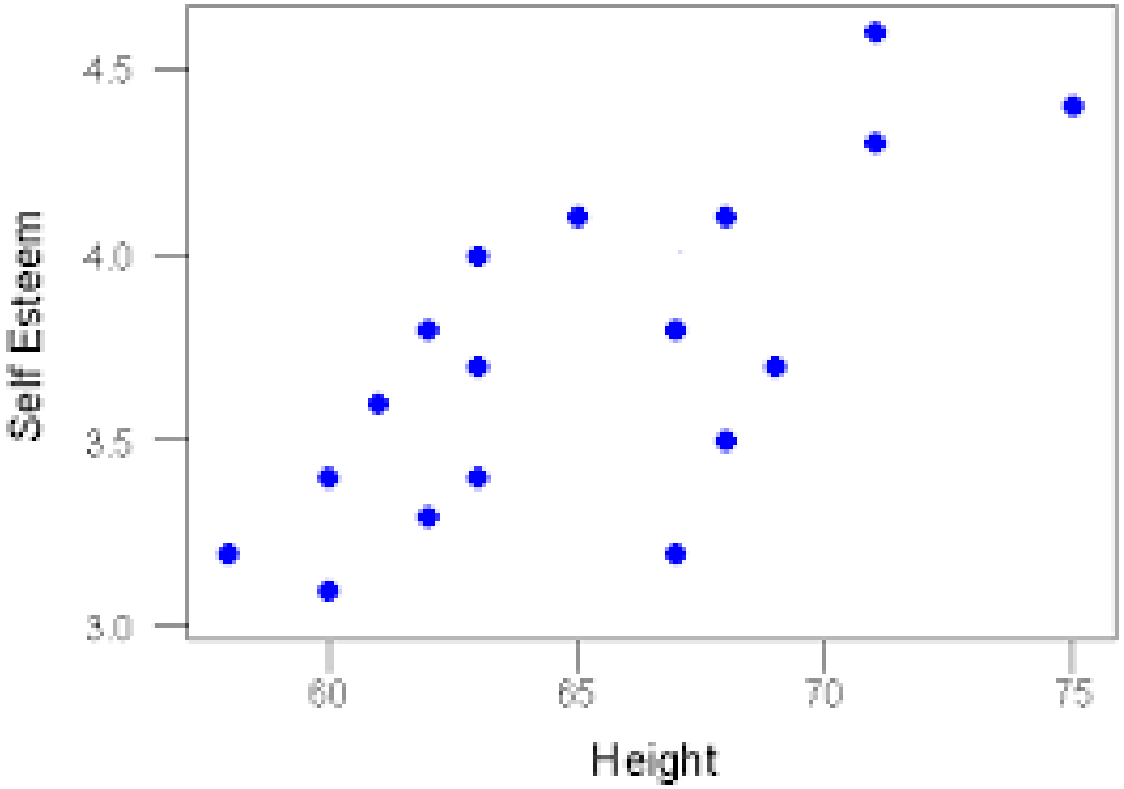
Evaluation – F1 measure

Universität Bielefeld

- 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
    - This is called *macro F1*

