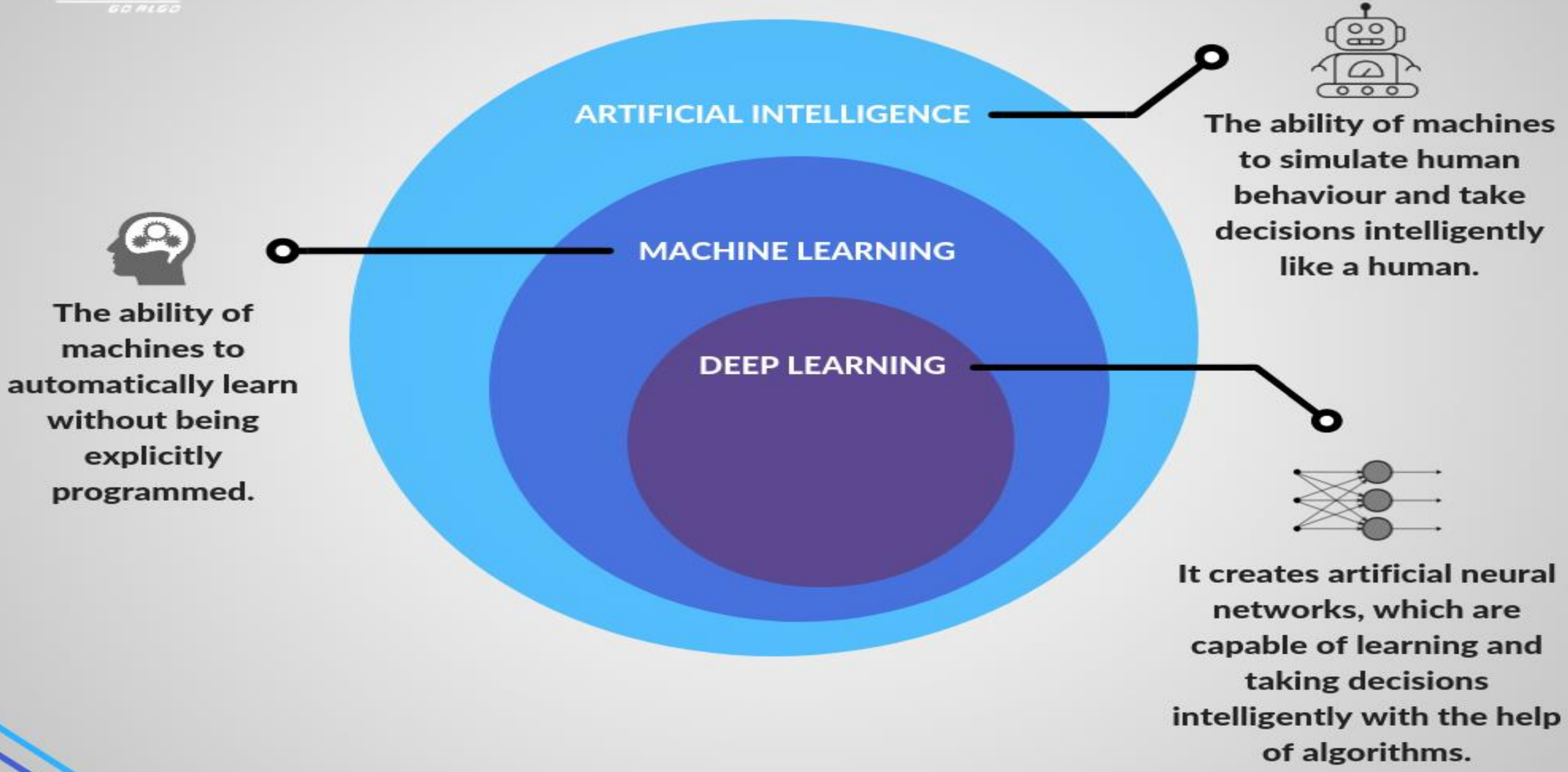
# Deep NLP workshop


Shahid Beheshti University (Dec 2019)

Abbas Hosseini
Ehsan Taher

# outline

- introduction to neural network
- RNN
- NLP tasks
- Word2Vec and Word embeddings
- pandas
- data preparing
- introduction to Keras


ARTIFICIAL INTELLIGENCE
The ability of machines to simulate human behaviour and take decisions intelligently like a human.
MACHINE LEARNING
The ability of machines to automatically learn without being explicitly programmed.
DEEP LEARNING
It creates artificial neural networks, which are capable of learning and taking decisions intelligently with the help of algorithms.

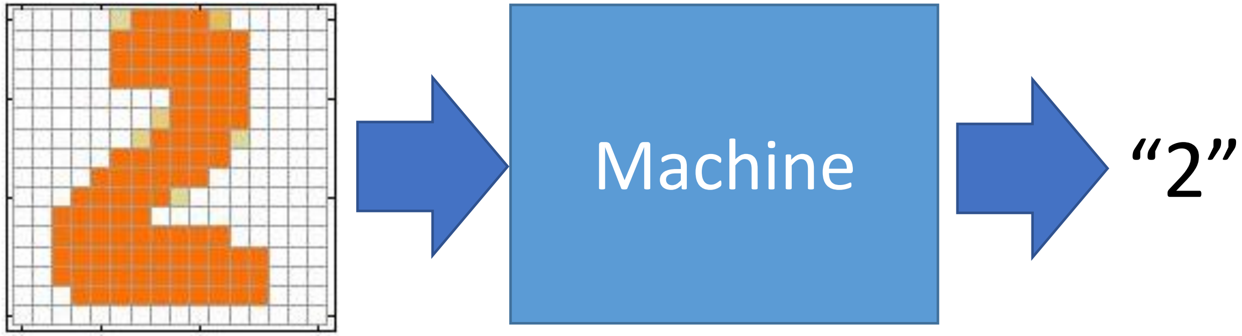
# Deep learning attracts lots of attention.

- Google Trends

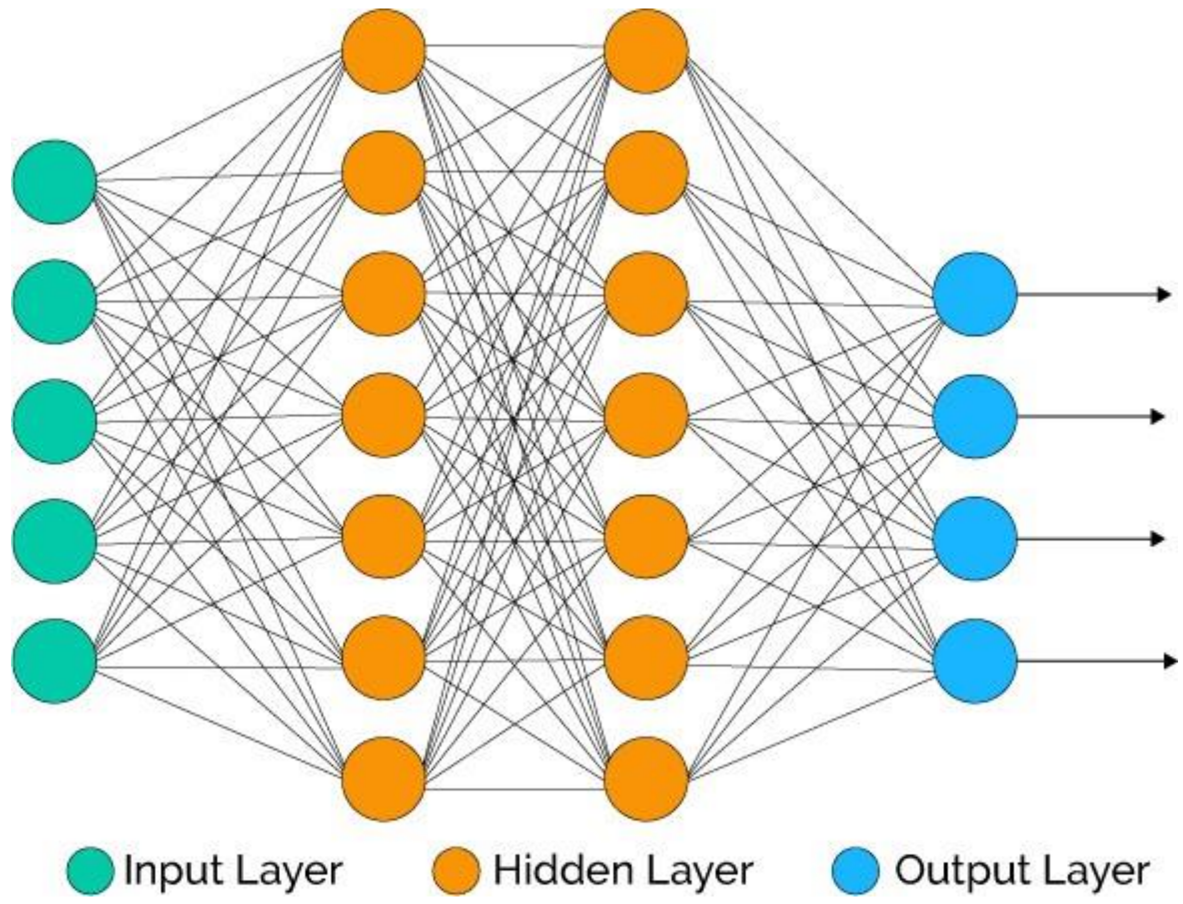
Deep learning obtains many exciting results.

