Text Classification & Word Embedding

Classification, Evaluation, & Embedding

Ayoub Bagheri
a.bagheri@uu.nl
https://ayoubbagheri.nl/bigdata_NLP/

Outline

- Classification algorithms
- Evaluation
- Word embedding
- Skipgram learning
- Pre-trained embeddings
- State-of-the-art

Classification Algorithms

Hand-coded rules

- Rules based on combinations of words or other features
- Accuracy can be high: If rules carefully refined by expert
- But building and maintaining these rules is expensive
- Data/Domain specifics

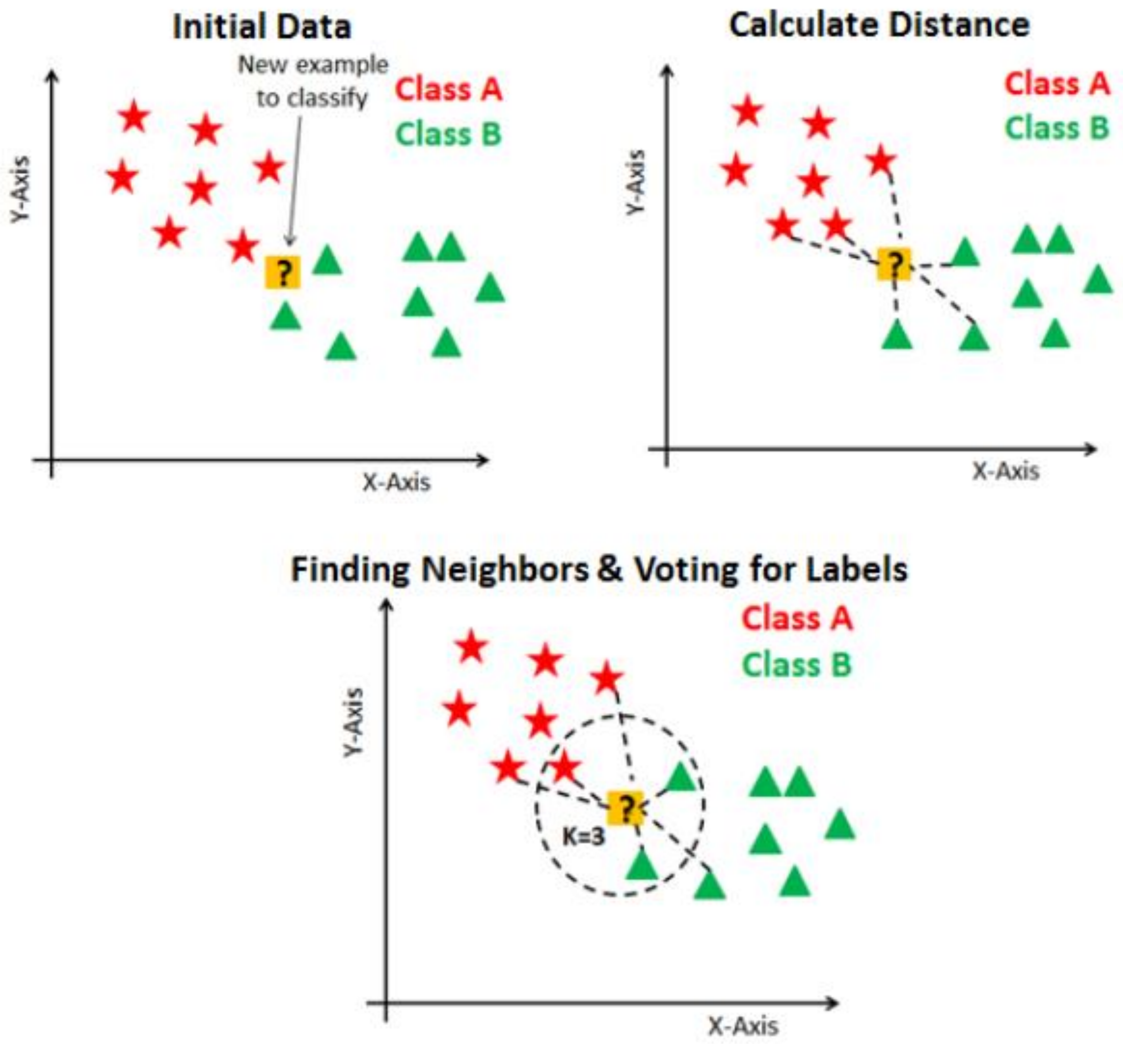
Supervised Machine Learning

- *Input:*
  - a document d
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
  - A training set of m hand-labeled documents $(d_1,c_1),\ldots,(d_m,c_m)$
- *Output:*
  - a learned classifier $y{:}d \rightarrow c$

Some of the methods

- K-nearest neighbors
- Naïve Bayes
- Logistic regression
- Support-vector machines
- Neural networks
- Deep learning