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

Evaluation - Correlation

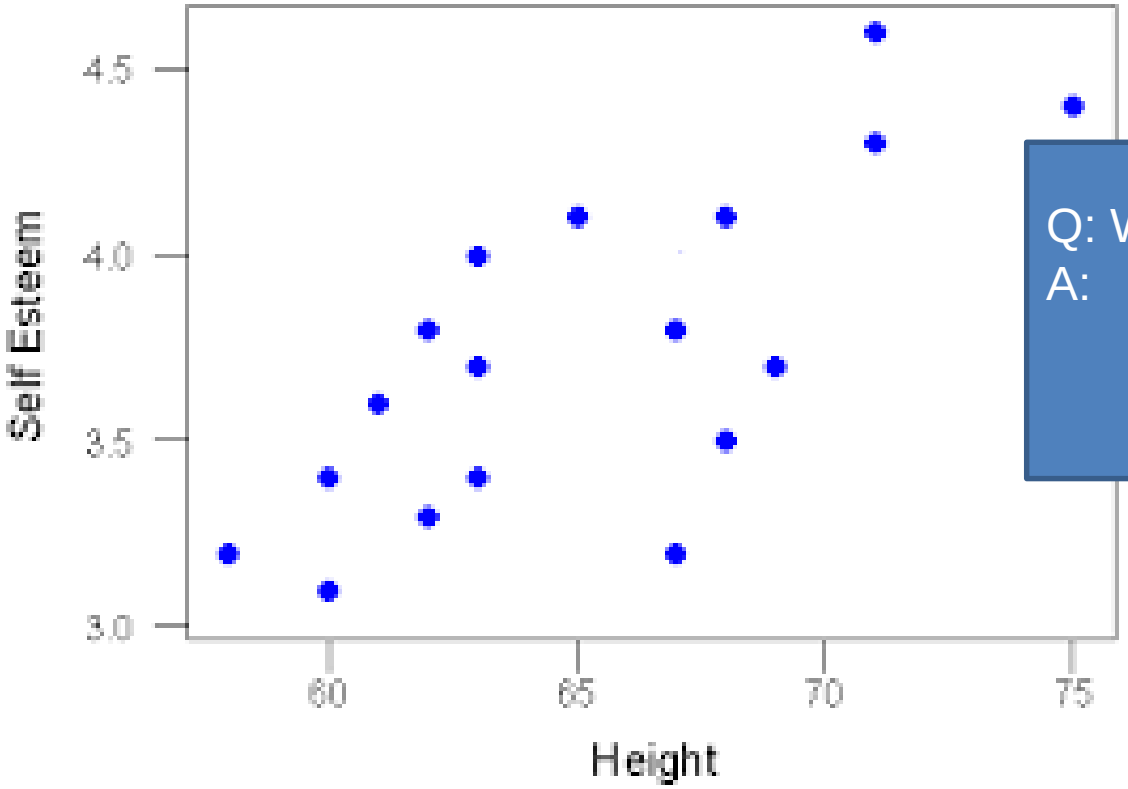



```
>>> from scipy.stats import pearsonr
>>> # try also spearmanr
>>> x=[1,2,3]; y=[-1,-4,6]
>>> pearsonr(x,y)
0.682
```