# Example Application

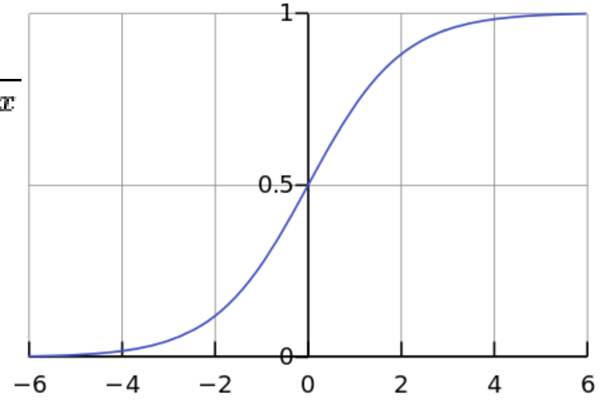
- Handwriting Digit Recognition

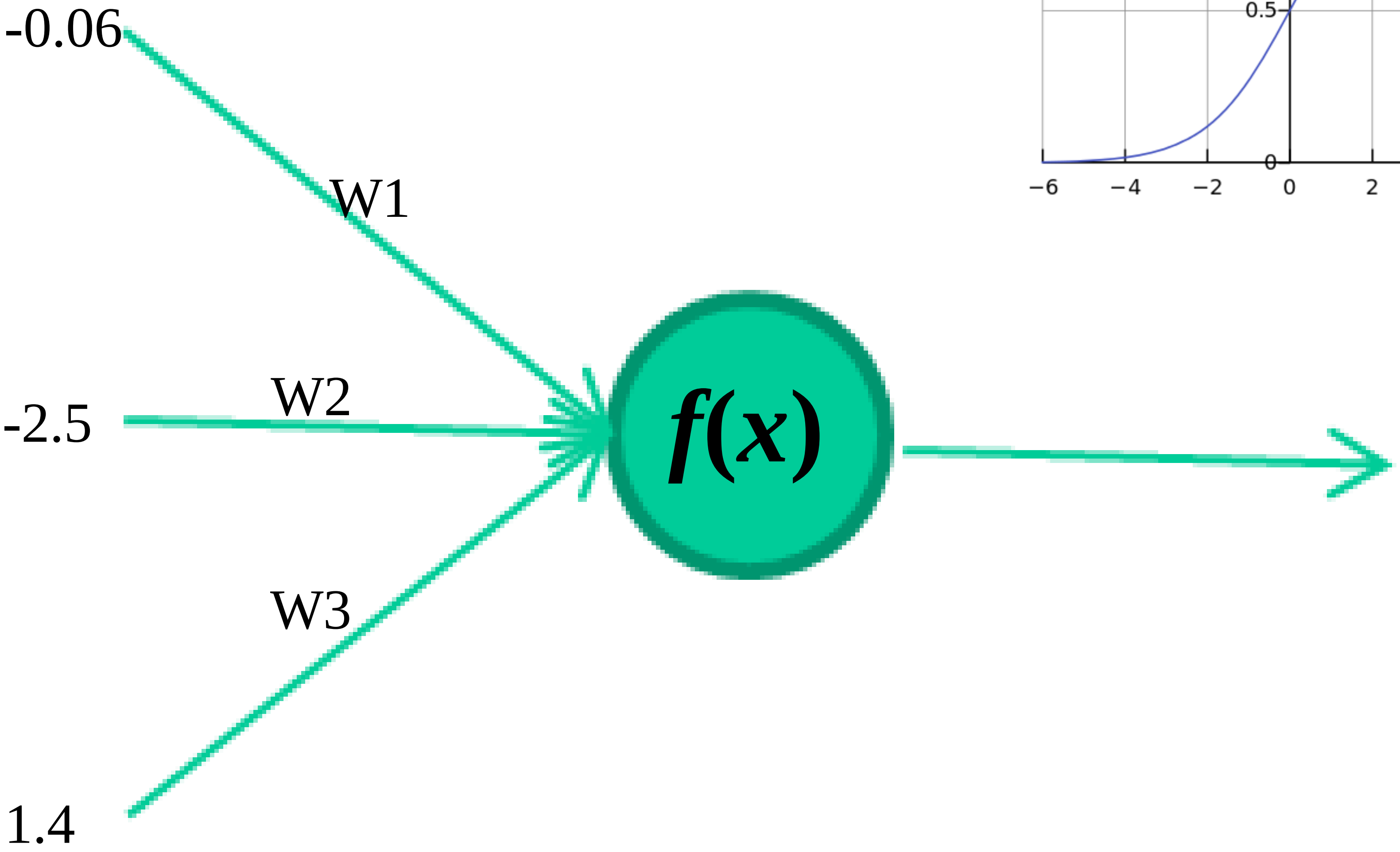


# Neural Networks

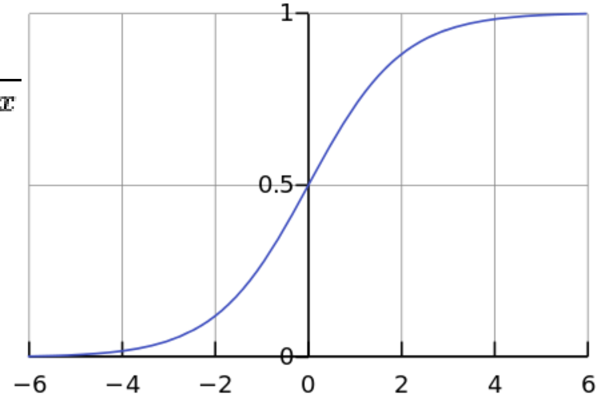


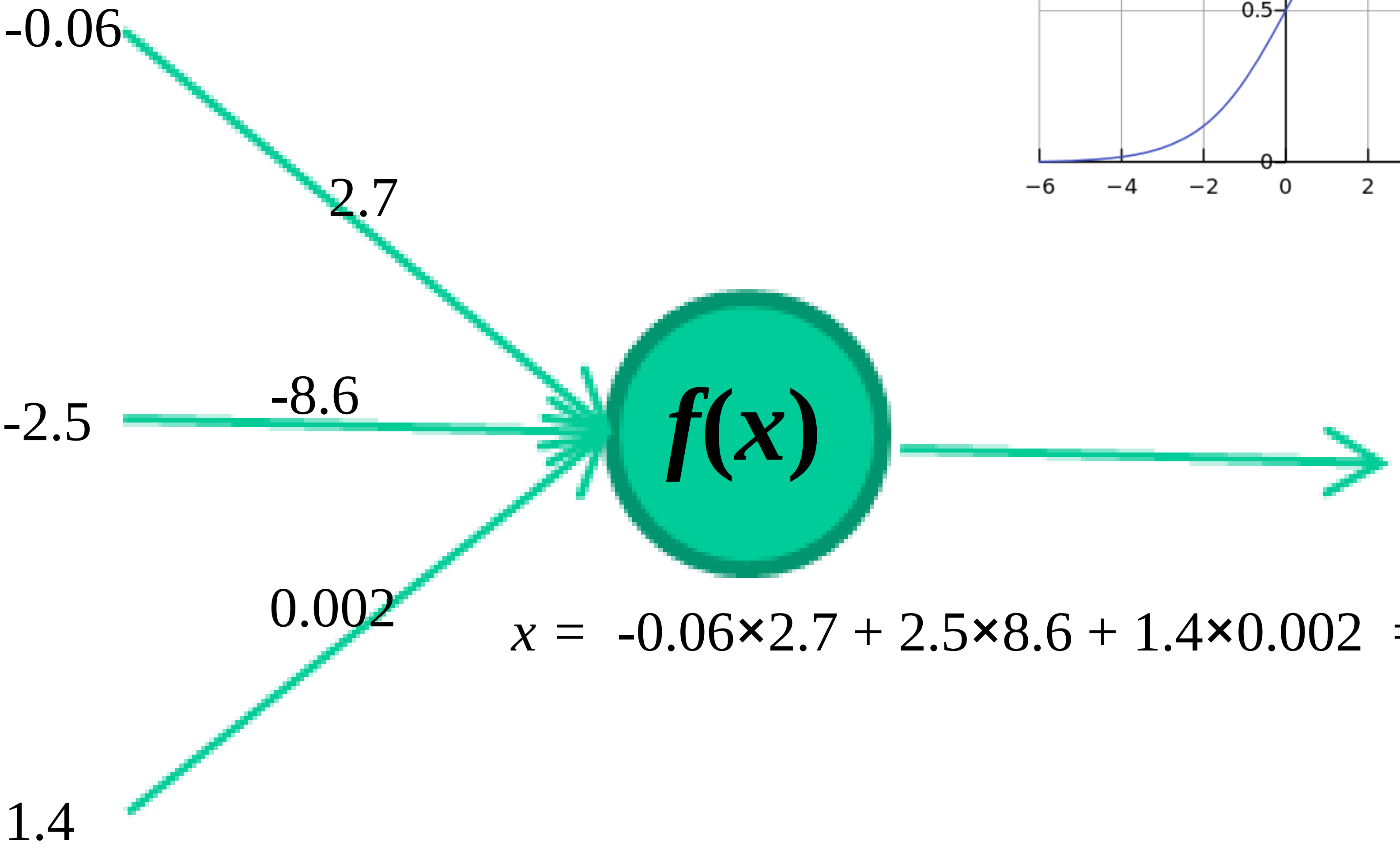
$$f(x) = \frac{1}{1 + e^{-x}}$$



$$f(x) = \frac{1}{1 + e^{-x}}$$

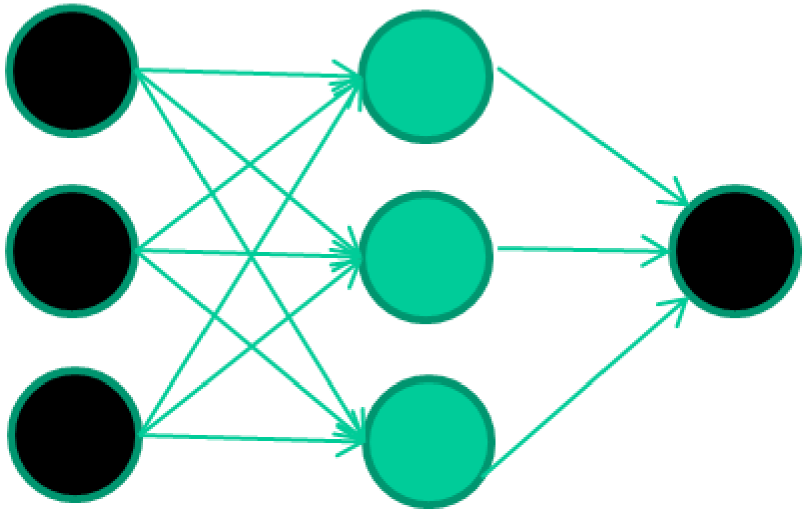



$$x = -0.06 \times 2.7 + 2.5 \times 8.6 + 1.4 \times 0.002 = 21.34$$

*A dataset*

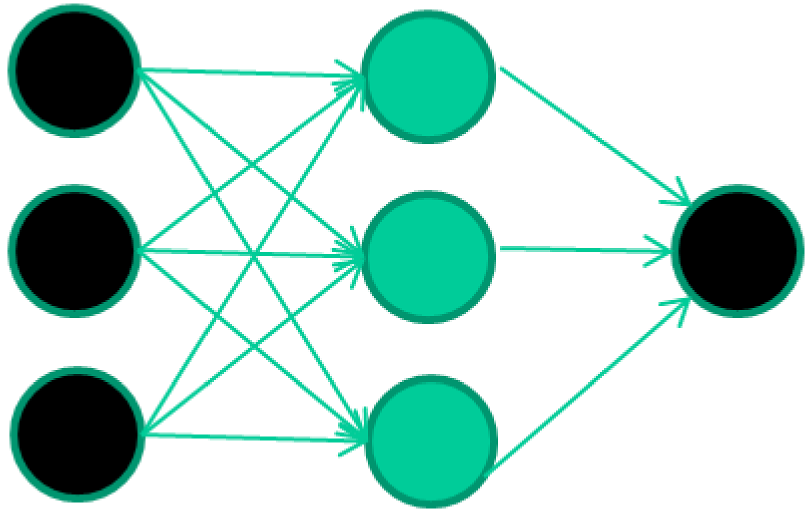
| ***Fields*** | | | ***class*** |
|---|---|---|---|
| 1.4 | 2.7 | 1.9 | 0 |
| 3.8 | 3.4 | 3.2 | 0 |
| 6.4 | 2.8 | 1.7 | 1 |
| 4.1 | 0.1 | 0.2 | 0 |
| etc … | | | |



*Training the neural network*

| ***Fields*** | | | ***class*** |
|---|---|---|---|
| 1.4 | 2.7 | 1.9 | 0 |
| 3.8 | 3.4 | 3.2 | 0 |
| 6.4 | 2.8 | 1.7 | 1 |
| 4.1 | 0.1 | 0.2 | 0 |

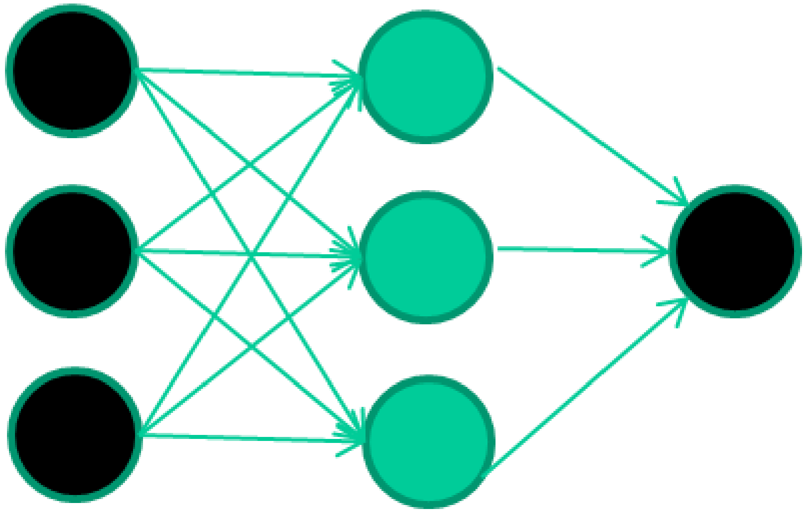
etc …



*Training data*

| ***Fields*** | | | ***class*** |
|---|---|---|---|
| 1.4 | 2.7 | 1.9 | 0 |
| 3.8 | 3.4 | 3.2 | 0 |
| 6.4 | 2.8 | 1.7 | 1 |
| 4.1 | 0.1 | 0.2 | 0 |
| etc … | | | |

**Initialise with random weights**

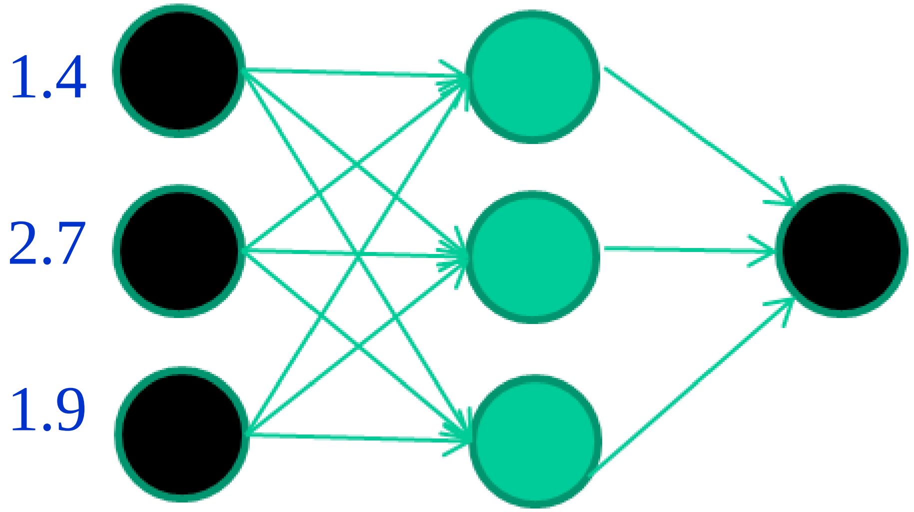


*Training data*

| ***Fields*** | | | ***class*** |
|---|---|---|---|
| 1.4 | 2.7 | 1.9 | 0 |
| 3.8 | 3.4 | 3.2 | 0 |
| 6.4 | 2.8 | 1.7 | 1 |
| 4.1 | 0.1 | 0.2 | 0 |

etc …

**Present a training pattern**

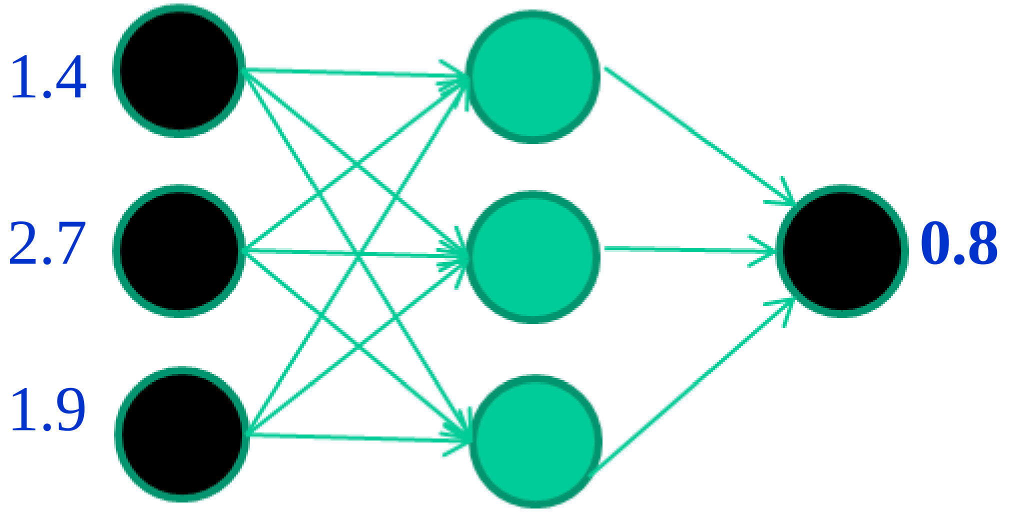


*Training data*

| ***Fields*** | | | ***class*** |
|---|---|---|---|
| 1.4 | 2.7 | 1.9 | 0 |
| 3.8 | 3.4 | 3.2 | 0 |
| 6.4 | 2.8 | 1.7 | 1 |
| 4.1 | 0.1 | 0.2 | 0 |

etc …

**Feed it through to get output**

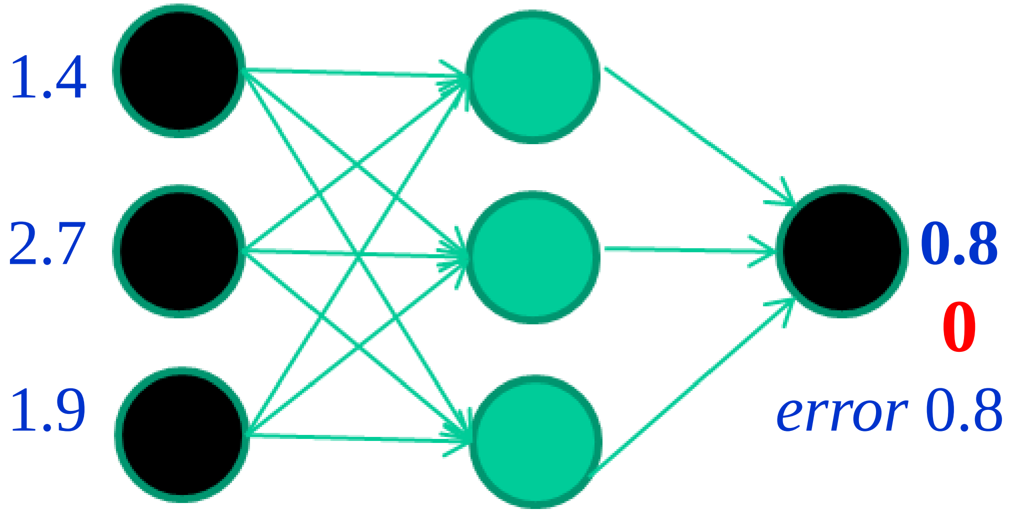


*Training data*

| ***Fields*** | | | ***class*** |
|---|---|---|---|
| 1.4 | 2.7 | 1.9 | 0 |
| 3.8 | 3.4 | 3.2 | 0 |
| 6.4 | 2.8 | 1.7 | 1 |
| 4.1 | 0.1 | 0.2 | 0 |

etc …

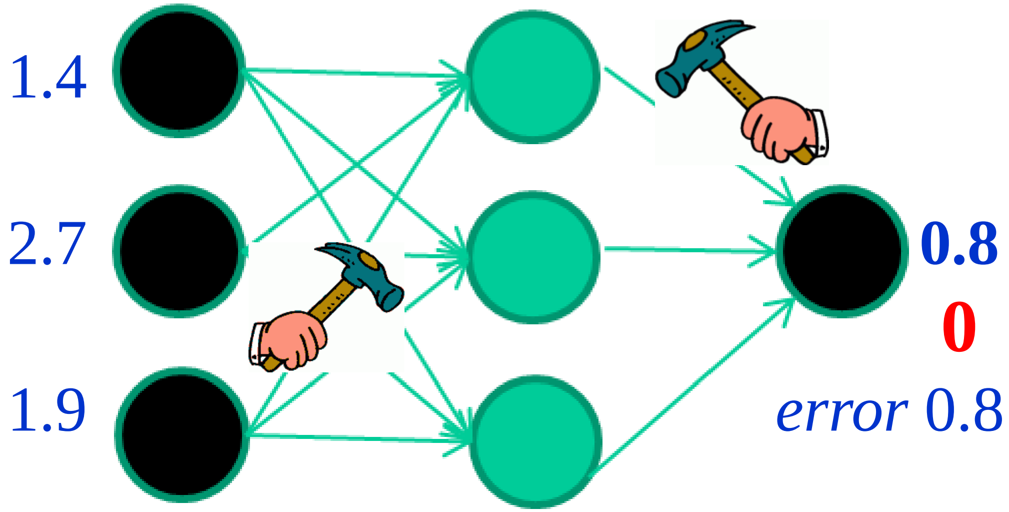
**Compare with target output**



*Training data*

| ***Fields*** | | | ***class*** |
|---|---|---|---|
| 1.4 | 2.7 | 1.9 | 0 |
| 3.8 | 3.4 | 3.2 | 0 |
| 6.4 | 2.8 | 1.7 | 1 |
| 4.1 | 0.1 | 0.2 | 0 |
| etc … | | | |

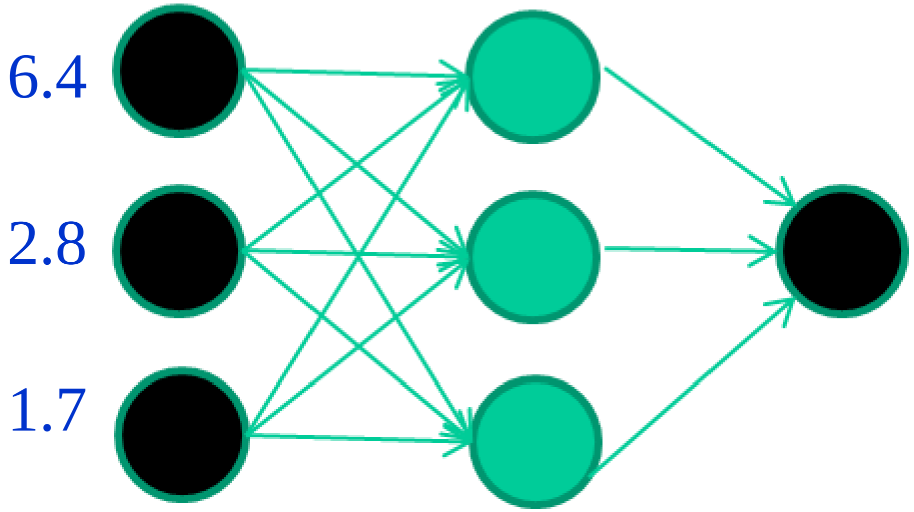
**Adjust weights based on error**



*Training data*

| ***Fields*** | | | ***class*** |
|---|---|---|---|
| 1.4 | 2.7 | 1.9 | 0 |
| 3.8 | 3.4 | 3.2 | 0 |
| 6.4 | 2.8 | 1.7 | 1 |
| 4.1 | 0.1 | 0.2 | 0 |
| etc … | | | |

