Automatic evaluation

- Use a test set that is already labeled
- Hide the existing labels, let the model predict new labels, and compare the two
- For each test instance, we have a **gold label** and a **predicted label**

Accuracy

Accuracy simply refers to the proportion of matching labels (gold label = predicted label):

$$\frac{\text{number of instances with matching labels}}{\text{total number of instances}}$$

Accuracy

- Example: A model that predicts whether a given message is spam or not.
- 1685 test instances, distributed as follows:

	Gold label YES (1)	Gold label NO (0)
System prediction YES (1)	83	22
System prediction NO (0)	41	1539

- Accuracy: $\frac{83+1539}{83+22+41+1539} = \frac{1622}{1685} = 96.26\%$

Contingency tables

- This is a **contingency table** (or **confusion matrix**):

	Gold label YES (1)	Gold label NO (0)
System prediction YES (1)	true positive	false positive
System prediction NO (0)	false negative	true negative

- These names are indeed a bit confusing…
  - positive = obtained a YES prediction
  - negative = obtained a NO prediction
  - true = prediction is true (predicted = gold label)
  - false = prediction is false (predicted ≠ gold label)

Precision and recall

	Gold label YES (1)	Gold label NO (0)
System prediction YES (1)	true positive	false positive
System prediction NO (0)	false negative	true negative

- $\text{Precision} = \frac{\text{true positives}}{\text{true positives}+\text{false positives}}$
- $\text{Recall} = \frac{\text{true positives}}{\text{true positives}+\text{false negatives}}$
- Intuition: We want our model to report "all the truth and nothing but the truth"