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

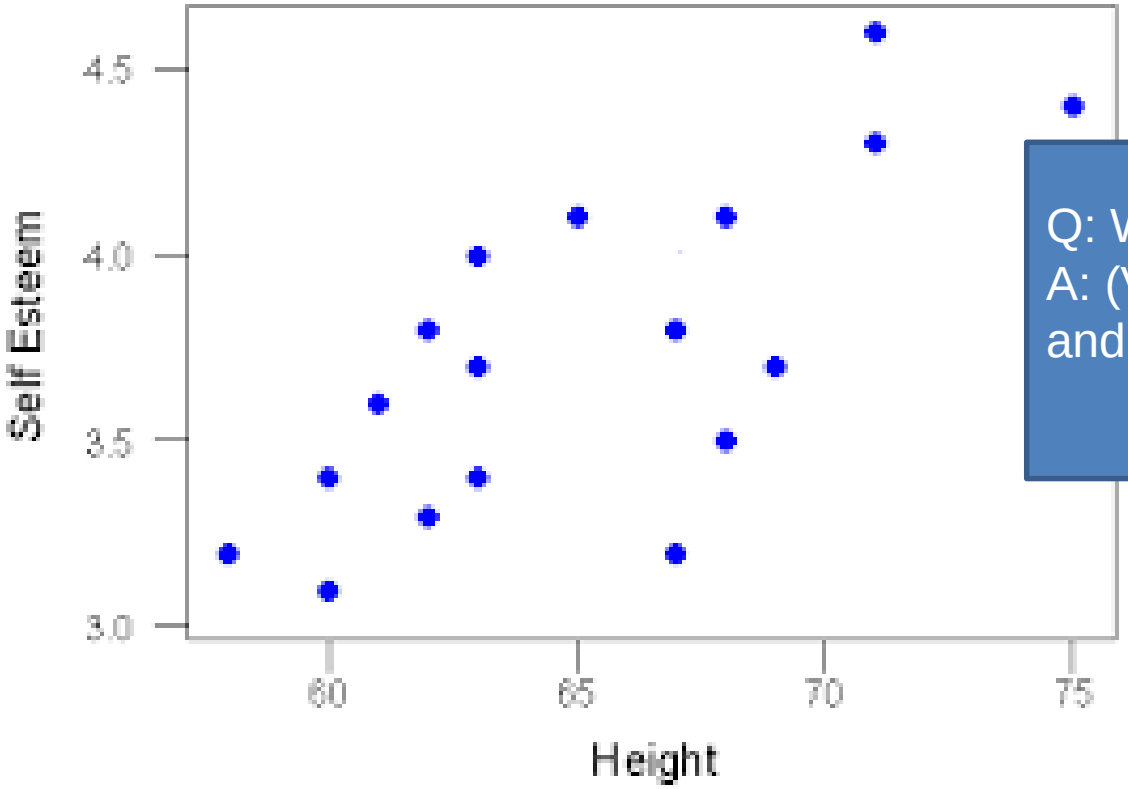


Q: What are we correlating?
A:

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

DISCUSS

Universität Bielefeld

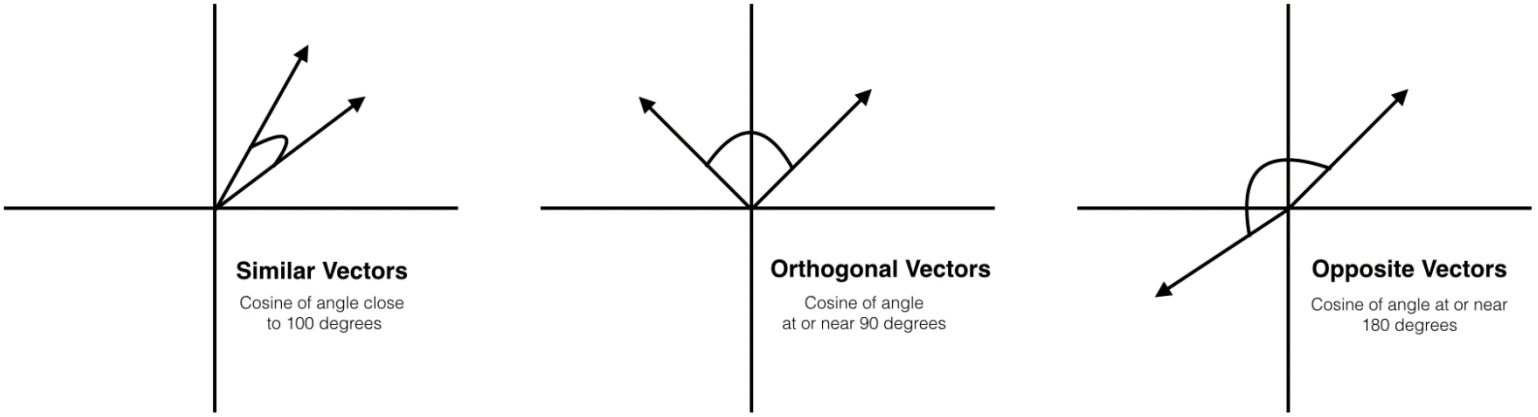
When would you prefer correlation over MSE?
(and vice versa)

Evaluation – Cosine Similarity

Universität Bielefeld

For two vectors $\boldsymbol{x}, \boldsymbol{y}$, their *cosine similarity* is:

$$\frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{y}||} \in [-1,1]$$



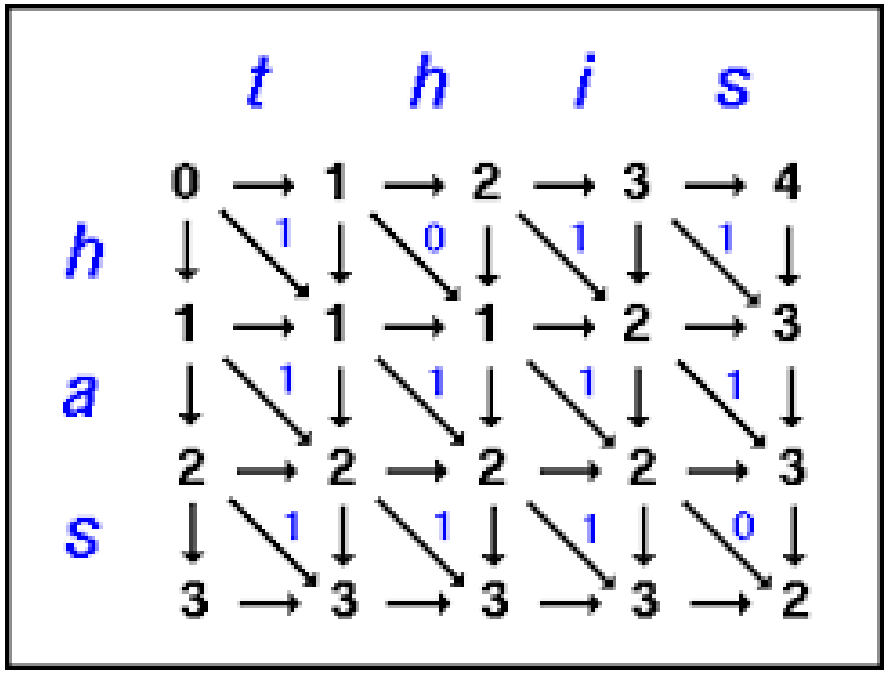
Can use when your output is a **vector**

CITEC NLLG

Evaluation – Edit distance

Universität Bielefeld

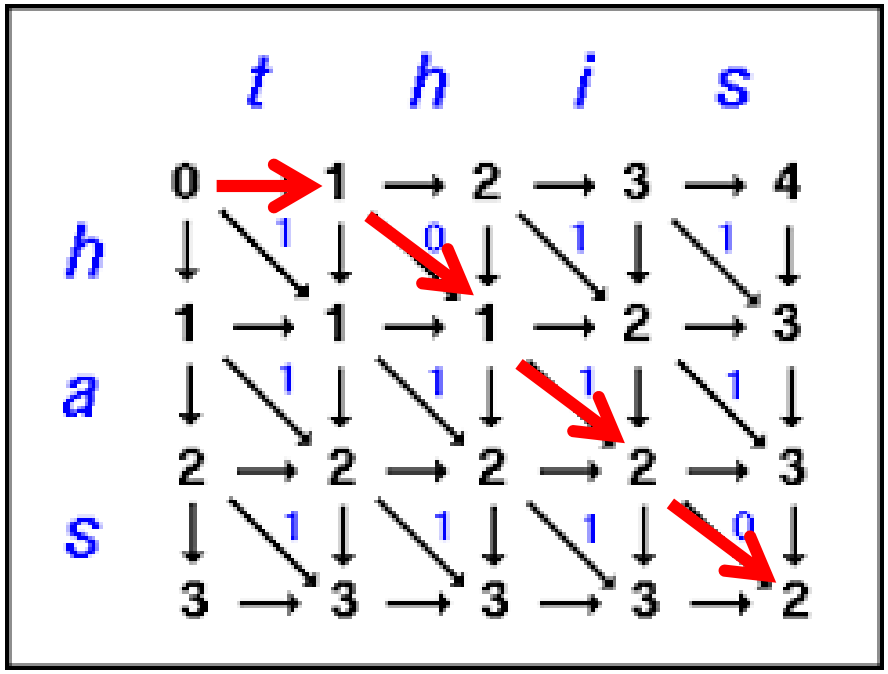
- Minimal number of insertions, deletions and substitutions to transform one sequence into another



Evaluation – Edit distance

Universität Bielefeld

- 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)

CITEC NLLG

Evaluation – Others

Universität Bielefeld

- 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 important field of research
- For higher level NLP tasks, some automatic measures may correlate poorly with human evaluation → need for new measures

DISCUSS

Universität Bielefeld

How would you evaluate machine translation and why is it difficult?

Summary

Universität Bielefeld

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

- Cross-Entropy vs Square Loss for Neural Nets

Summary

Universität Bielefeld

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