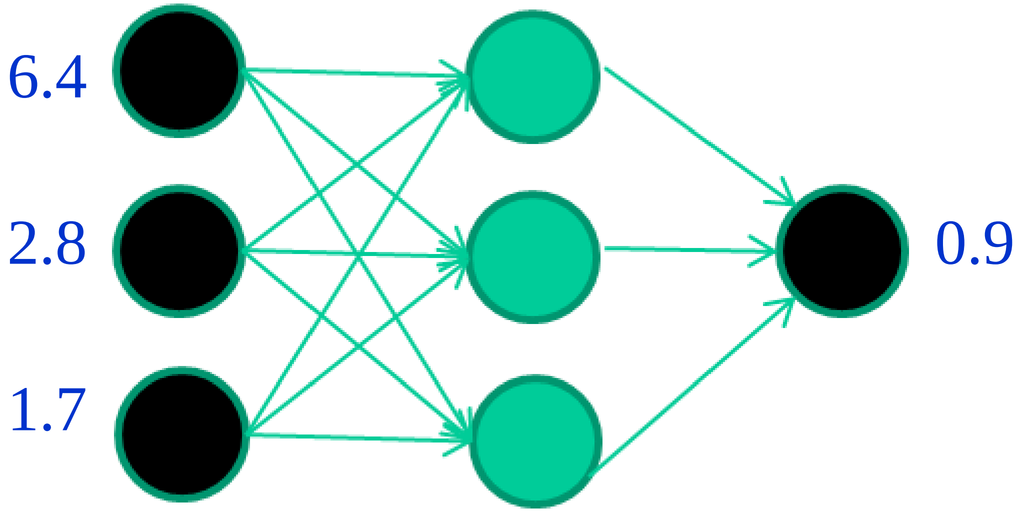
**Present a training pattern**



*Training data*

| ***Fields*** | | | ***class*** |
| --- | --- | --- | --- |
| 1.4 | 2.7 | 1.9 | 0 |
| 3.8 | 3.4 | 3.2 | 0 |
| 6.4 | 2.8 | 1.7 | 1 |
| 4.1 | 0.1 | 0.2 | 0 |
| etc … | | | |

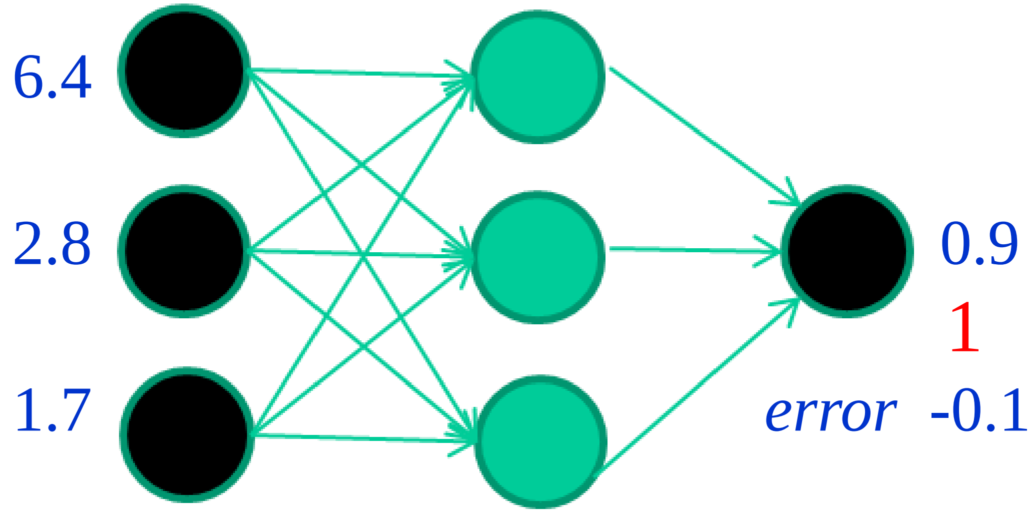
**Feed it through to get output**



*Training data*

| ***Fields*** | | | ***class*** |
|---|---|---|---|
| 1.4 | 2.7 | 1.9 | 0 |
| 3.8 | 3.4 | 3.2 | 0 |
| 6.4 | 2.8 | 1.7 | 1 |
| 4.1 | 0.1 | 0.2 | 0 |
| etc … | | | |

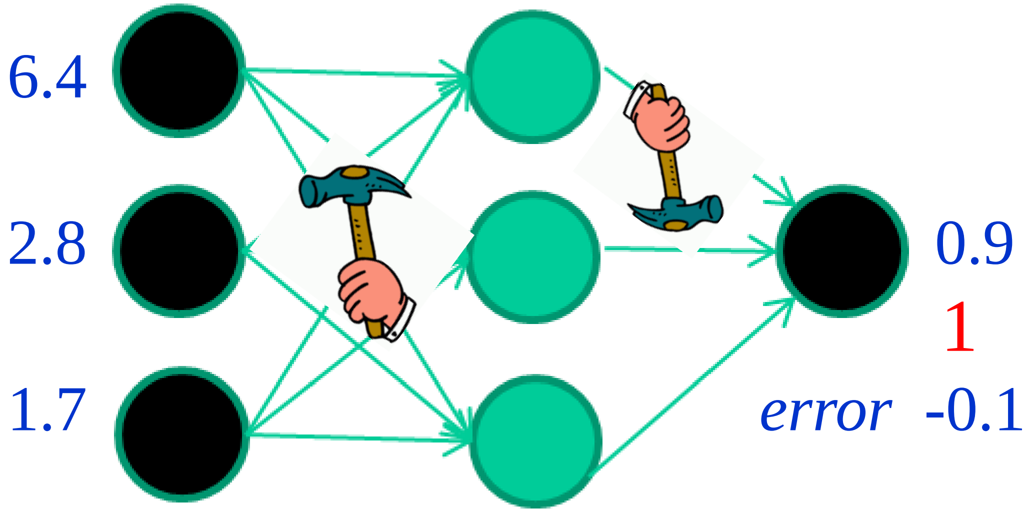
**Compare with target output**



*Training data*

| ***Fields*** | | | ***class*** |
|---|---|---|---|
| 1.4 | 2.7 | 1.9 | 0 |
| 3.8 | 3.4 | 3.2 | 0 |
| 6.4 | 2.8 | 1.7 | 1 |
| 4.1 | 0.1 | 0.2 | 0 |
| etc … | | | |

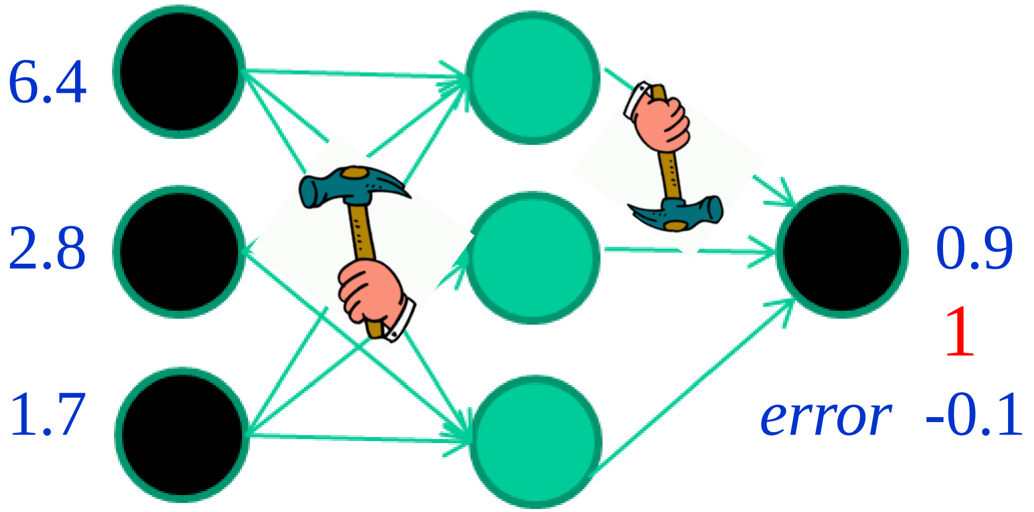
**Adjust weights based on error**



*Training data*

| ***Fields*** | | | ***class*** |
|---|---|---|---|
| 1.4 | 2.7 | 1.9 | 0 |
| 3.8 | 3.4 | 3.2 | 0 |
| 6.4 | 2.8 | 1.7 | 1 |
| 4.1 | 0.1 | 0.2 | 0 |
| etc … | | | |

**And so on ….**

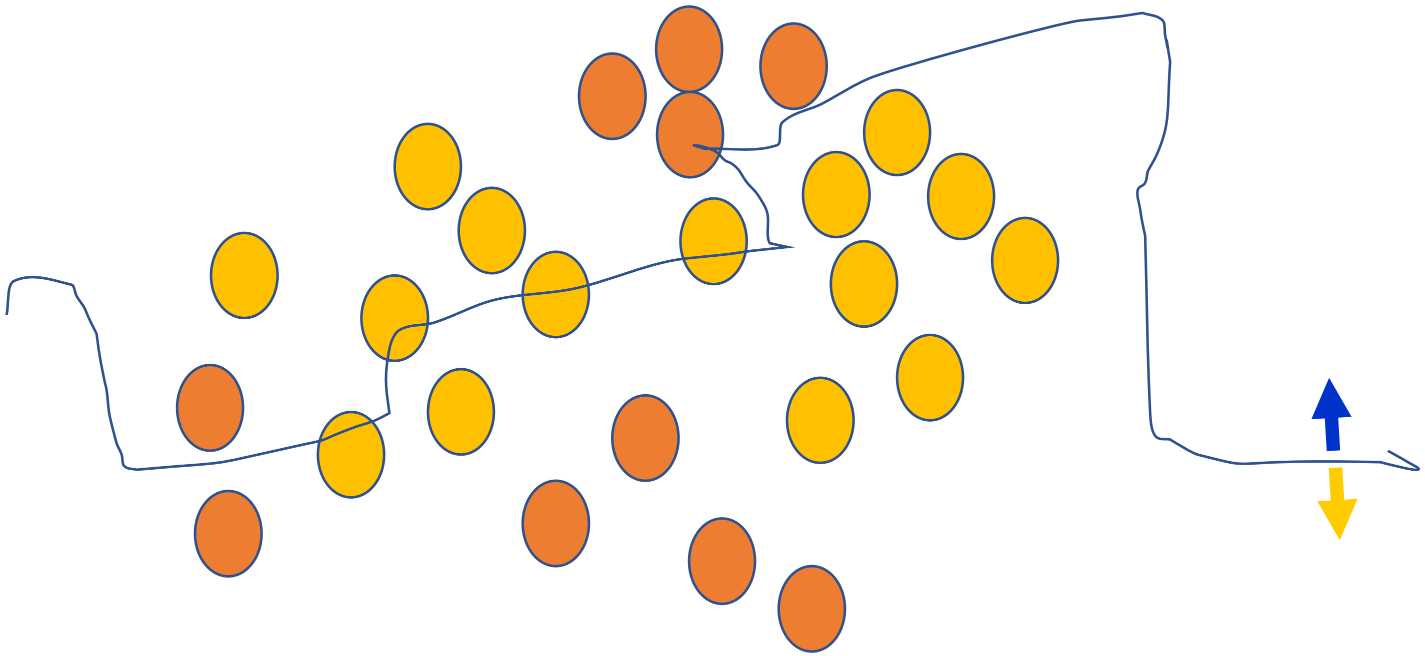



**Repeat this thousands, maybe millions of times – each time taking a random training instance, and making slight weight adjustments**

*Algorithms for weight adjustment are designed to make changes that will reduce the error*