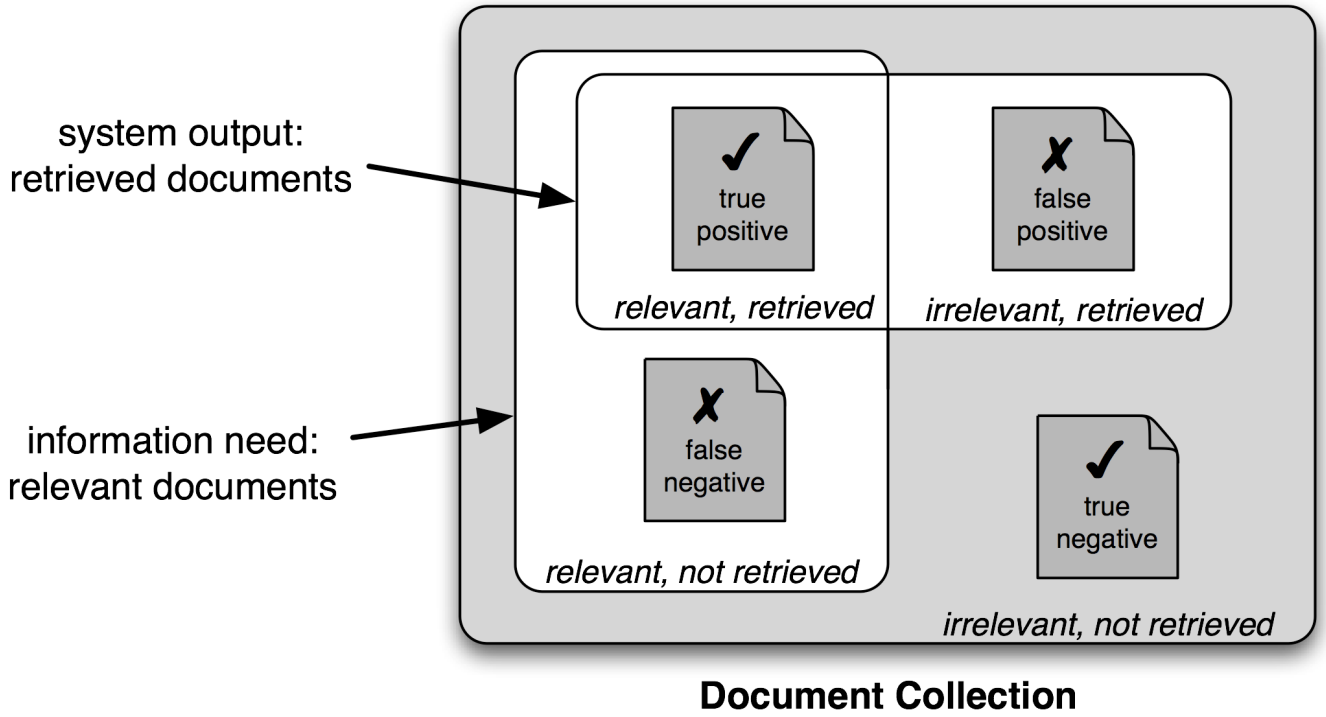
recall

precision

Intuition: Information retrieval

- Goal: Find all the documents on a particular topic out of 100 000 documents
  - Say there are 5
  - The system delivers 10 documents: 2 relevant and 8 irrelevant
  - What is the accuracy?
- For information retrieval tasks, we focus on:
  - The relevant documents
  - The documents returned by the system
- We do not care about:
  - The irrelevant documents which are not returned

Intuition: Information retrieval



Precision and recall

What about our spam classification model?

- Precision: $\frac{83}{83+22} = 79.0\%$
- Recall: $\frac{83}{83+41} = 66.9\%$

F-measure

- It is a bit inconvenient to have to report two figures, precision and recall...
- Can we combine them into one, take some “average”?
- **F-measure** is the harmonic mean of precision and recall
- **F_1-measure** is the (most popular) variant that weighs precision and recall equally:

$$F_1 = \frac{2 * P * R}{P + R}$$

Precision and recall

- Precision and recall are usually defined on the YES class, but they can also be defined on any other class

	Gold label YES (1)	Gold label NO (0)
System prediction YES (1)	83	22
System prediction NO (0)	41	1539

- Precision of NO: $\frac{1539}{41+1539}$
- Recall of NO: $\frac{1539}{22+1539}$

Multi-class settings

		gold labels urgent	normal	spam	
system output	urgent	8	10	1	$\text{precision}_u = \frac{8}{8+10+1}$
	normal	5	60	50	$\text{precision}_n = \frac{60}{5+60+50}$
	spam	3	30	200	$\text{precision}_s = \frac{200}{3+30+200}$
		$\text{recall}_u = \frac{8}{8+5+3}$	$\text{recall}_n = \frac{60}{10+60+30}$	$\text{recall}_s = \frac{200}{1+50+200}$	

Jurafsky & Martin 3rd edition, chapter 4.7

https://web.stanford.edu/~jurafsky/slp3/4.pdf

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$\text{precision} = \frac{8}{8+11} = .42$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$\text{precision} = \frac{60}{60+55} = .52$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$\text{precision} = \frac{200}{200+33} = .86$

$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$

Experimental setup

- We need training data for the training phase
  - Labeled instances
  - The model learns to associate instances with labels
- We need test data for the testing phase
  - Labeled instances
  - Hide the labels, let the model predict new labels, and compare the two to compute performance
- Training and test data must not overlap!

A three-way split

- In practice, you may want to experiment with several models, compare their quality, and select the best one
  - Different learning algorithms, different features, ...

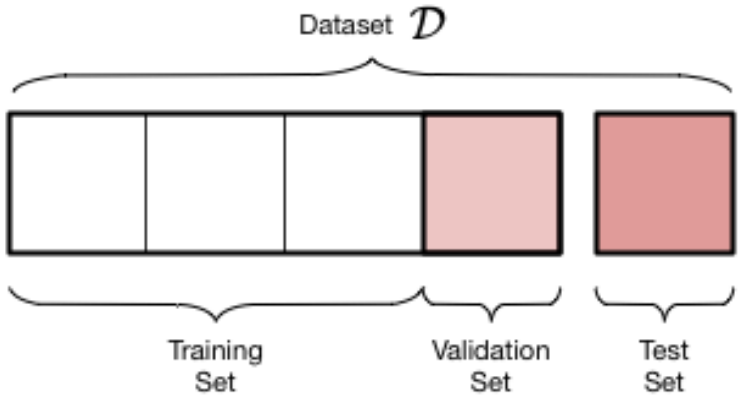
- We need two sets of held-out data:
  - development (test) set / validation set
  - test set