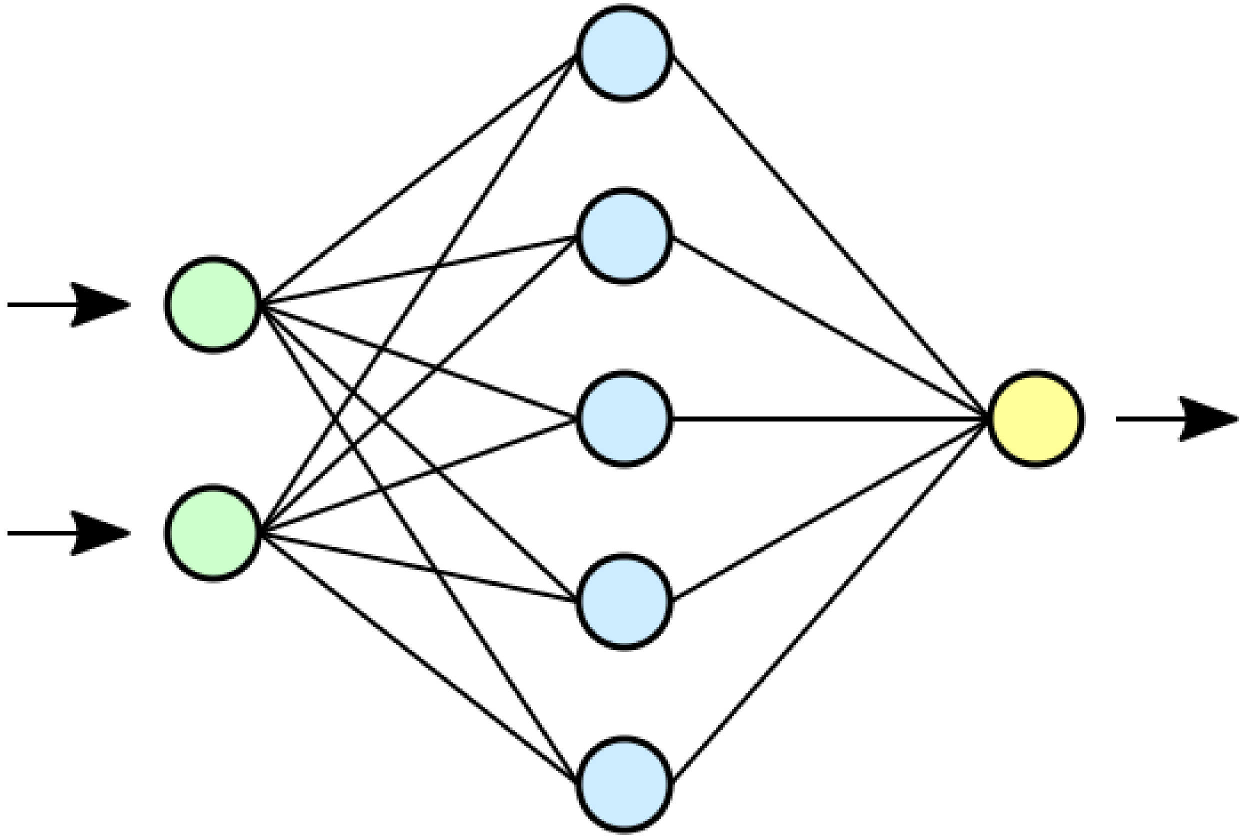
K-nearest neighbor



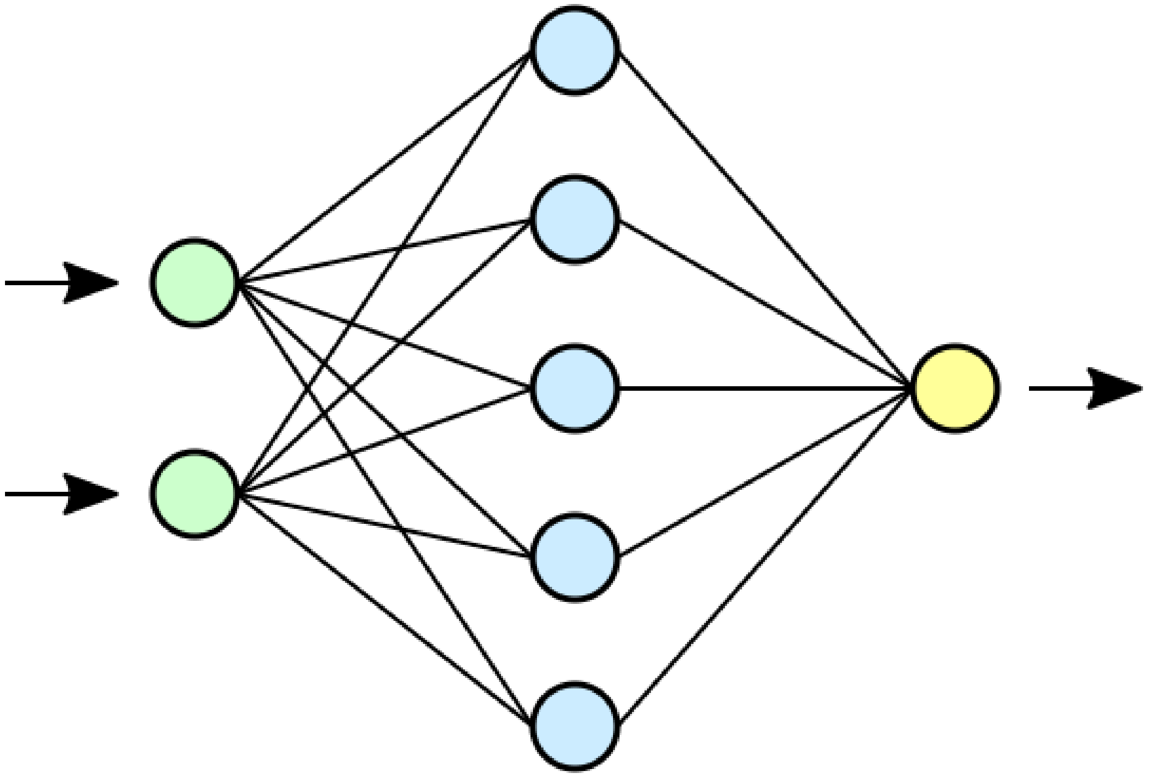
Neural networks, “deep learning”

- Compositional approach to curve-fitting;
- “Biologically inspired”
  (but don’t take that too seriously);
- Sound cool.

Neural network



Neural network



"Hidden" nodes:

Example:

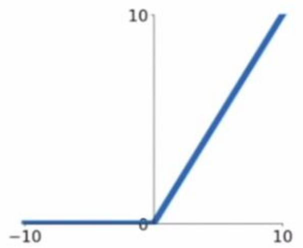
$$h_1 = f(w_{11}x_1 + w_{12}x_2)$$

Output:

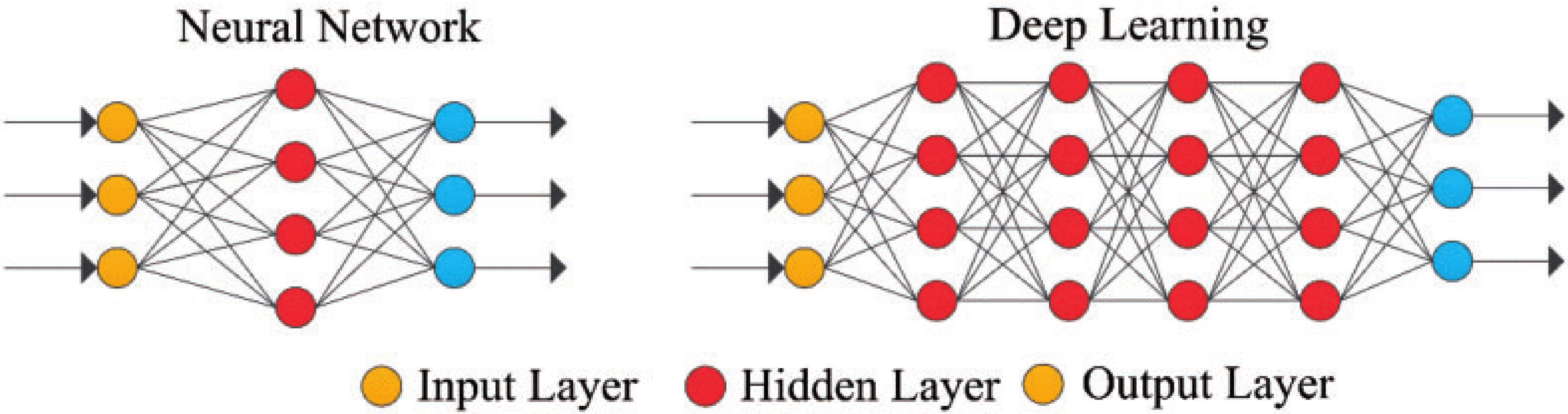
$$y = f(w_{21}h_1 + w_{22}h_2 + \cdots + w_{25}h_5)$$

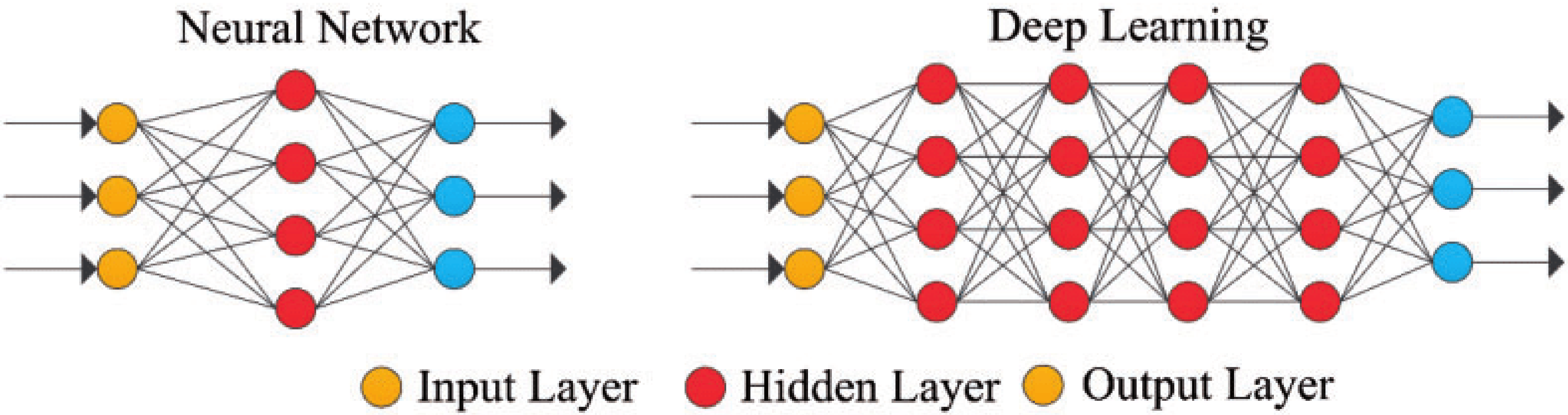
"Activation function":

- REIu $f(z) =$
- …



What makes a neural net “deep”?





Keep doing

$$z = g^{(n_h)}(g^{(\dots)}(g^{(2)}(g^{(1)}(\mathbf{x}))))$$

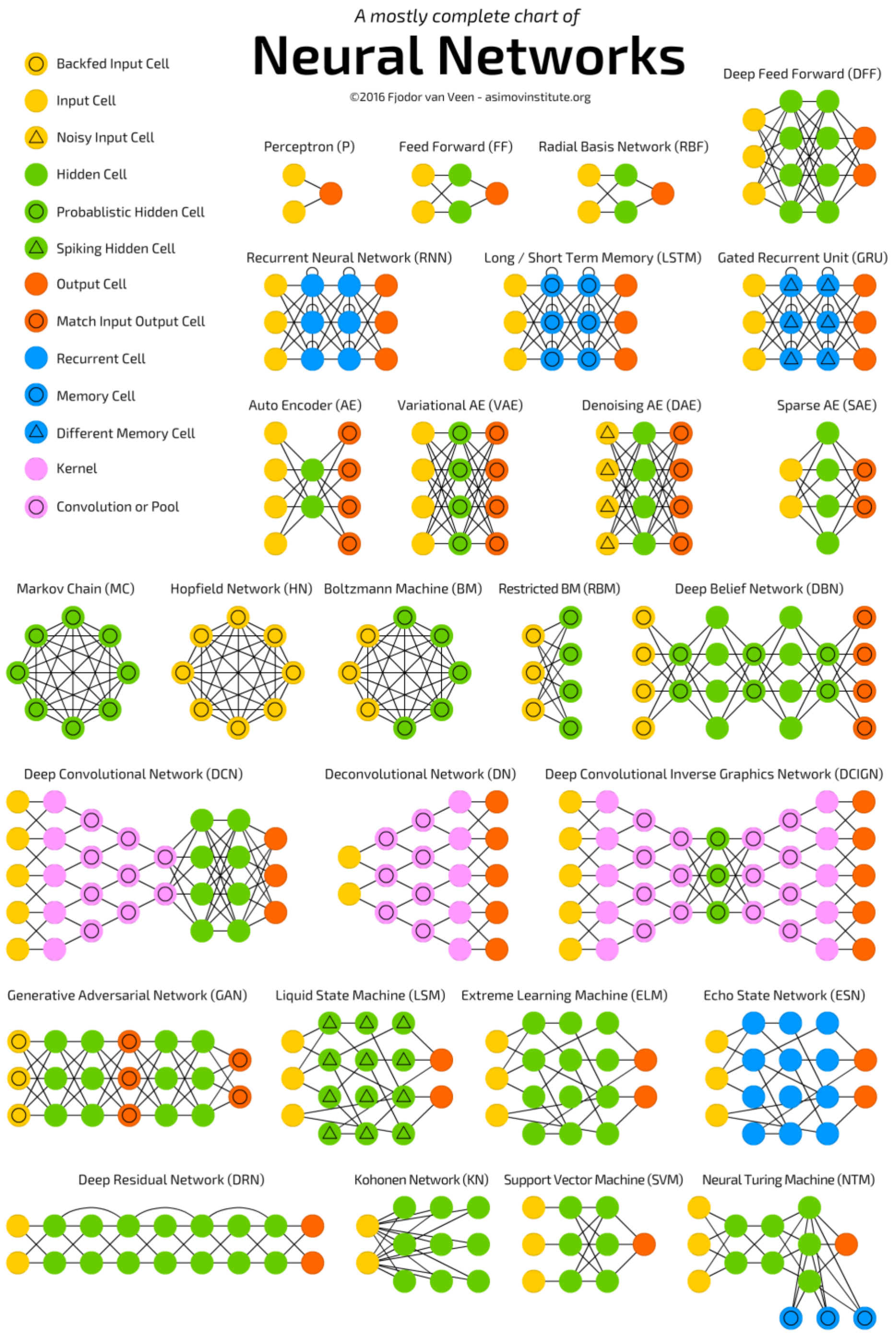
then $y \approx f(z)$.