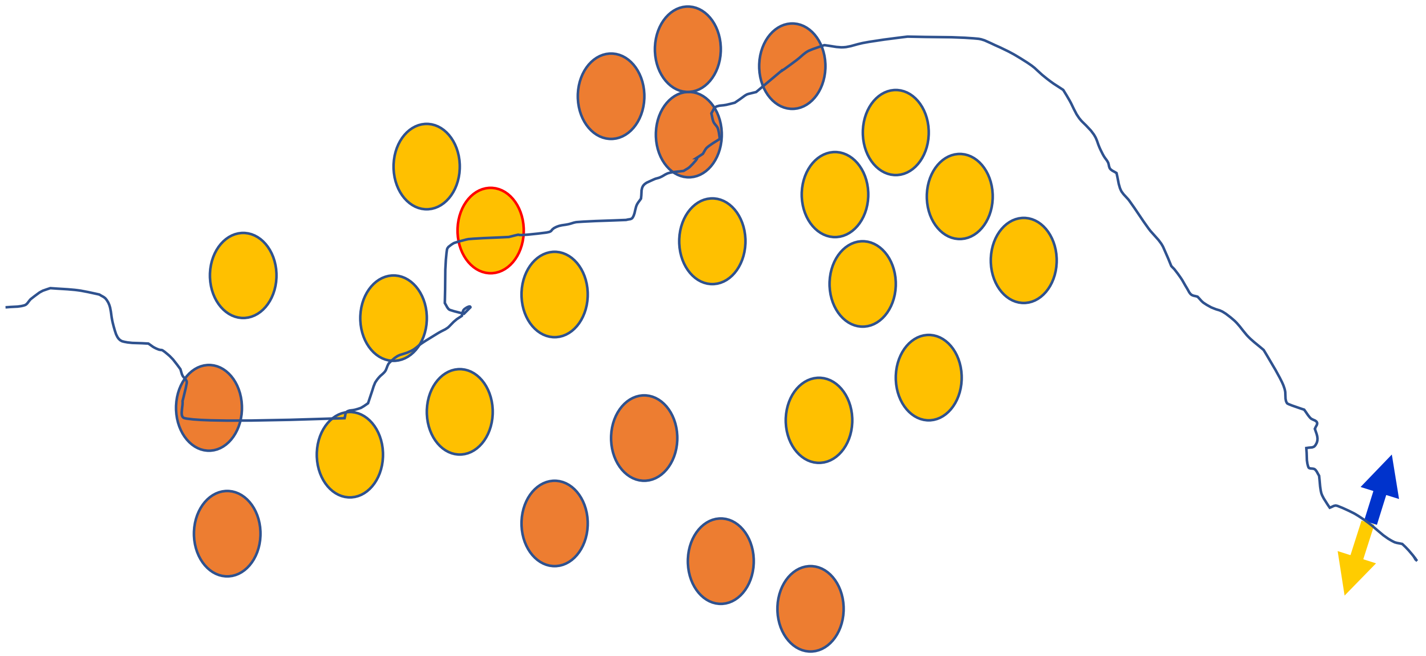
# The decision boundary perspective…

**Initial random weights**



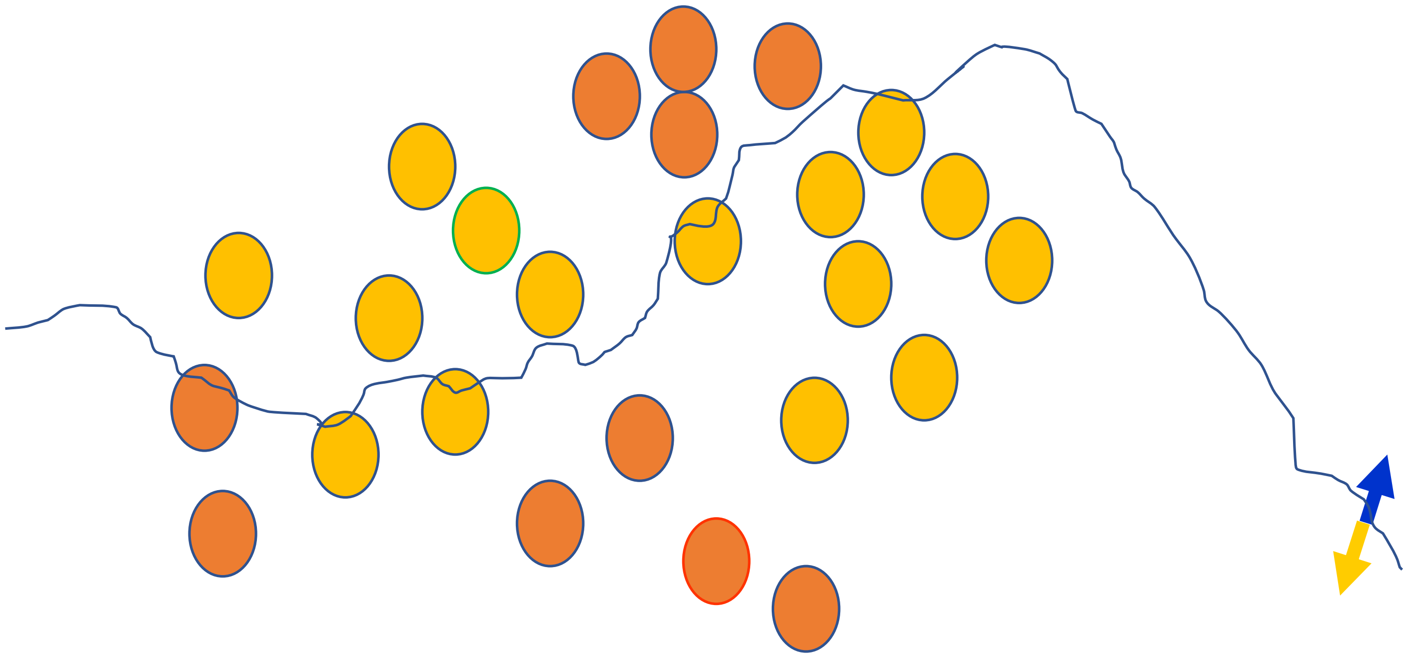
# The decision boundary perspective…

**Present a training instance / adjust the weights**



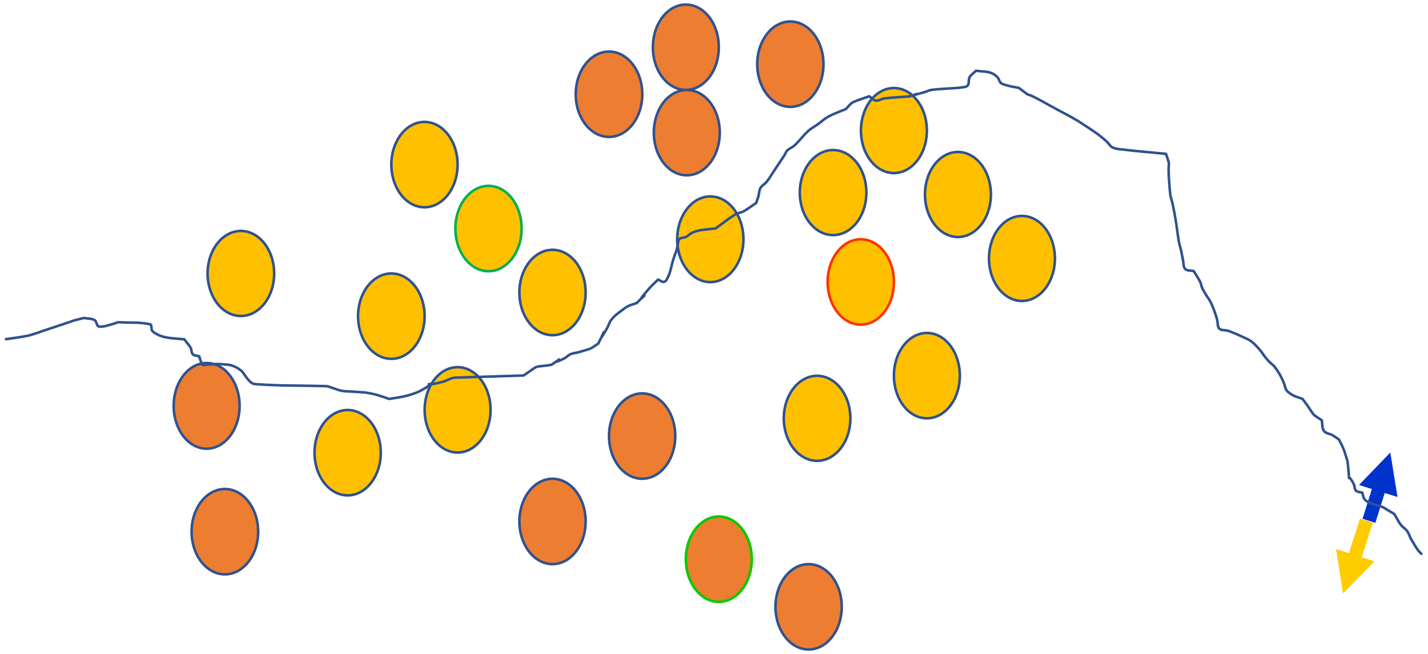
# The decision boundary perspective…

**Present a training instance / adjust the weights**



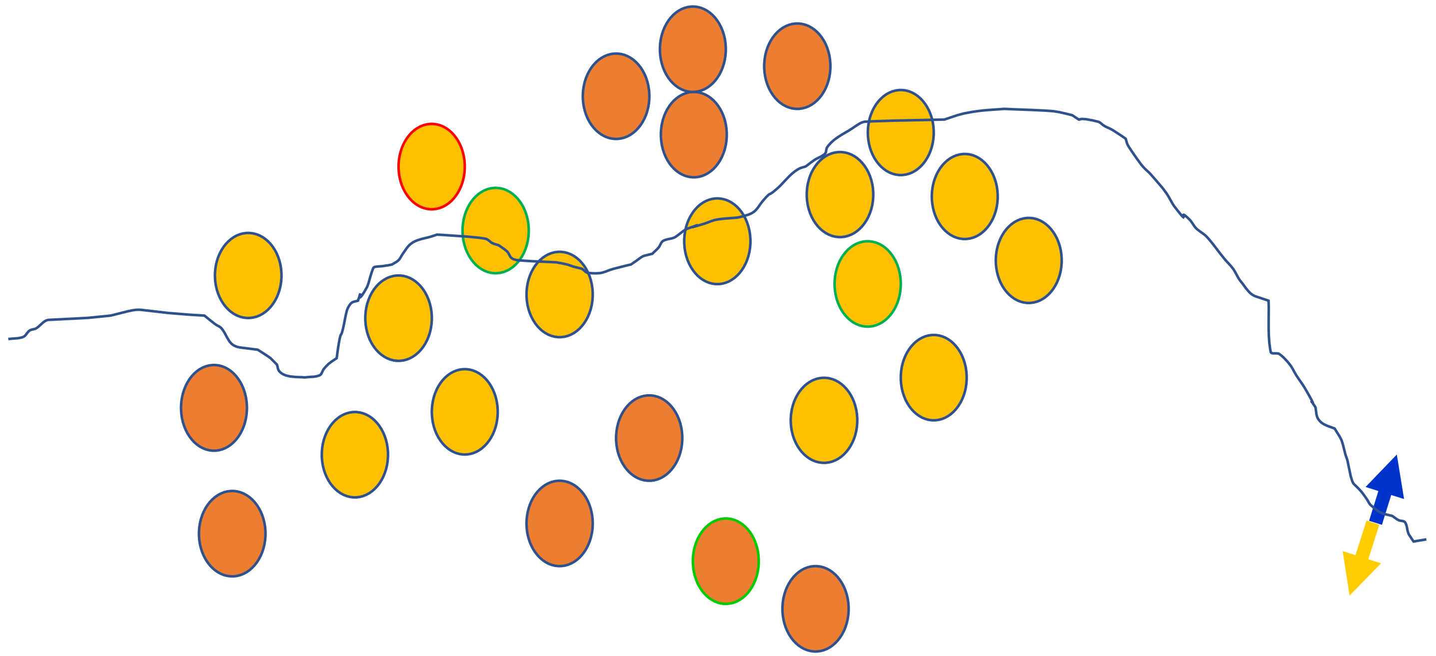
# The decision boundary perspective…

**Present a training instance / adjust the weights**



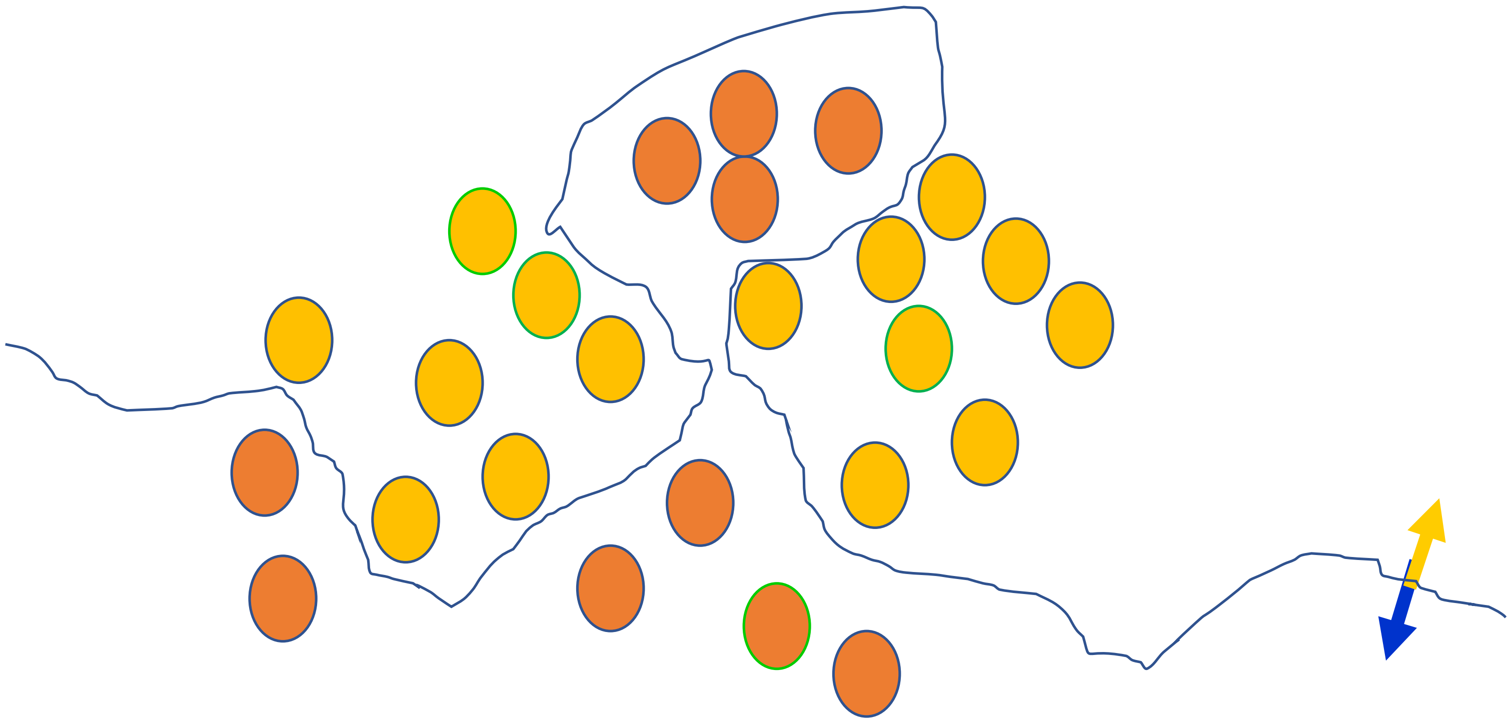
# The decision boundary perspective…

**Present a training instance / adjust the weights**



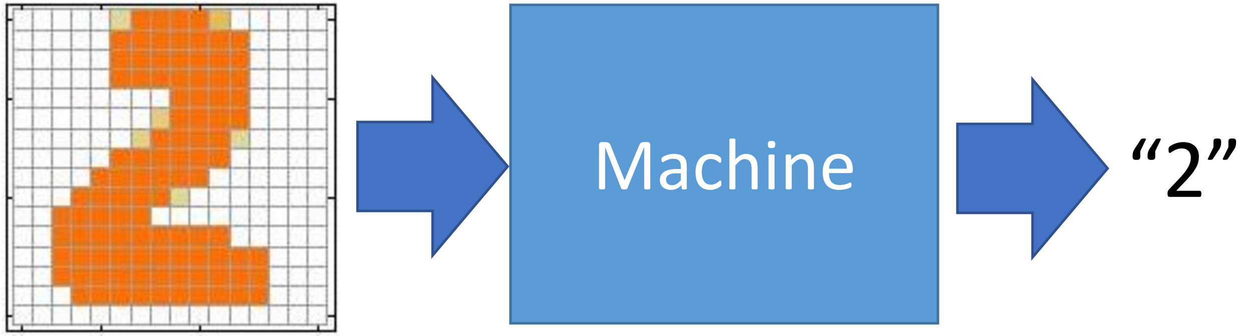
# The decision boundary perspective…

**Eventually ….**



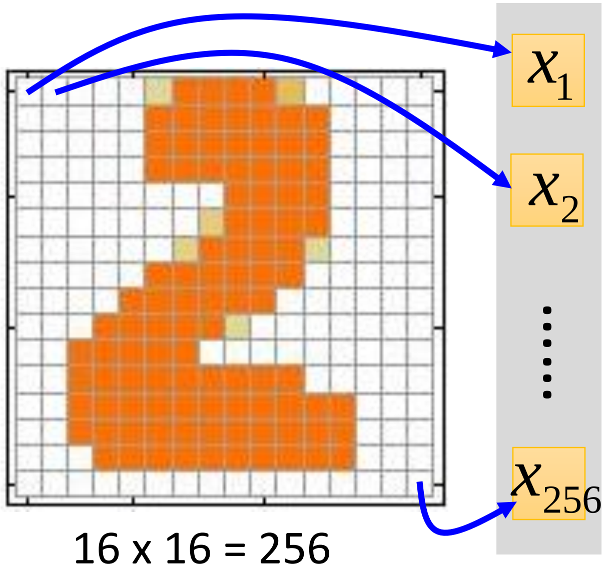
# Example Application

- Handwriting Digit Recognition



# Handwriting Digit Recognition

**Input**



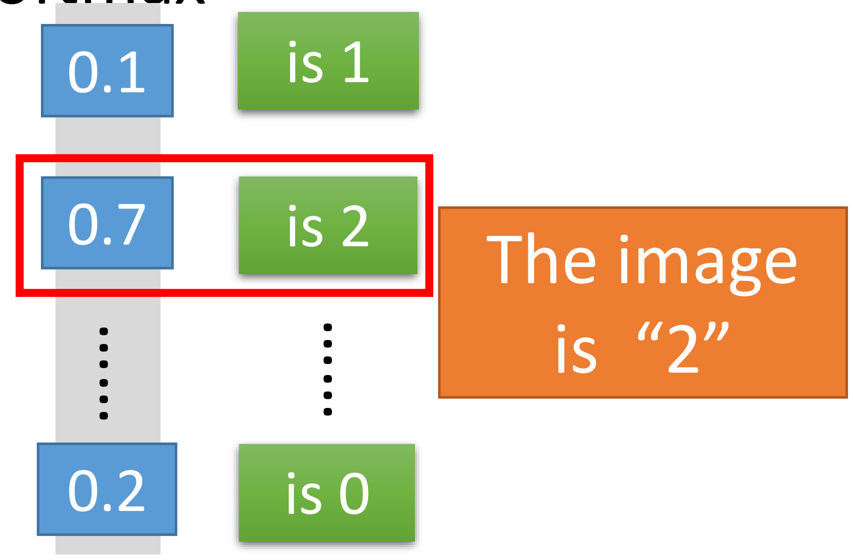

16 x 16 = 256

Ink → 1

No ink → 0

**Output**

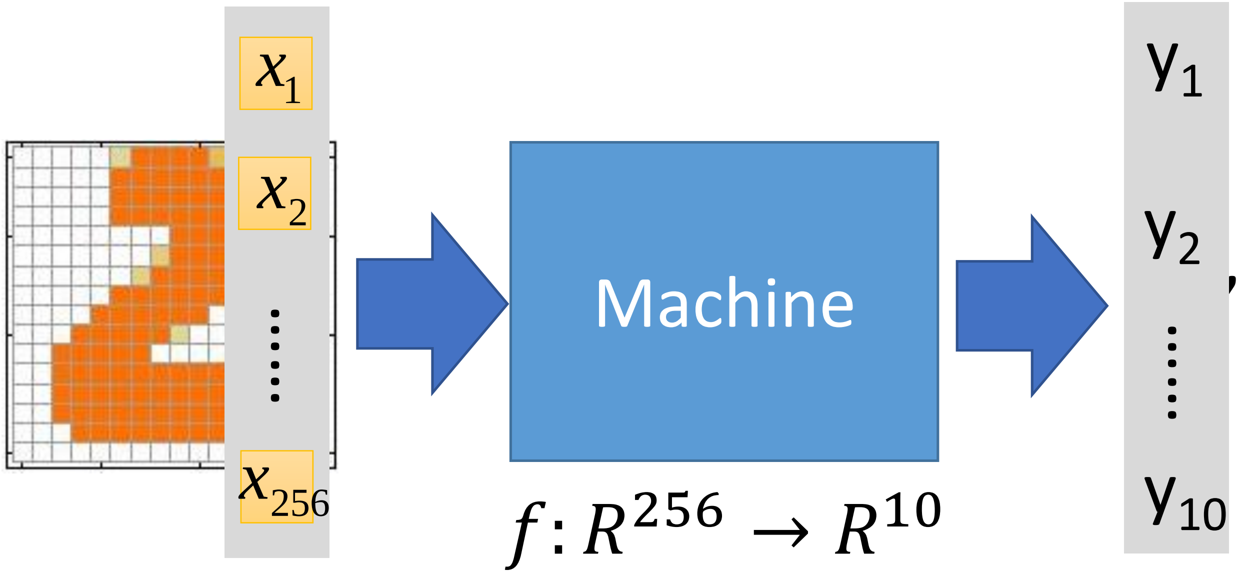
softmax




Each dimension represents the confidence of a digit.