To select the best model setup

To evaluate the best model setup objectively

In practice

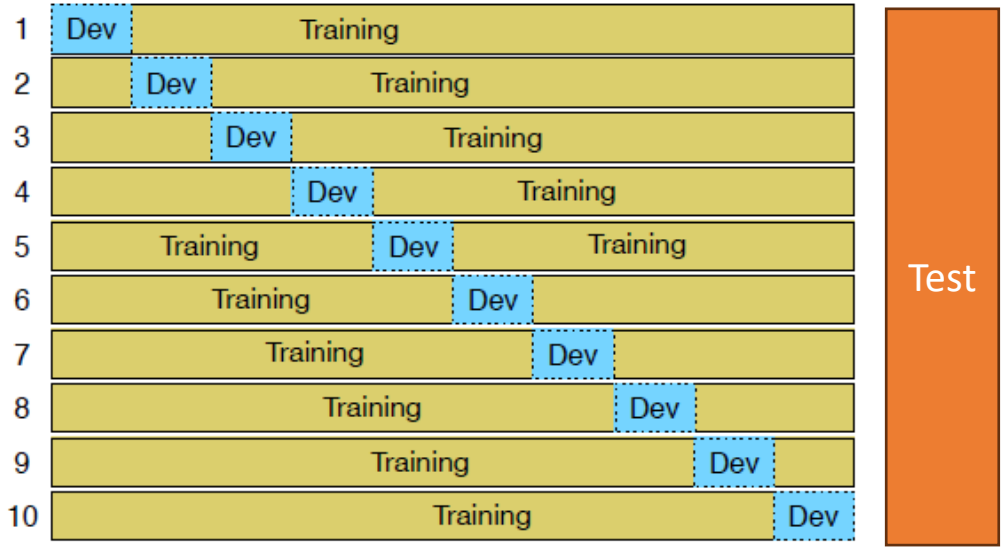
- Common splits:
  - 80% training,
    10% validation,
    10% test
  - 80% training,
    20% validation,
    test from different source
  - Shuffle before splitting



- What if this leaves you with too little data?
  - K-fold cross-validation
  - Leave-one-out cross-validation

K-fold cross-validation

- Train K models with different parts held out for validation
- Average validation scores
- Typical example: K=10