Deep learning

- Output of each hidden layer is input to subsequent one
- Allow representation learning by building complex features out of simpler ones
- Go deep: exponential advantages, less overfitting
- Aggressive parameterization + aggressive regularization
- Compositional: efficient parametrization
- Learn relevant features: “End-to-end”

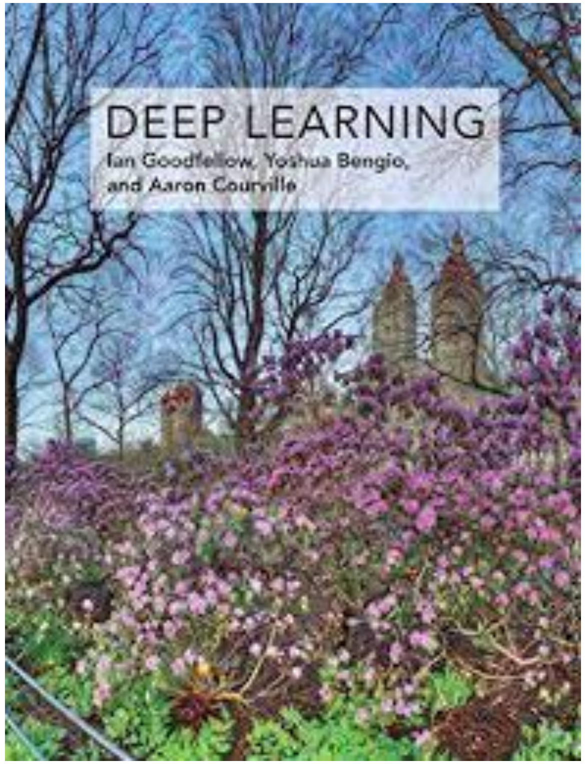
Different architectures

- By adjusting the arrows, layers, and activation functions, you can create models that are tailored to specific data, e.g.
- Convolutional (CNN): images, text, sound
- Recurrent (RNN): time series, text
- Graph (GNN): networks
- …

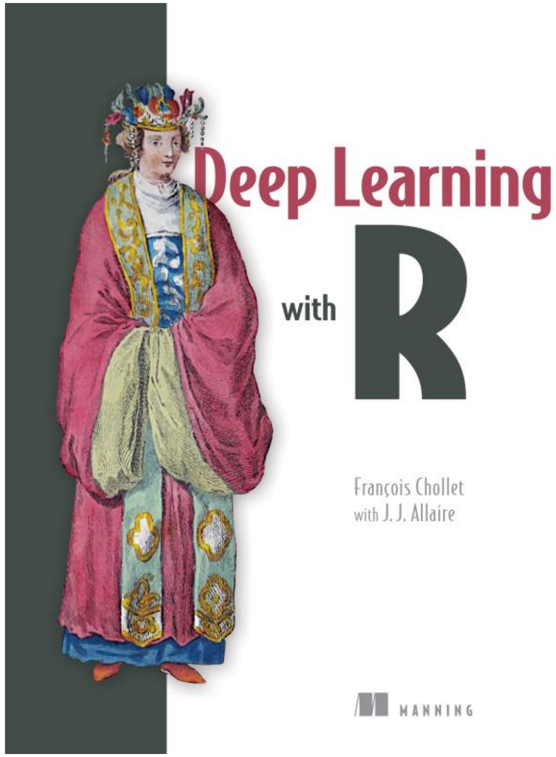


Deep learning in practice

- Good places to start:
  - https://keras.rstudio.com/
- ISLR Chapter 10



Goodfellow et al.



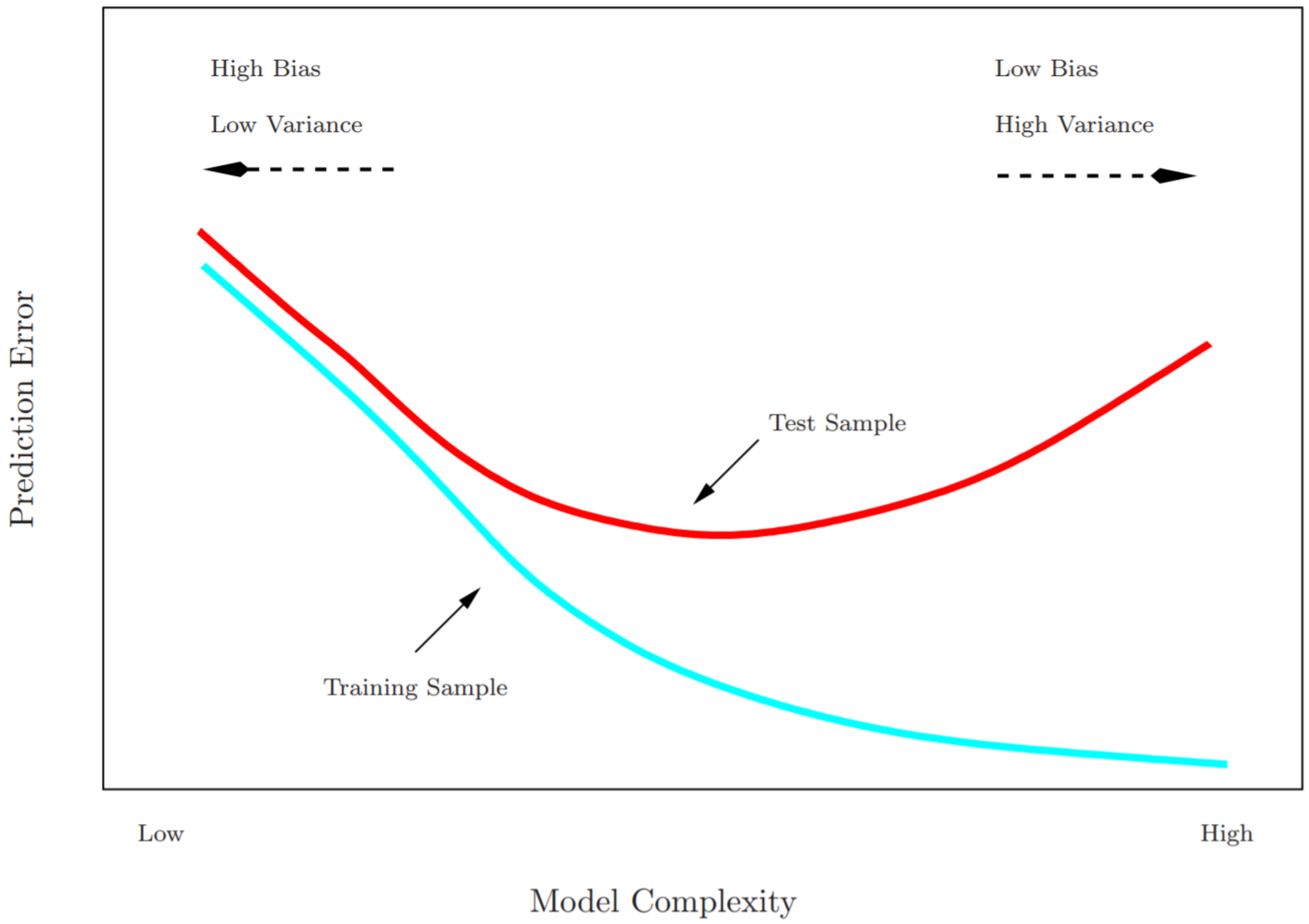
Chollet (R/Python version)