# Example Application

- Handwriting Digit Recognition




$f: R^{256} \rightarrow R^{10}$

In deep learning, the function $f$ is represented by neural network

# RNN

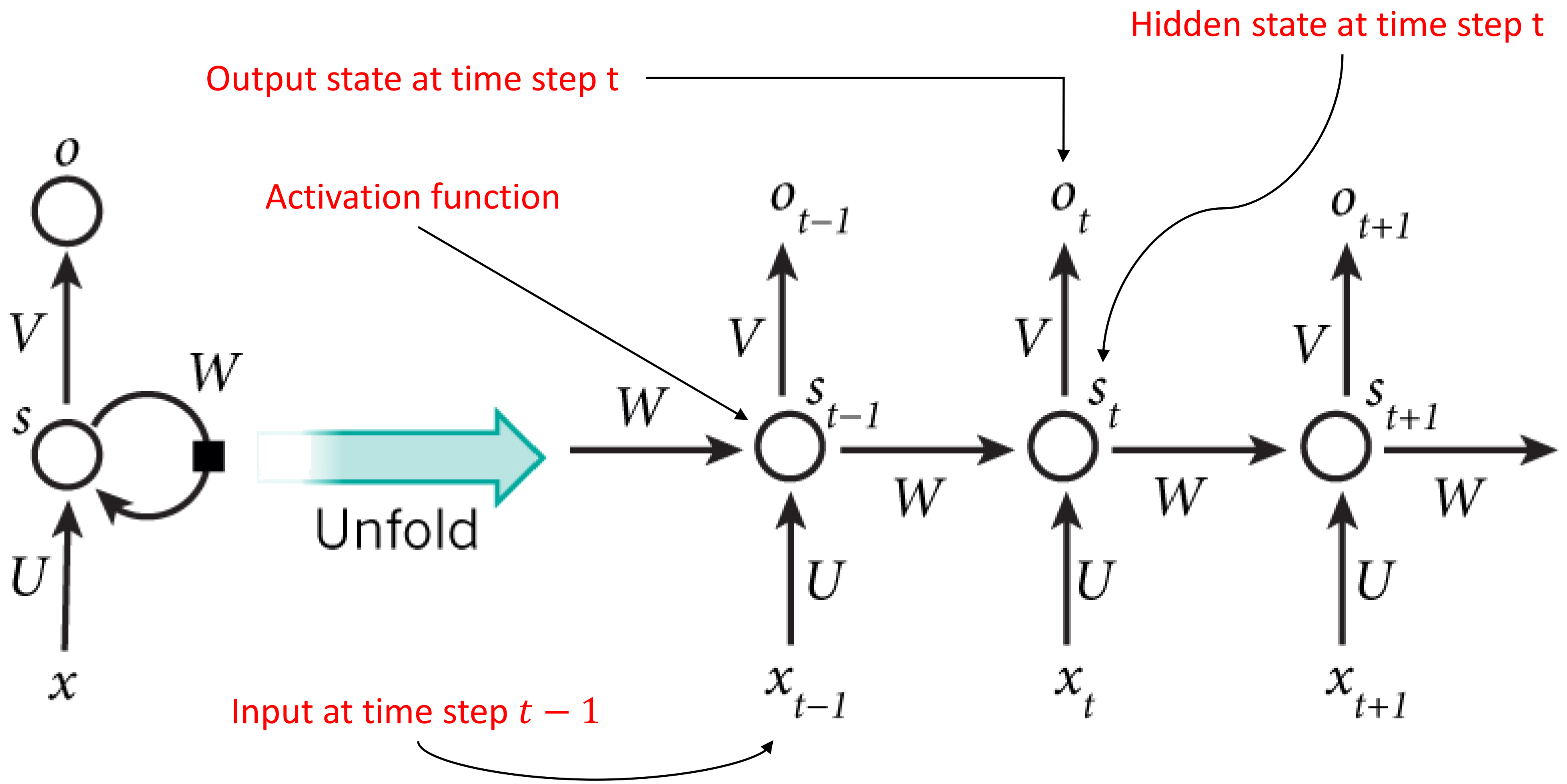
# Motivation

- Humans don't start their thinking from scratch every second
  - Thoughts have persistence
- Traditional neural networks can't characterize this phenomena
  - Ex: classify what is happening at every point in a movie
  - How a neural network can inform later events about the previous ones
- Recurrent neural networks address this issue
- How?

# What are RNNs?

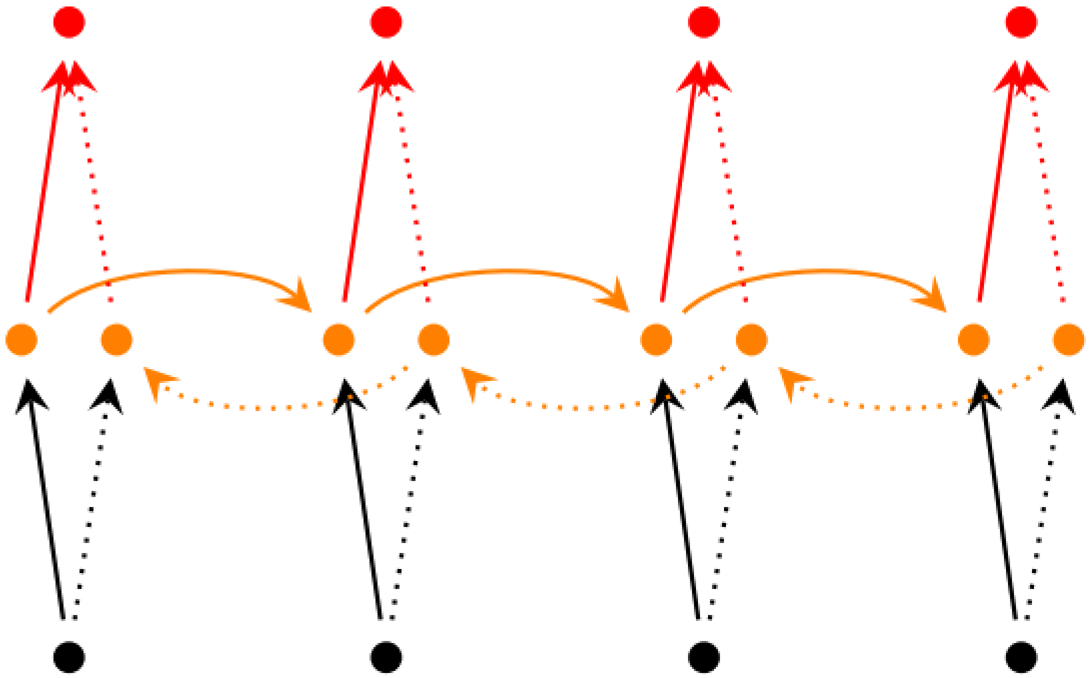
- Main idea is to make use of sequential information
- How RNN is different from neural network?
  - Vanilla neural networks assume all inputs and outputs are independent of each other
  - But for many tasks, that's a very bad idea
- What RNN does?
  - Perform the same task for every element of a sequence (that's what recurrent stands for)
  - Output depends on the previous computations!
- Another way of interpretation – RNNs have a "memory"
  - To store previous computations

# Recurrent Neural Networks



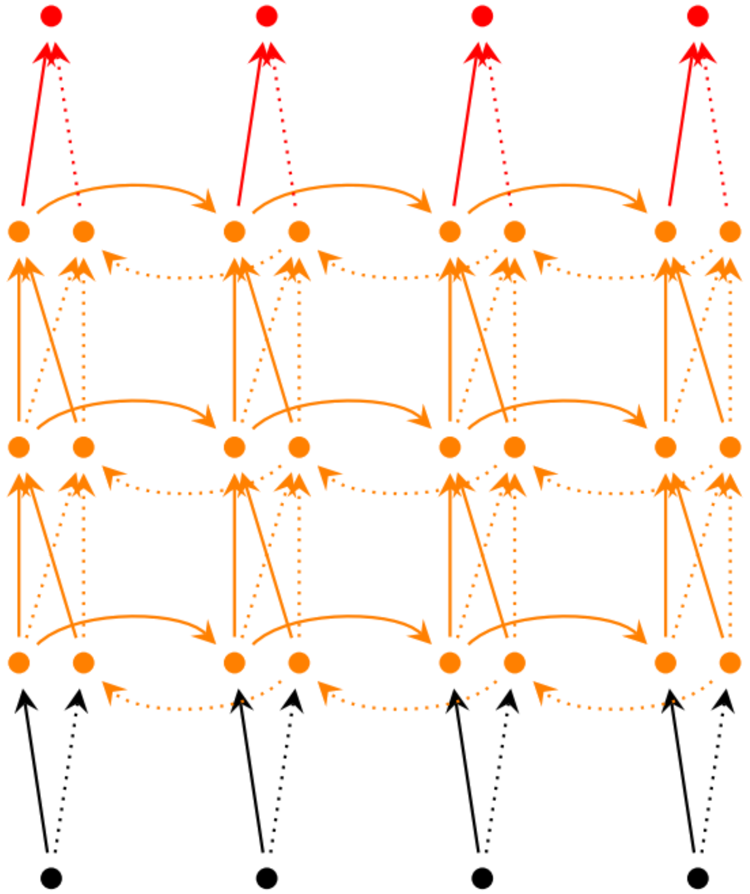
# RNN Extensions

- Bidirectional RNNs



# RNN Extensions

- Deep (Bidirectional) RNNs



# NLP Tasks

# NLP Tasks

1. Classify the entire document
   (“text categorization”)

# Sentiment classification


What features of the text could help predict # of stars?
(e.g., using a log-linear model) How to identify more?
Are the features hard to compute? (syntax? sarcasm?)

**An extremely versatile machine!**, November 22, 2006

By **Dr. Nickolas E. Jorgensen "njorgens3"**

**This review is from: Cuisinart DGB-600BC Grind & Brew, Brushed Chrome (Kitchen)**

This coffee-maker does so much! It makes weak, watery coffee! It grinds beans if you want it to! It inexplicably floods the entire counter with half-brewed coffee when you aren't looking! Perhaps it could be used to irrigate crops... It is time-consuming to clean, but in fairness I should also point out that the stainless-steel thermal carafe is a durable item that has withstood being hurled onto the floor in rage several times. And if all these features weren't enough, it's pretty expensive too. If faced with the choice between having a car door repeatedly slamming into my genitalia and buying this coffee-maker, I'd unhesitatingly choose the Cuisinart! The coffee would be lousy, but at least I could still have children...

# Other text categorization tasks

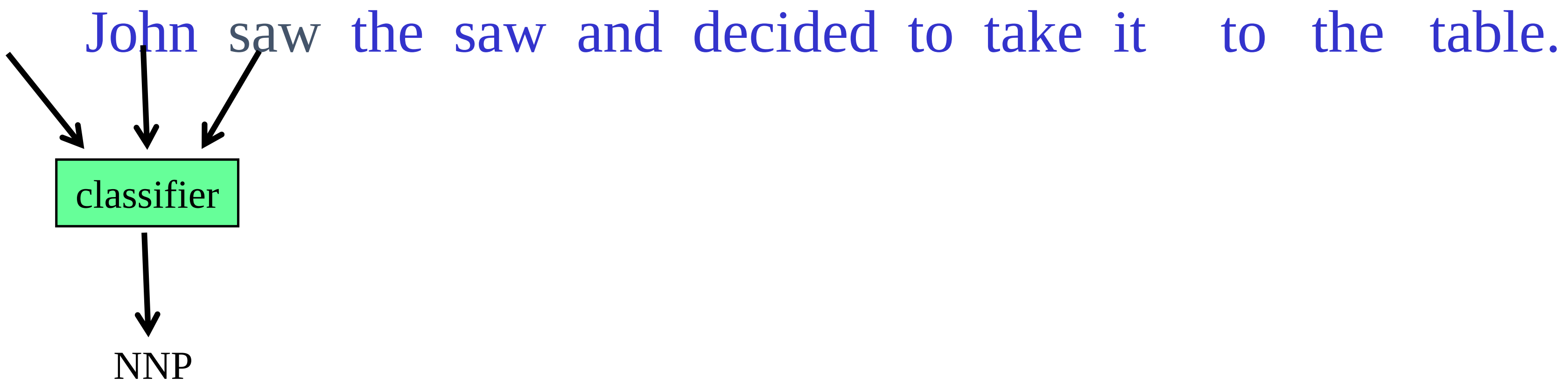
- Is it spam? (see features)
- What medical billing code for this visit?
- What grade, as an answer to this essay question?
- Is it interesting to this user?
  - News filtering; helpdesk routing
- Is it interesting to this NLP program?
  - If it’s Spanish, translate it from Spanish
  - If it’s subjective, run the sentiment classifier
  - If it’s an appointment, run information extraction
- Where should it be filed?
  - Which mail folder? (work, friends, junk, urgent ...)
  - Yahoo! / Open Directory / digital libraries

# NLP Tasks

1. Classify the entire document
2. Classify individual word tokens

# Sequence Labeling as Classification

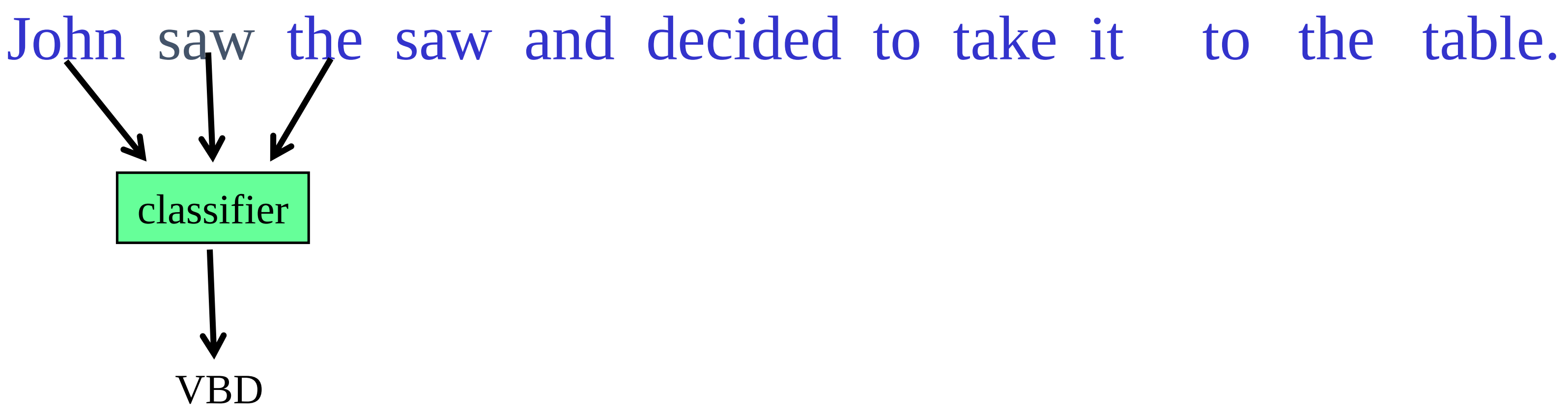
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).




Slide from Ray Mooney

# Sequence Labeling as Classification

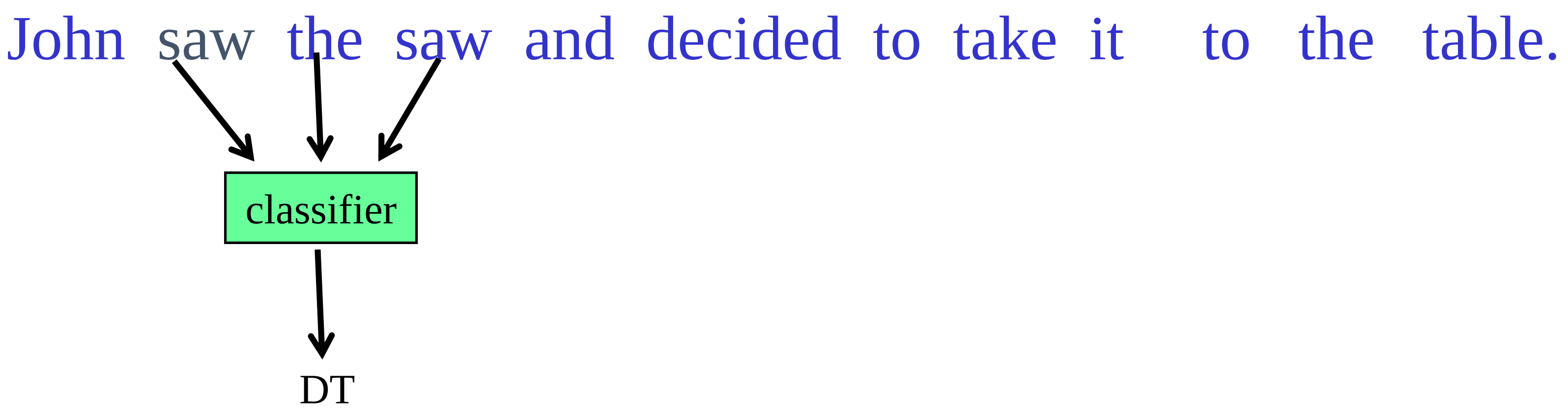
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).