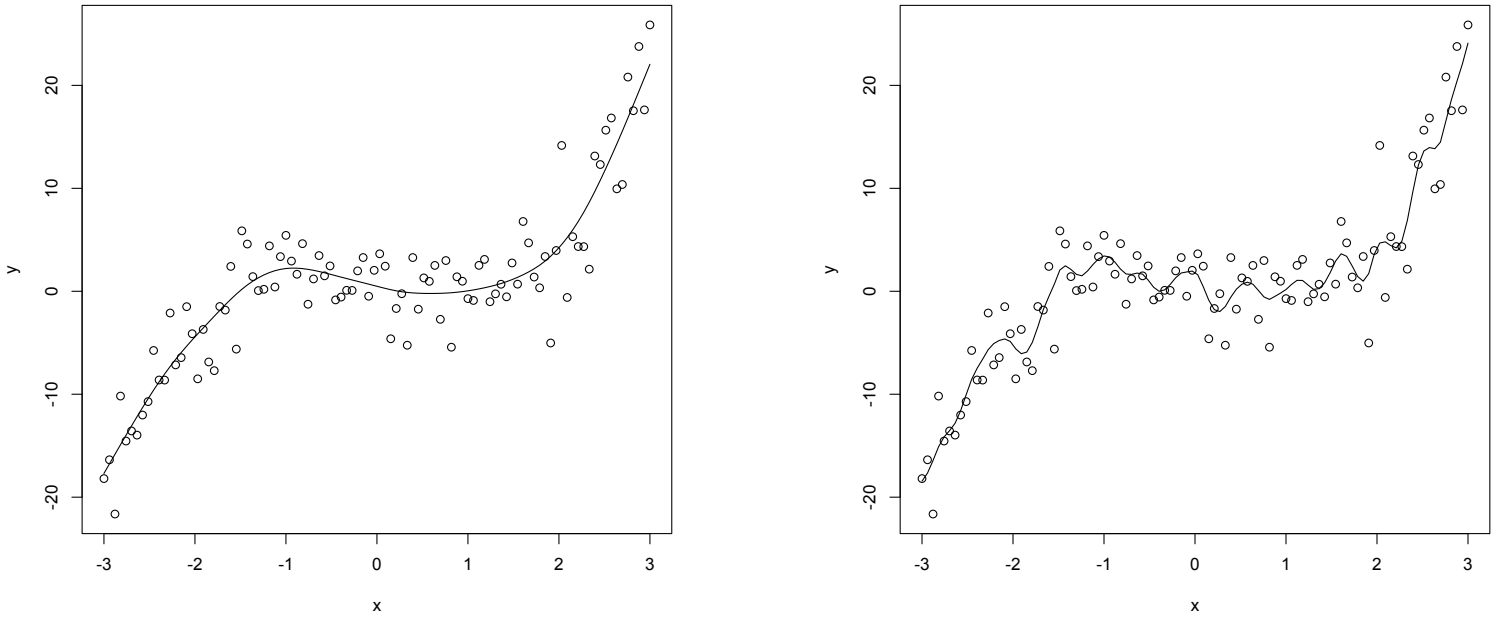
Leave-one-out cross-validation

- For each instance *i* in the dataset:
  - Train model on all instances except *i*
  - Validate model on *i*
- Average validation scores

- Requires training as many models as there are instances!
- This is only viable if the number of instances is small (~100).

How to detect overfitting?

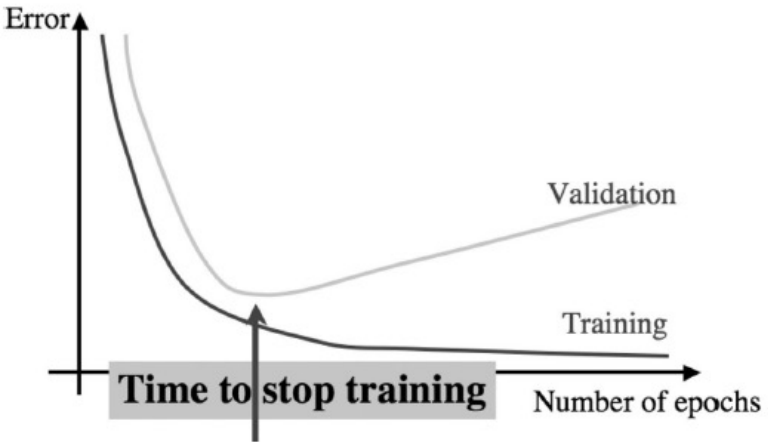
Two curve estimates for the same data set:



For this simple 2D problem it is visually clear that the right curve overfits the data.

Most machine learning problems involve higher-dimensional data. How can we detect overfitting in that case?

Early stopping



Principle:

Stop the learning when the error starts increasing on the validation data set

Idea:

When the error starts increasing on the validation data, we are likely overfitting to the training data

Regularization

- The lecture states that regularization reduces overfitting
- But all the regularizer does is punish large weights

- Why do smaller weights overfit less?
- How do I choose between different regularizers (Ridge, Lasso, Elastic Nets)?