Evaluation

No free lunch

"Any two optimization algorithms are equivalent when their performance is averaged across all possible problems"

(Wolpert & MacReady)


High Bias
Low Variance
Low Bias
High Variance
Test Sample
Training Sample
Prediction Error
Low
High
Model Complexity

Confusion matrix

		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	**Negative Predictive Value** $\frac{TN}{(TN + FN)}$	**Accuracy** $\frac{TP + TN}{(TP + TN + FP + FN)}$

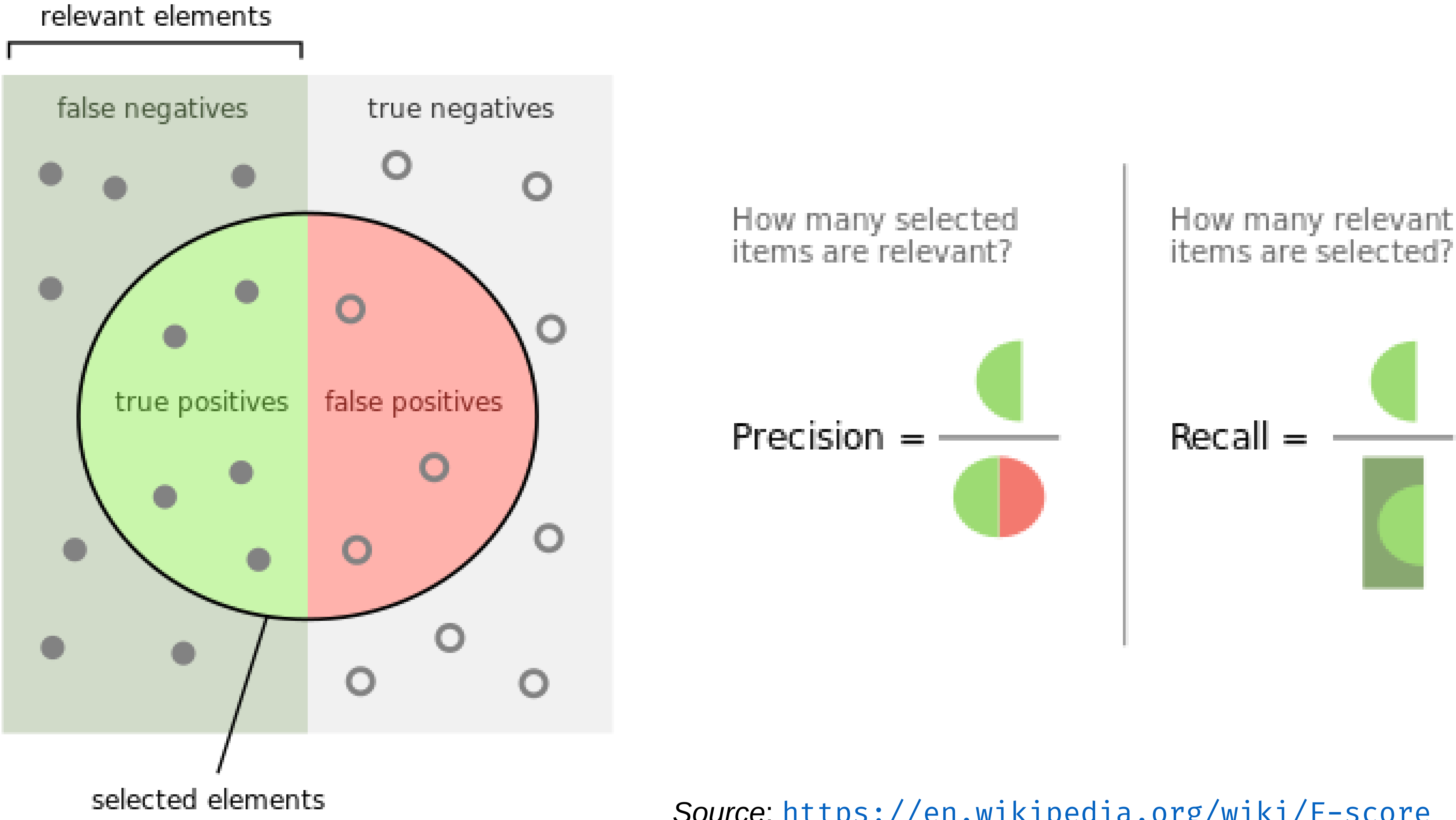
Accuracy

- Accuracy is a valid choice of evaluation for classification problems which are well balanced and not skewed.