Slide from Ray Mooney

# Sequence Labeling as Classification

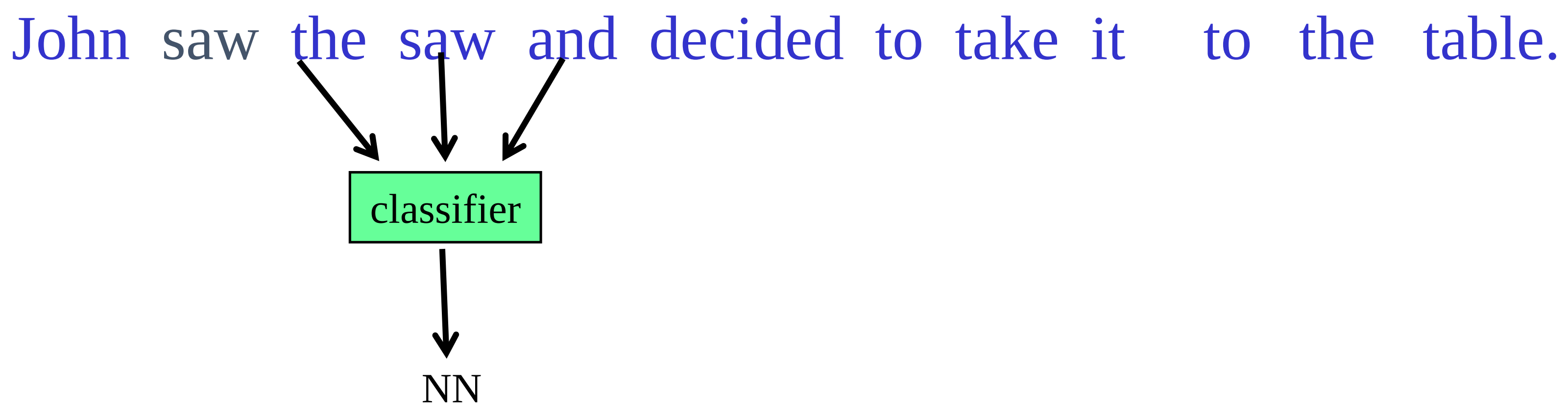
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).




Slide from Ray Mooney

# Sequence Labeling as Classification

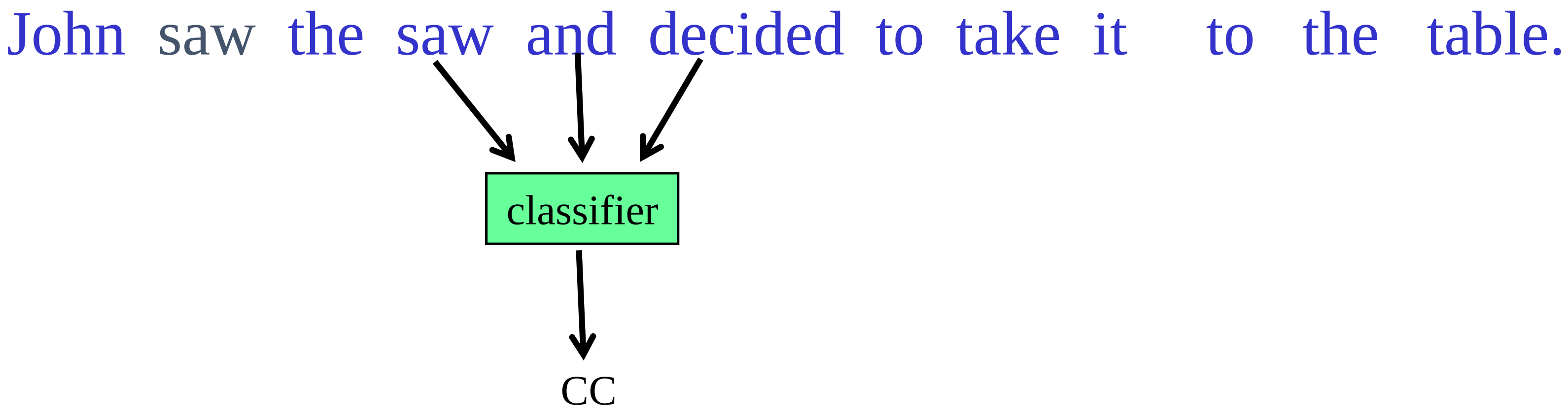
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).




Slide from Ray Mooney

# Sequence Labeling as Classification

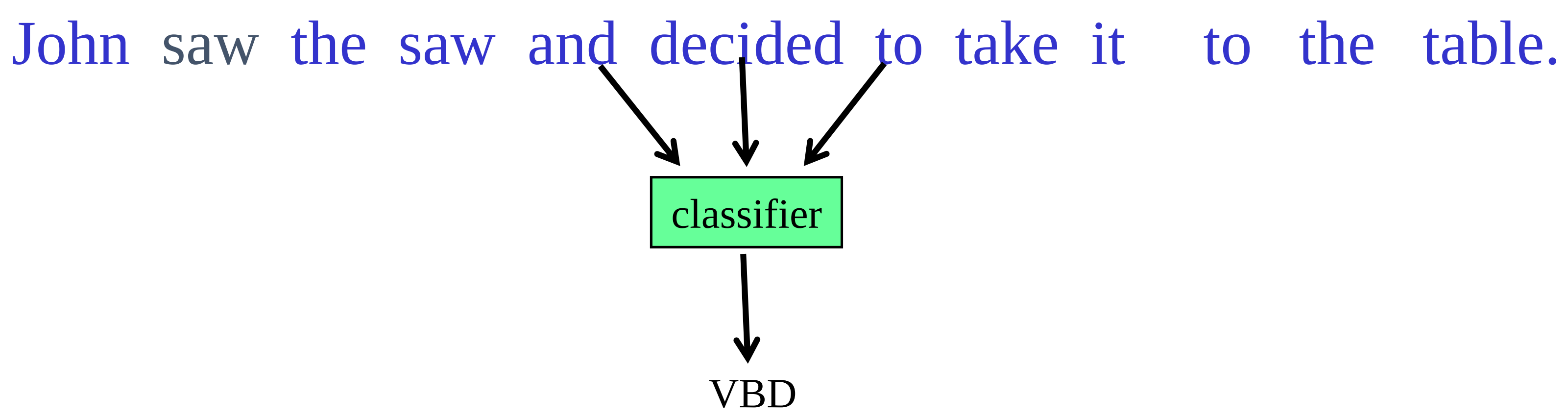
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).




Slide from Ray Mooney

# Sequence Labeling as Classification

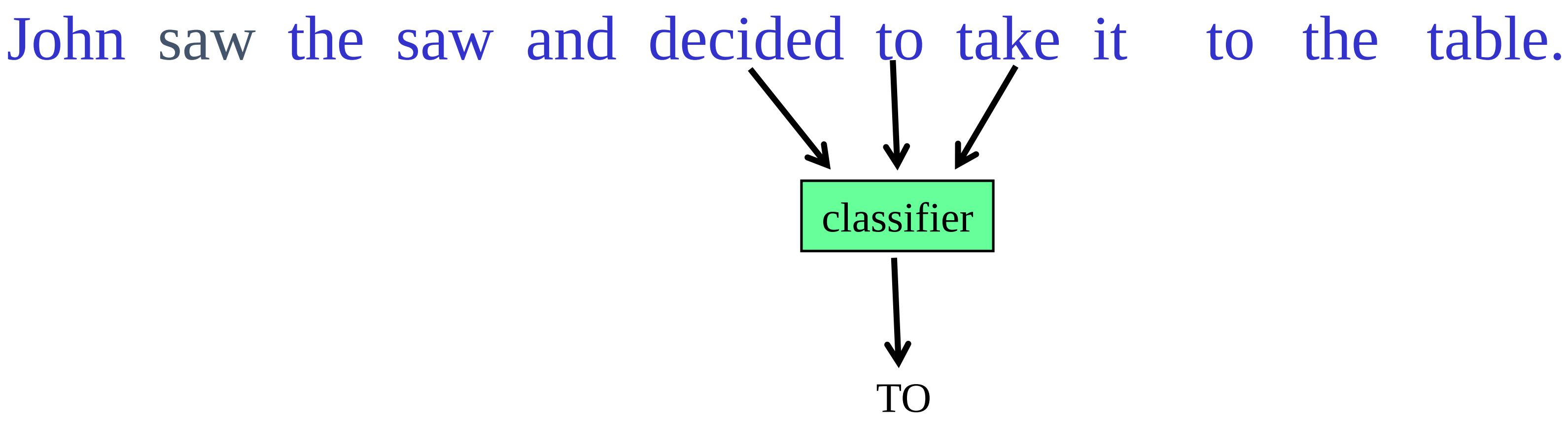
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).




Slide from Ray Mooney

# Sequence Labeling as Classification

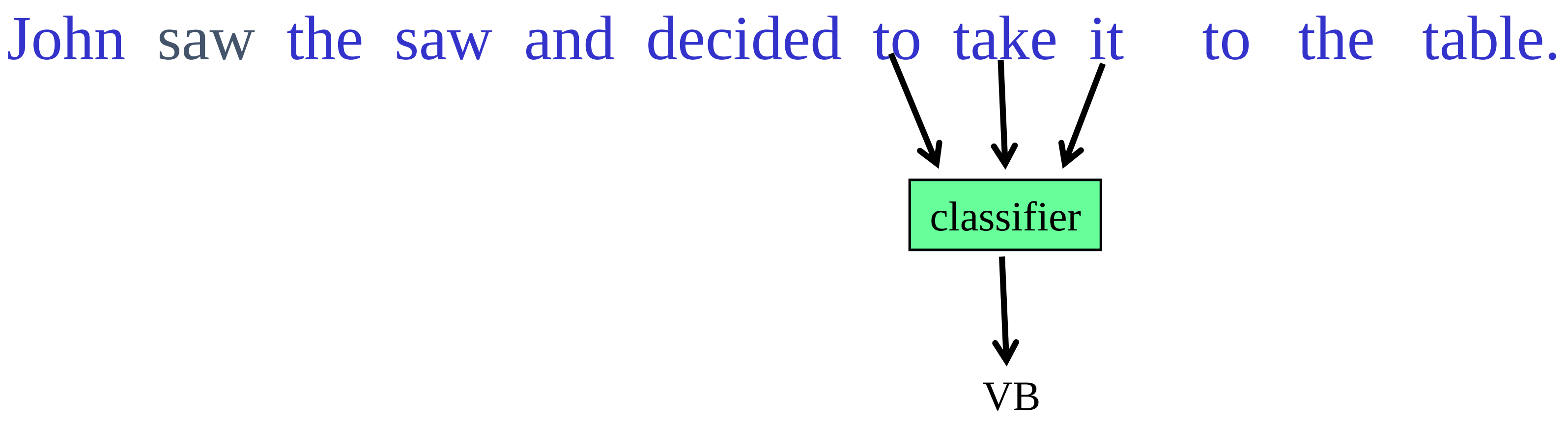
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).




Slide from Ray Mooney

# Sequence Labeling as Classification

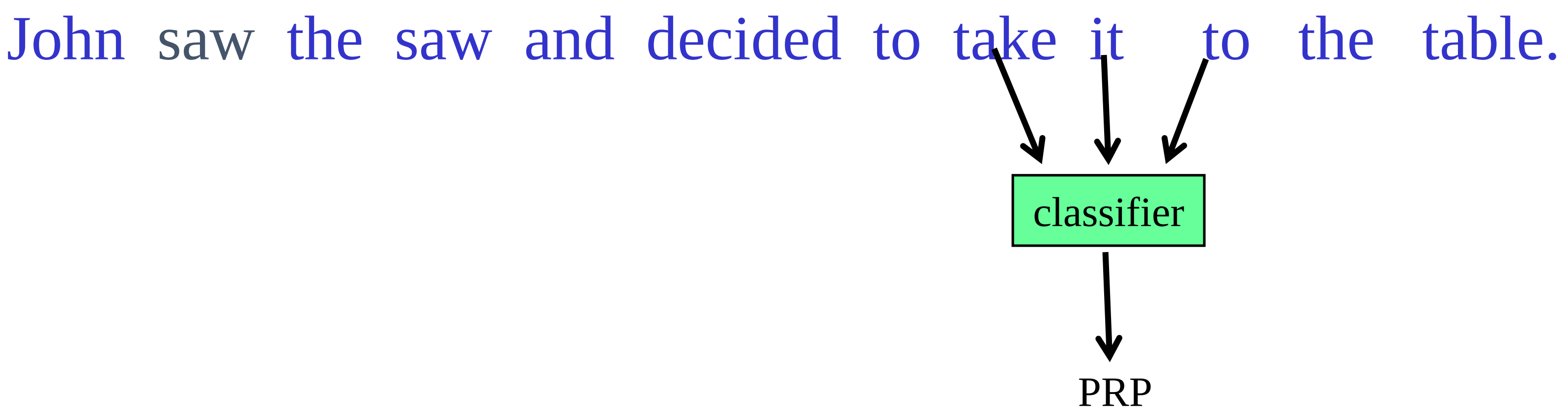
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).




Slide from Ray Mooney

# Sequence Labeling as Classification

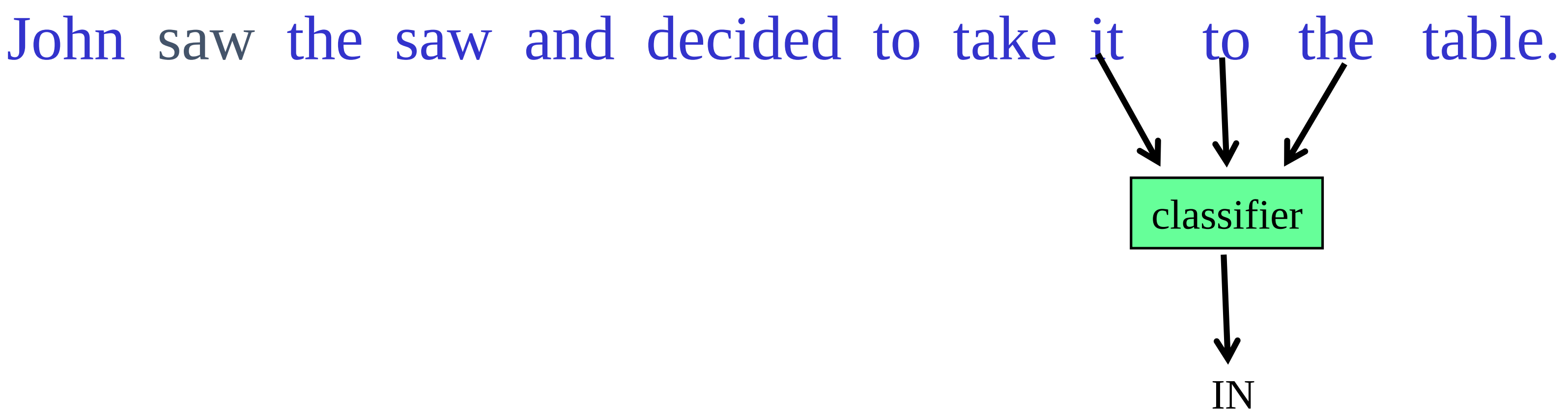
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).




Slide from Ray Mooney

# Sequence Labeling as Classification

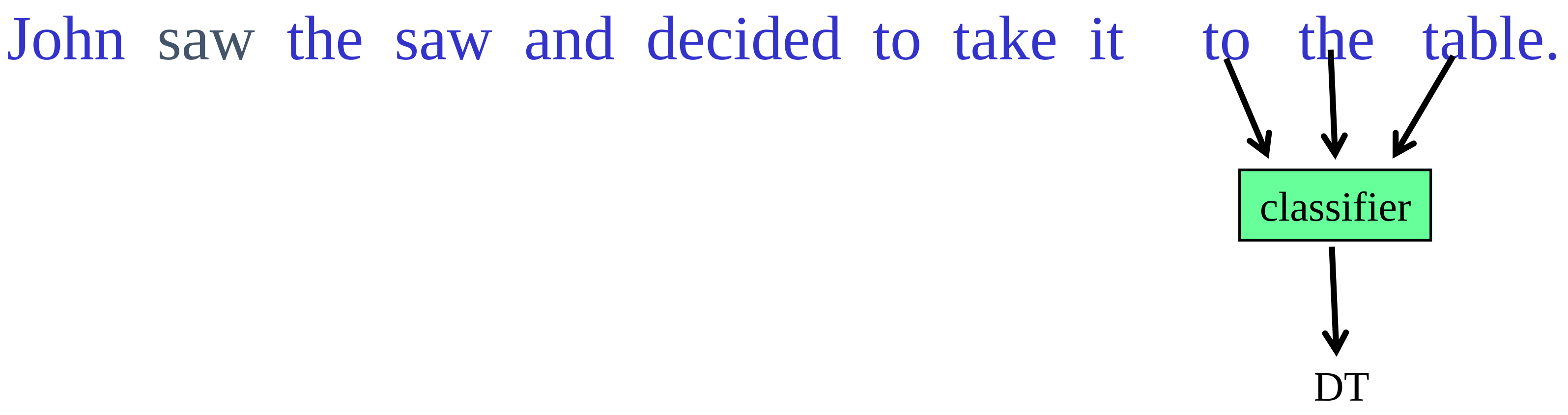
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).