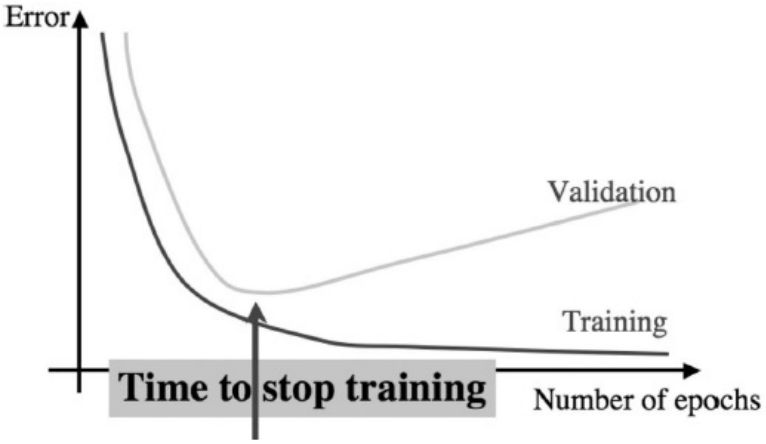
Regularization

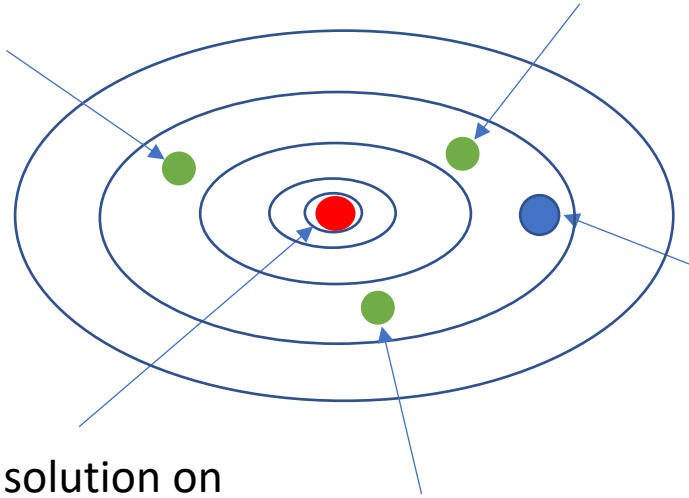
- Replace
  - $-Loss(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w})$ with
  - $-Loss(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w}) - \alpha R(\boldsymbol{w})$
- Where
  - $R(\boldsymbol{w})$ punishes large weights
  - α determines how hard they should be punished
  - α is to be tuned on a validation set

Regularization

- Large weights = complexity
- Large weights = ability to lower the loss by over-adapting
- We want a model which is as **simple** as possible, even if the loss is slightly higher
  - Because this model will generalize better
  - Also, in many cases, easier to train
- The choice between different regularizers should be informed by experiments, as usual
- Modern deep learning mostly uses **Dropout**.