Precision and recall

- **Precision**: % of selected items that are correct
  Recall: % of correct items that are selected

- Precision is a valid choice of evaluation metric when we want to be very sure of our prediction.
- Recall is a valid choice of evaluation metric when we want to capture as many positives as possible.



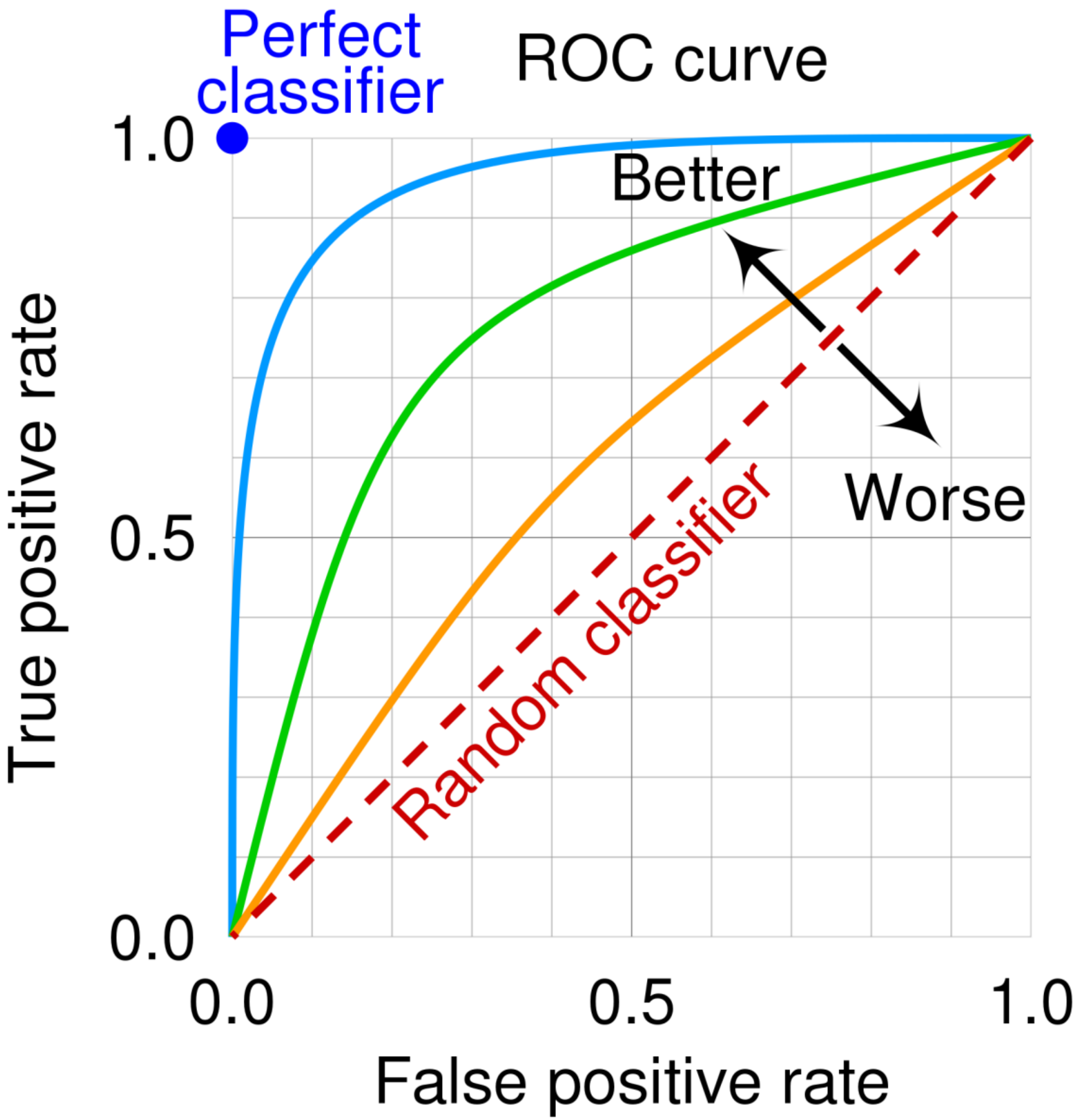
Source: https://en.wikipedia.org/wiki/F-score

A combined measure: F

- A combined measure that assesses the P/R tradeoff is F measure (weighted harmonic mean):

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

- The harmonic mean is a very conservative average
- People usually use balanced F1 measure
  - i.e., with β = 1 (that is, α = ½): $F = 2PR/(P+R)$



Perfect classifier
ROC curve
1.0
Better
Worse
0.5
Random classifier
0.0
0.0
0.5
1.0
True positive rate
False positive rate

Word Embedding

Word representations

How can we represent the meaning of words?

So, we can ask:

- How similar is cat to dog, or Paris to London?
- How similar is document A to document B?

Word as vectors

Can we represent words as vectors?

The vector representations should:

- capture semantics
  - similar words should be close to each other in the vector space
  - relation between two vectors should reflect the relationship between the two words
- be efficient (vectors with fewer dimensions are easier to work with)
- be interpretable

Word as vectors

How similar are the following two words? (not similar 0–10 very similar)

smart and **intelligent:**
easy and **big**:
easy and **difficult**:
hard and **difficult**:

Word as vectors

How similar are the following two words? (not similar 0–10 very similar)

smart and **intelligent: 9.20**
easy and **big**: **1.12**
easy and **difficult**: **0.58**
hard and **difficult**: **8.77**

(SimLex-999 dataset, https://fh295.github.io/simlex.html)

Words as Vectors

One-hot encoding

Map each word to a unique identifier

e.g. cat (3) and dog (5).

- Vector representation: all zeros, except 1 at the ID

cat	0	0	1	0	0	0	0
dog	0	0	0	0	1	0	0
car	0	0	0	0	0	0	1

One-hot encoding

Map each word to a unique identifier

e.g. cat (3) and dog (5).

- Vector representation: all zeros, except 1 at the ID

cat	0	0	1	0	0	0	0
dog	0	0	0	0	1	0	0
car	0	0	0	0	0	0	1

What are limitations of one-hot encodings?

One-hot encoding

Map each word to a unique identifier

e.g. cat (3) and dog (5).

• Vector representation: all zeros, except 1 at the ID

cat	0	0	1	0	0	0	0
dog	0	0	0	0	1	0	0
car	0	0	0	0	0	0	1

Even related words have distinct vectors!