Slide from Ray Mooney

# Sequence Labeling as Classification

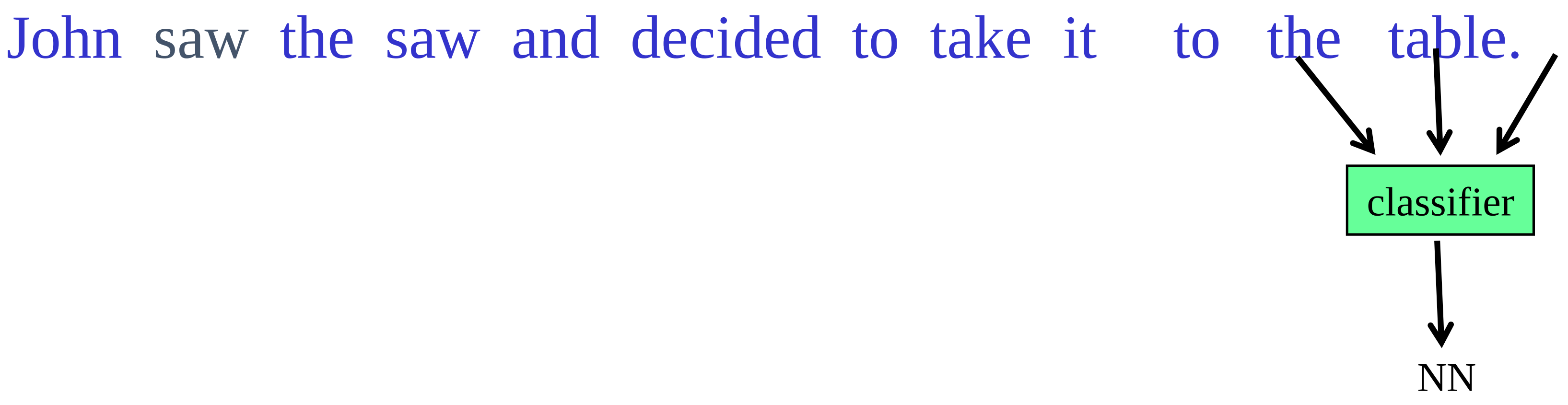
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).




Slide from Ray Mooney

# Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



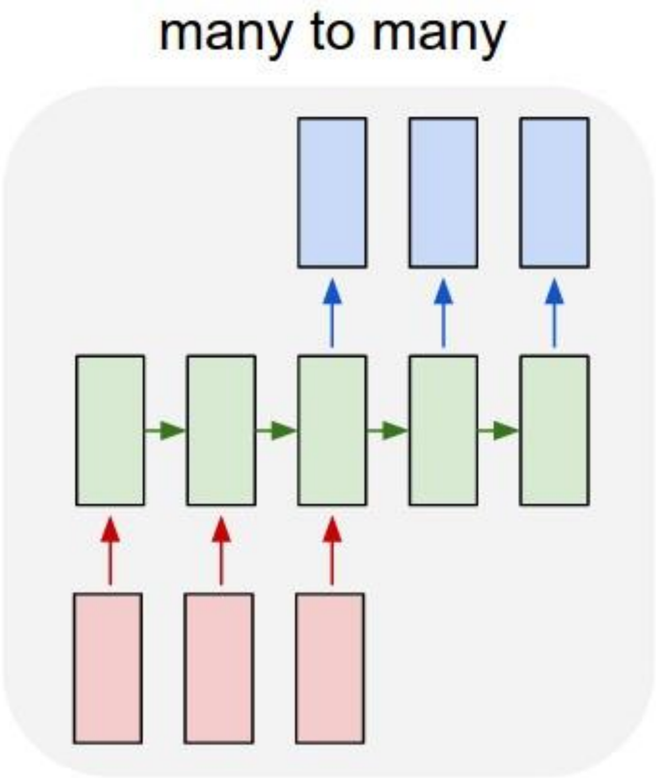

Slide from Ray Mooney

# NLP Tasks

1. Classify the entire document
2. Classify individual word tokens
3. Generating new text

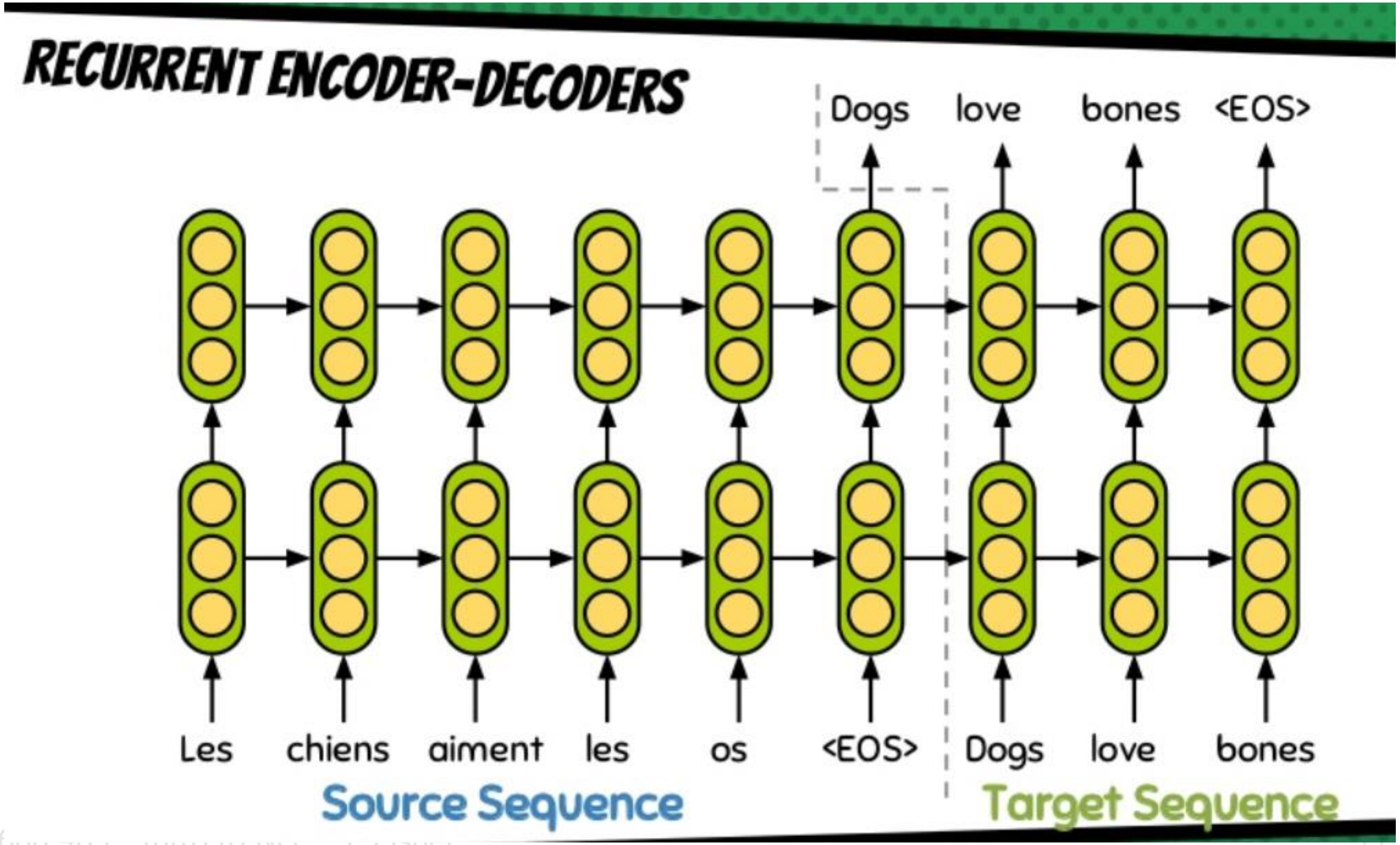
# Generating new text

- Architecture



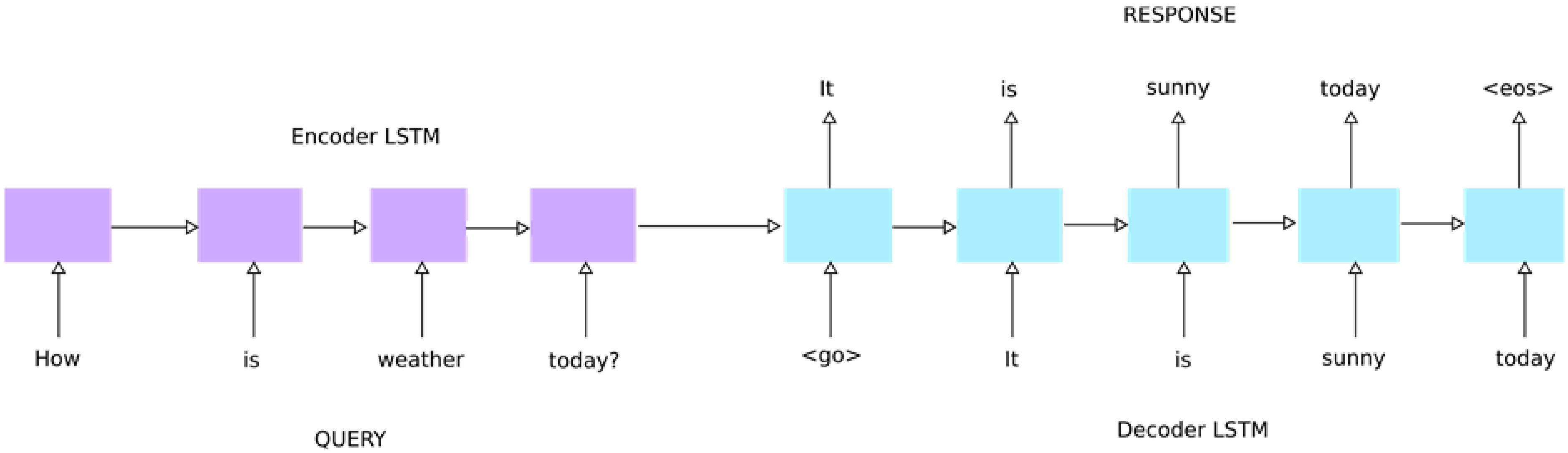
# Generating new text

- Machine Translation



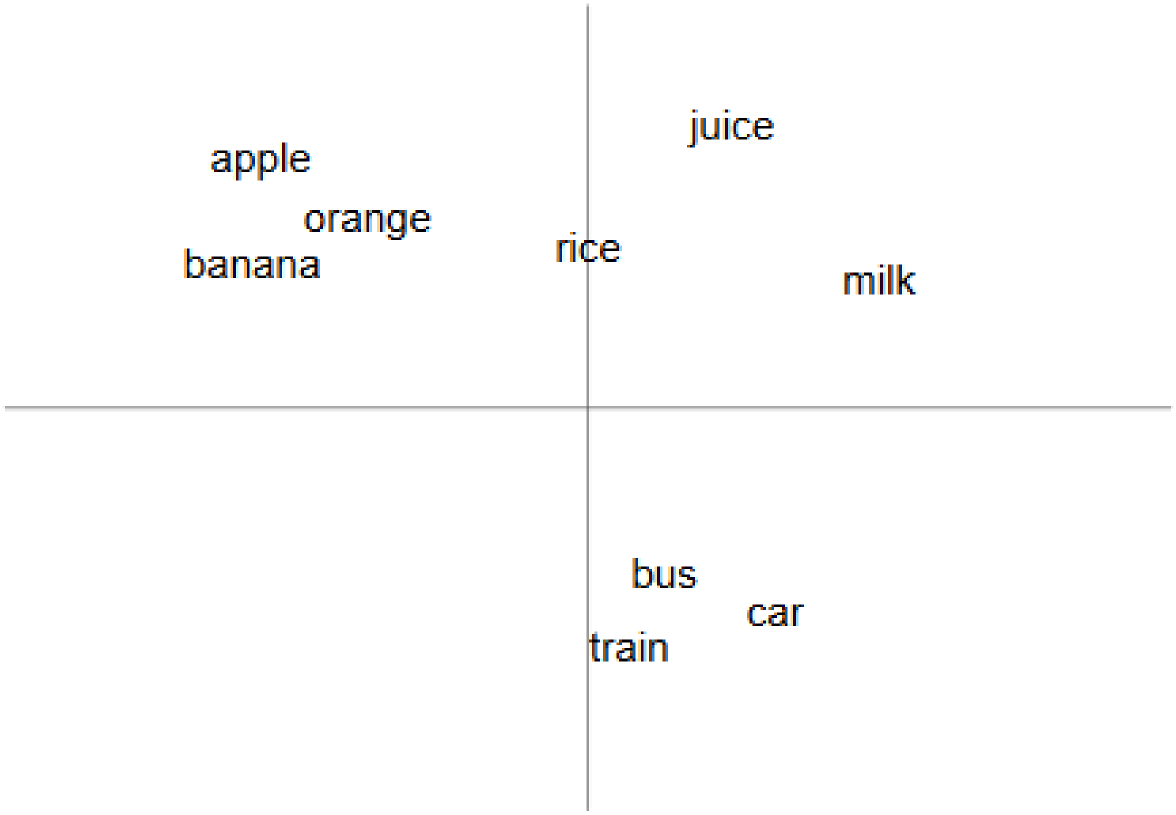
# Generating new text

- Question Answering



# Word2Vec and Word embeddings

# *“A word is known by the company it keeps”*



# Word Representations

| Traditional Method - Bag of Words Model | Word Embeddings |
|---|---|
| • Uses one hot encoding<br><br>• Each word in the vocabulary is represented by one bit position in a HUGE vector.<br><br>• For example, if we have a vocabulary of 10,000 words, and "Hello" is the $4^{th}$ word in the dictionary, it would be represented by: [ 0 0 0 1 0 0 . . . . . . . . 0 0 0 0 ]<br><br>• Context information is not utilized | • Stores each word in as a point in space, where it is represented by a vector of fixed number of dimensions (generally 300)<br><br>• Unsupervised, built just by reading huge corpus<br><br>• For example, "Hello" might be represented as :<br>[0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02]<br><br>• Dimensions are basically projections along different axes, more of a mathematical concept. |

# The Power of Word Vectors

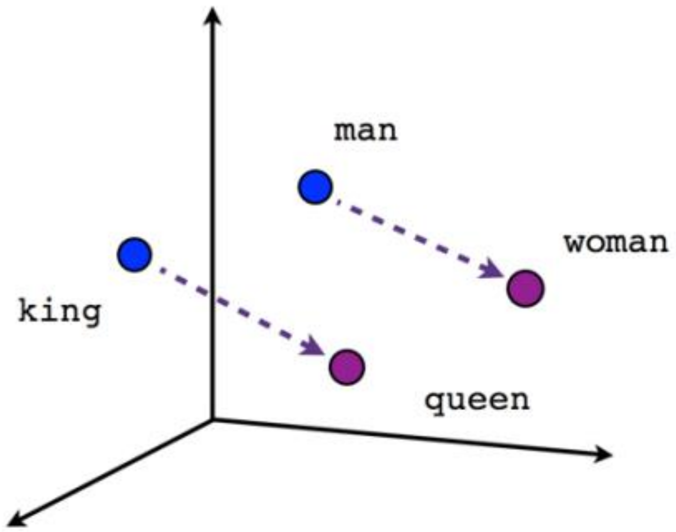
- They provide a fresh perspective to ***ALL*** problems in NLP, and not just solve one problem.