Early stopping



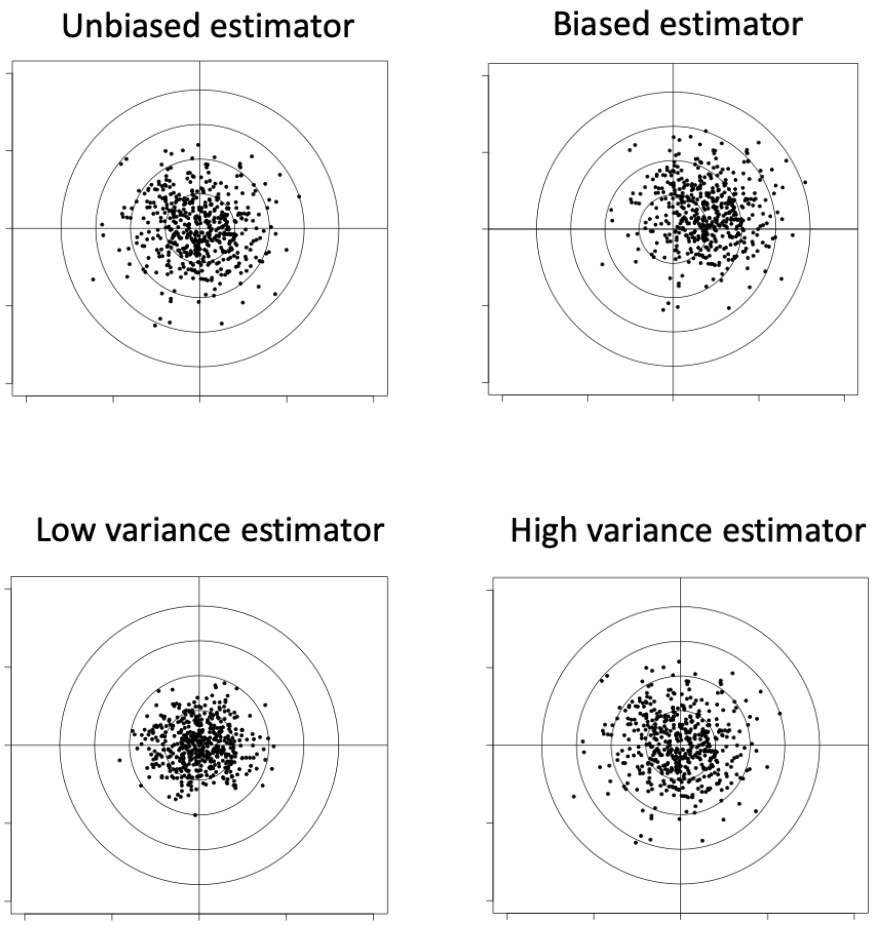


True solution (optimal solution on future data sets)

Optimal solution on training data set

- It seems early stopping is very dependent on the start state?
- Can early stopping give a worse result than not stopping early?

Bias and variance – what is most important to control?



High bias means the solutions on different data sets are not centred on the right target (the truth).

High variance means the solutions on different data sets have large spread.

- What factors influence bias?

- What factors influence variance?

- Why can't I have low bias and variance at the same time?

Ensemble classifiers

Key idea:

- To learn: Train many weak classifiers

- To predict: apply all the classifiers and decide the answer by a vote

Questions:

- What makes ensemble classifiers so effective? Any intuition?
- How many weak classifiers do we typically need?
- Would it be better to train many *strong* classifiers?

Ensemble classifiers

Questions:

- What makes ensemble classifiers so effective? Any intuition?
- How many weak classifiers do we typically need?
- Would it be better to train many *strong* classifiers?

Answers:

- We minimize the probability of missing an extremely lucky classifier or landing on an extremely unlucky one
- No clear answer, depends on the task
- Of course! But strong classifiers:
  - are more computationally expensive
  - might not be possible to train at all

Pros and cons of ANNs

The course introduces different machine learning methods, such as:

- k-nearest neighbor regression and classification
- linear regression and classification
- logistic regression and classification
- artificial neural networks

If we average performance over all imaginable problems, then no method is better than any other. So ANNs must be good for some problems and bad for others.

What type of problems are ANNs good for?

Any particular strong and weak sides of ANNs?



Other learning algorithms for ANNs

The purpose of backpropagation is to find weights that minimize a given cost function. It uses gradient descent (or a variant thereof).

There are many other optimization methods, for example:

- Hill climbing
- Simulated annealing
- Evolutionary algorithms
- Newton's method
- Conjugate gradient
- Quasi-Newton

Are these used, too? Are they better?

Other learning algorithms for ANNs

The purpose of backpropagation is to find weights that minimize a given cost function. It uses gradient descent (or a variant thereof).

There are many other optimization methods, for example:

- Hill climbing
- Simulated annealing
- Evolutionary algorithms
- Newton's method
- Conjugate gradient
- Quasi-Newton

Are these used, too? Are they better?

Short answer: no, almost never (mostly because of computational efficiency)

Variants of gradient descent

- Reminder: What is the difference between these? What are their pros/cons?
- Which one is applied most in practice? Why?