High number of dimensions

Distributional hypothesis: Words that occur in similar contexts tend to have similar meanings.

You shall know a word by the company it keeps. (Firth, J. R. 1957:11)

Word vectors based on co-occurrences

documents as context
word-document matrix

	doc_1	doc_2	doc_3	doc_4	doc_5	doc_6	doc_7
cat	5	2	0	1	4	0	0
dog	7	3	1	0	2	0	0
car	0	0	1	3	2	1	1

Word vectors based on co-occurrences

documents as context
word-document matrix

	doc_1	doc_2	doc_3	doc_4	doc_5	doc_6	doc_7
cat	5	2	0	1	4	0	0
dog	7	3	1	0	2	0	0
car	0	0	1	3	2	1	1

neighboring words as context
word-word matrix

	cat	dog	car	bike	book	house	tree
cat	0	3	1	1	1	2	3
dog	3	0	2	1	1	3	1
car	0	0	1	3	2	1	1

Word vectors based on co-occurrences

There are many variants:

- Context (words, documents, which window size, etc.)
- Weighting (raw frequency, etc.)

Vectors are sparse: Many zero entries.

Therefore: Dimensionality reduction is often used (e.g., SVD)

These methods are sometimes called **count-based** methods as they work directly on **co-occurrence** counts.

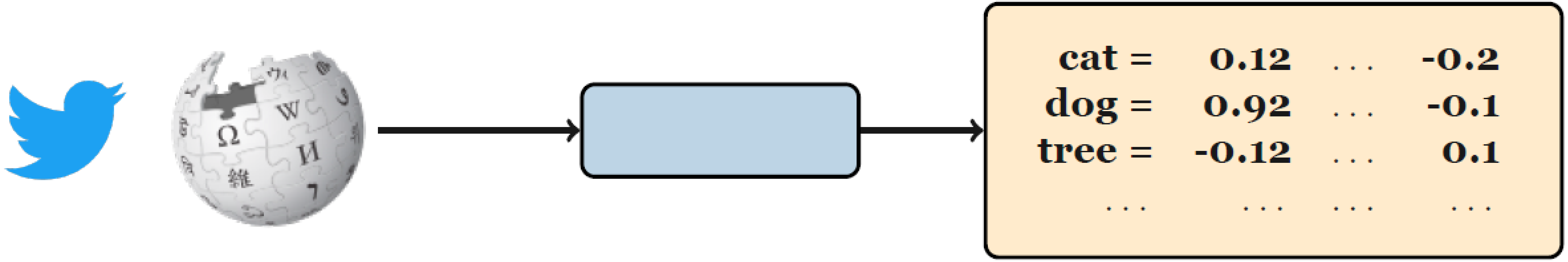
Word embeddings

- Vectors are short; typically 50-1024 dimensions ☺
- Vectors are dense (mostly non-zero values)
- Very effective for many NLP tasks ☺
- Individual dimensions are less interpretable ☹

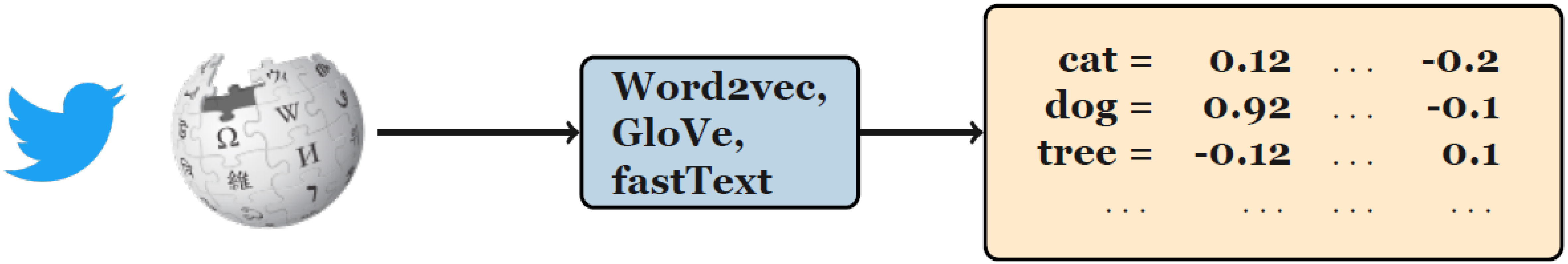
cat	0.52	0.48	-0.01	⋯	0.28
dog	0.32	0.42	-0.09	⋯	0.78

How do we learn word embeddings?

Learning word embeddings



Learning word embeddings



Training data for word embeddings

- Use **text itself** as training data for the model!
  - A form of self-supervision.

- Train a **classifier** (neural network, logistic regression, or SVM, etc.) to predict the next word given previous words.

Exercise: Word prediction task

Yesterday I went to the ?

A new study has highlighted the positive ?

Which word comes next?

Word2Vec

- Popular embedding method
- Very fast to train

- **Can you find 5 nearest words to “epidemiology”? (use Word2Vec all)**
- https://projector.tensorflow.org/

Word2Vec

The domestic **cat** is a small, typically furry carnivorous mammal

w_{-2} w_{-1} w_0 w_1 w_2 w_3 w_4 w_5

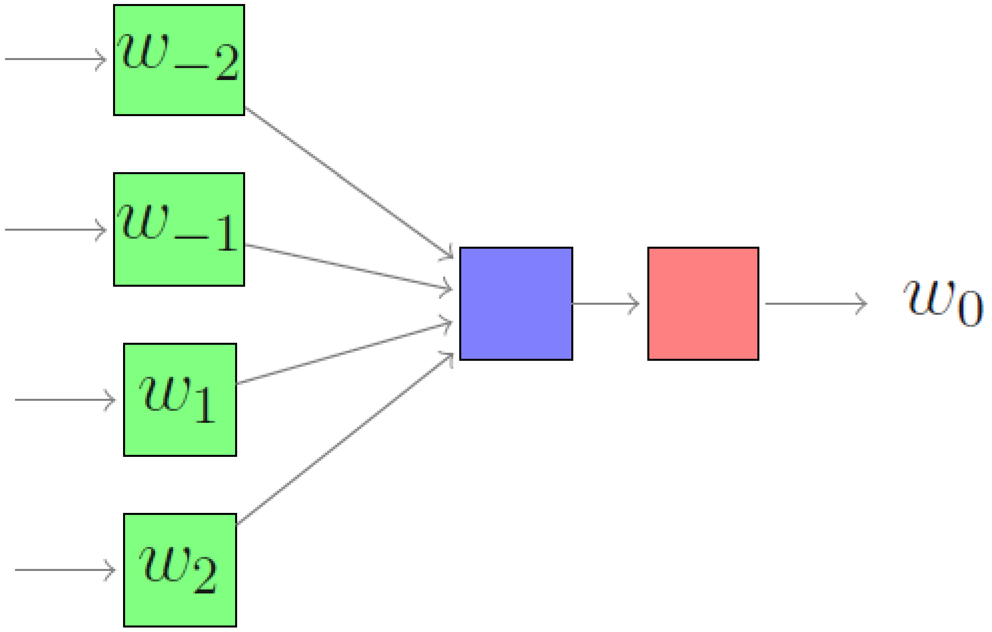
We have **target** words (cat) and **context** words (here: window size = 5).

Word2Vec

- Instead of **counting** how often each word w occurs near a target word
  - Train a classifier on a binary **prediction** task:
    - Is w likely to show up near target?
- We don’t actually care about this task
  - But we'll take the learned classifier weights as the word embeddings
- Big idea: **self-supervision**
  - A word c that occurs near target in the corpus as the gold "correct answer" for supervised learning
  - **No need for human labels**
  - Bengio et al. (2003); Collobert et al. (2011)

Word2Vec algorithms

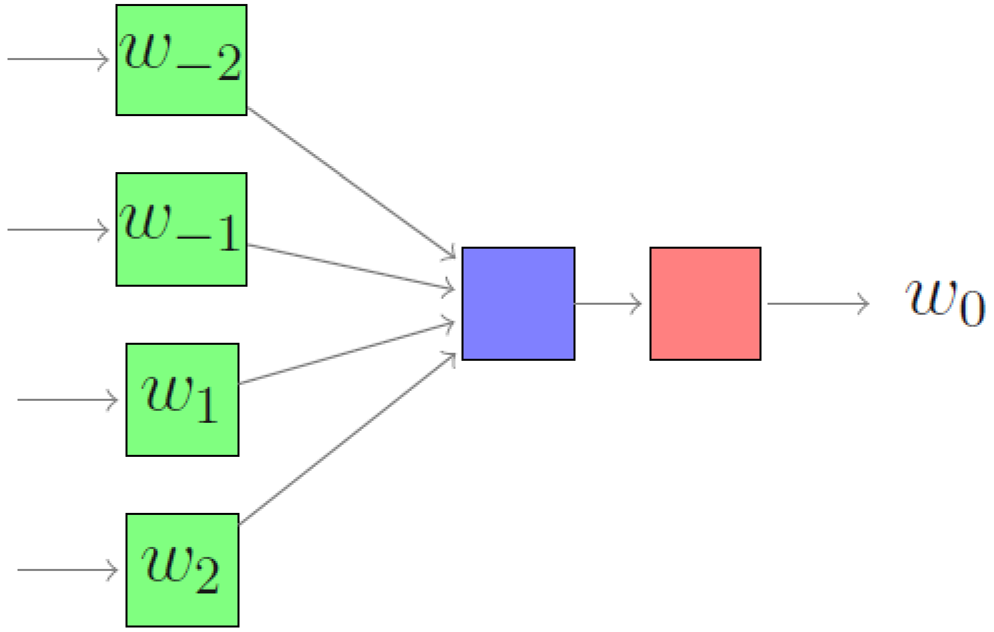
Continuous Bag-Of-Words (CBOW)



one snowy ? she went

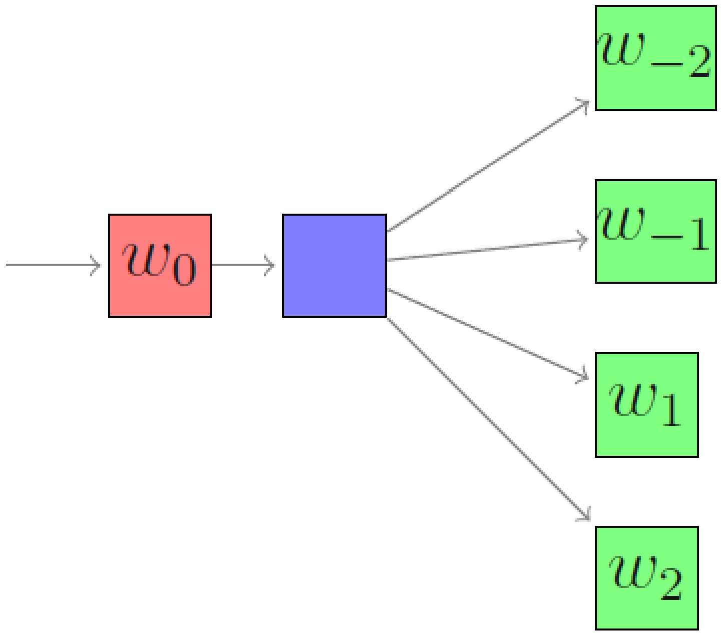
Word2Vec algorithms

Continuous Bag-Of-Words (CBOW)



one snowy ? she went

skipgram



? ? day ? ?

Skipgram overview

The domestic **cat** is a small, typically furry carnivorous mammal

1. **Create examples**
   - Positive examples: Target word and neighboring context
   - Negative examples: Target word and randomly sampled words from the lexicon (*negative sampling*)
2. Train a **logistic regression** model to distinguish between the positive and negative examples
3. The resulting **weights** are the embeddings!

word (w)	context (c)	label
cat	small	1
cat	furry	1
cat	car	0
...	...	...

Embedding vectors are essentially a byproduct!

Pre-trained Embeddings

Pre-trained embeddings

- I want to build a system to **solve a task** (e.g., sentiment analysis)
  - Use pre-trained embeddings. Should I **fine-tune**?
    - Lots of data: yes
    - Just a small dataset: no
- **Analysis** (e.g., bias, semantic change)
  - Train embeddings from scratch

Layer embedding in Keras

```
layer_embedding(input_dim = max_words, output_dim = dim_size,
                input_length = maxlen,
                # put weights into list and do not allow training
                weights = list(word_embeds), trainable = FALSE)
```

State-Of-The-Art

State-of-the-art

- Recurrent neural networks
  - LSTM
  - GRU
  - Bi-directional networks
- Contextual embeddings
- Transformers: https://app.inferkit.com/demo

Practical 2