- The need for unsupervised learning . (Supervised learning tends to be excessively dependant on hand-labelled data and often does not scale)

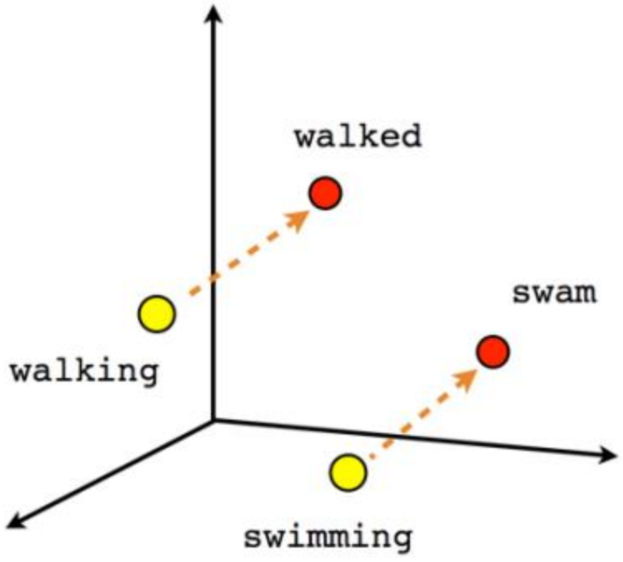
# some of famous word embeddings

- Word2Vec
- Glove
- fasttext
- ELMO
- BERT

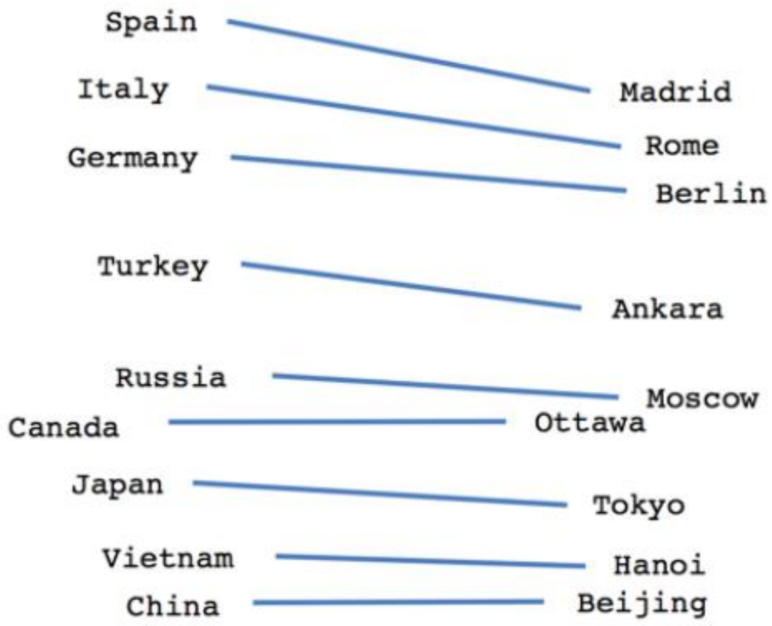
# Examples




Male-Female




Verb tense




Country-Capital

vector[Queen] =  vector[King]  - vector[Man] + vector[Woman]

So, how exactly does Word Embedding ‘solve all problems in NLP’?

# Applications of Word Vectors

1. Word Similarity

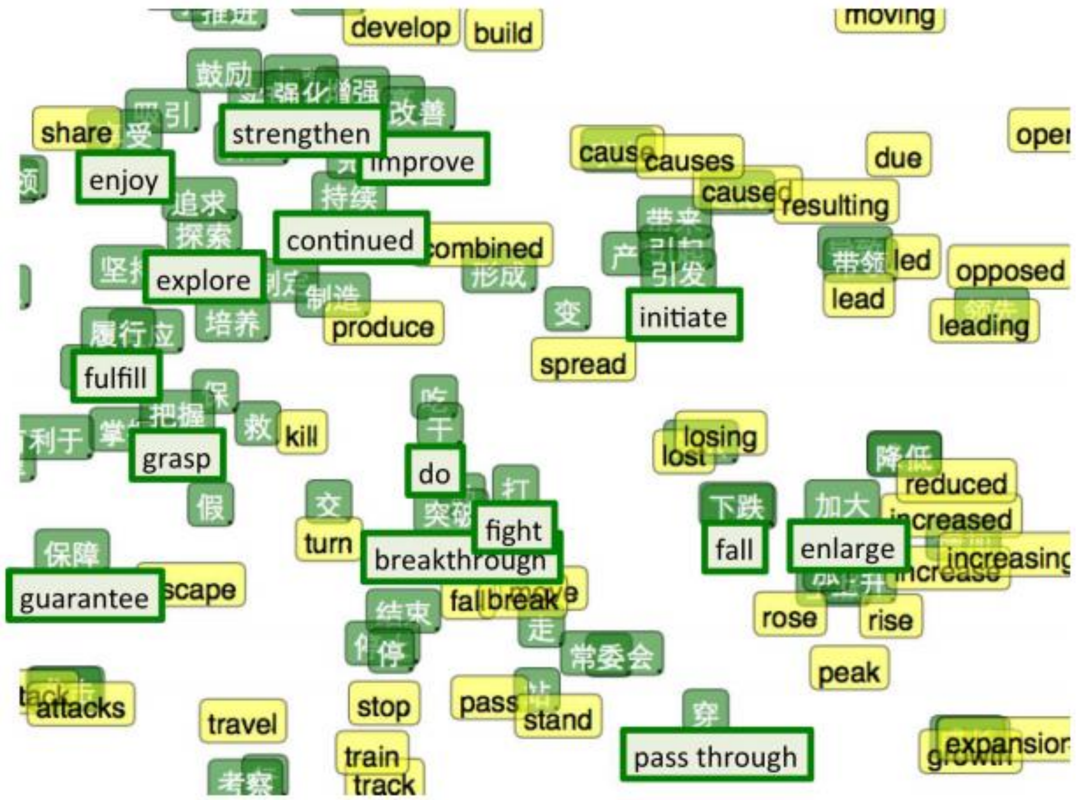
Classic Methods : Edit Distance, WordNet, Porter's Stemmer, Lemmatization using dictionaries

- Easily identifies similar words and synonyms since they occur in similar contexts
- Stemming (thought -> think)
- Inflections, Tense forms
- *eg. Think, thought, ponder, pondering,*
- *eg. Plane, Aircraft, Flight*

# Applications of Word Vectors

## 2. Machine Translation

Classic Methods： Rule-based machine translation, morphological t



# Applications of Word Vectors

## 3. Part-of-Speech and Named Entity Recognition

Classic Methods : Sequential Models (MEMM , Conditional Random Fields), Logistic Regression

| | POS WSJ (acc.) | NER CoNLL (F1) |
|---|---|---|
| State-of-the-art* | 97.24 | 89.31 |
| Supervised NN | **96.37** | **81.47** |
| Unsupervised pre-training followed by supervised NN** | **97.20** | **88.87** |
| + hand-crafted features*** | 97.29 | 89.59 |

# Applications of Word Vectors

## 3. Named Entity Recognition

Classic Methods : Sequential Models (MEMM , Conditional Random Fields), Logistic Regression

| | Arman | | Peyma | |
|---|---|---|---|---|
| | word | phrase | word | phrase |
| Bokaei and Mahmoudi (Bokaei and Mahmoudi, 2018) | 81.50 | 76.79 | - | - |
| Shahshahani et al.(Shahshahani et al., 2018) | - | - | 80.0 | - |
| Beheshti-NER (Our Model) | **84.03** | **79.93** | **90.59** | **87.62** |

# Applications of Word Vectors

## 4. Relation Extraction

Classic Methods : OpenIE, Linear programing models, Bootstrapping

| Relationship | Example 1 | Example 2 | Example 3 |
|---|---|---|---|
| France - Paris | Italy: Rome | Japan: Tokyo | Florida: Tallahassee |
| big - bigger | small: larger | cold: colder | quick: quicker |
| Miami - Florida | Baltimore: Maryland | Dallas: Texas | Kona: Hawaii |
| Einstein - scientist | Messi: midfielder | Mozart: violinist | Picasso: painter |
| Sarkozy - France | Berlusconi: Italy | Merkel: Germany | Koizumi: Japan |
| copper - Cu | zinc: Zn | gold: Au | uranium: plutonium |
| Berlusconi - Silvio | Sarkozy: Nicolas | Putin: Medvedev | Obama: Barack |
| Microsoft - Windows | Google: Android | IBM: Linux | Apple: iPhone |
| Microsoft - Ballmer | Google: Yahoo | IBM: McNealy | Apple: Jobs |
| Japan - sushi | Germany: bratwurst | France: tapas | USA: pizza |

# Applications of Word Vectors

## 5. Sentiment Analysis

Classic Methods : Naive Bayes, Random Forests/SVM

- Classifying sentences as positive and negative
- Building sentiment lexicons using seed sentiment sets
- No need for classifiers, we can just use cosine distances to compare unseen reviews to known reviews.

```
Enter word or sentence (EXIT to break): sad

Word: sad  Position in vocabulary: 4067

                                        Word       Cosine distance
------------------------------------------------------------------------
                                   saddening              0.727309
                                         Sad              0.661083
                                    saddened              0.660439
                                heartbreaking             0.657351
                                disheartening             0.650732
                                Meny_Friedman             0.648706
                         parishioner_Pat_Patello          0.647586
                                   saddens_me             0.640712
                                  distressing             0.639909
                            reminders_bobbing             0.635772
                              Turkoman_Shiites            0.635577
                                      saddest             0.634551
                                  unfortunate             0.627209
                                        sorry             0.619405
                                  bittersweet             0.617521
                                       tragic             0.611279
                                    regretful             0.603472
```

# Applications of Word Vectors

## 6. Co-reference Resolution

- Chaining entity mentions across multiple documents - can we find and unify the multiple contexts in which mentions occurs?

## 7. Clustering

- Words in the same class naturally occur in similar contexts, and this feature vector can directly be used with any conventional clustering algorithms (K-Means, agglomerative, etc). Human doesn't have to waste time hand-picking useful word features to cluster on.

## 8. Semantic Analysis of Documents

- Build word distributions for various topics, etc.

# Building these magical vectors . . .

- How do we actually build these super-intelligent vectors, that seem to have such magical powers?

- How to find a word’s friends?

- We will discuss the most famous method to build such lower-dimension vector representations for words based on their context
  - word2vec (*Google)*

# Word2Vec

**Efficient Estimation of Word Representations in Vector Space**

**Tomas Mikolov**
Google Inc., Mountain View, CA
tmikolov@google.com

**Kai Chen**
Google Inc., Mountain View, CA
kaichen@google.com

**Greg Corrado**
Google Inc., Mountain View, CA
gcorrado@google.com

**Jeffrey Dean**
Google Inc., Mountain View, CA
jeff@google.com

# Context windows

- Context can be anything – a surrounding n-gram, a randomly sampled set of words from a fixed size window around the word

For example, assume context is defined as the word following a word.

i.e. $\boldsymbol{context(w_i) = w_{i+1}}$

Corpus : I ate the cat

Training Set : I|ate, ate|the , the|cat, cat|.

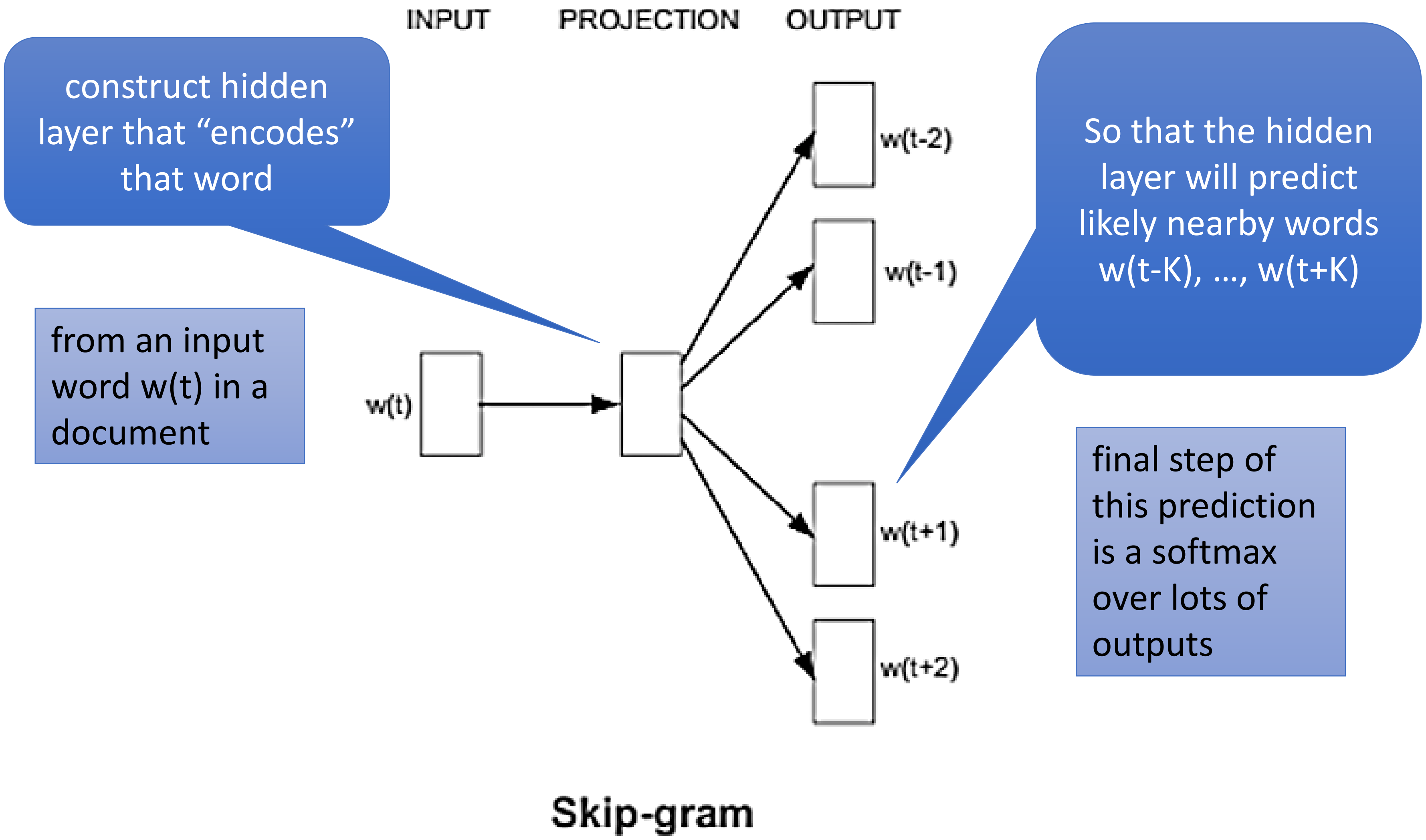
# Training Data

1. eat|apple
2. eat|orange
3. eat|rice
4. drink|juice
5. drink|milk
6. drink|water
7. orange|juice
8. apple|juice
9. rice|milk
10. milk|drink
11. water|drink
12. juice|drink

<u>Corpus:</u>

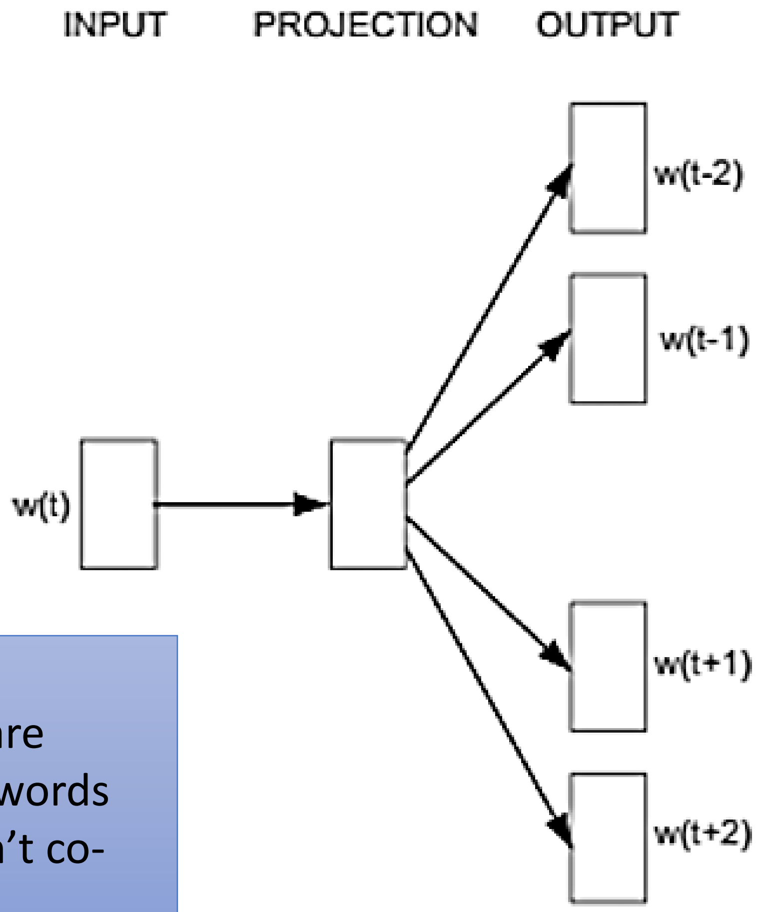
1. Milk and Juice are drinks
2. Apples, Oranges and Rice can be eaten
3. Apples and Orange are also juices
4. Rice milk is a actually a type of milk!

# Basic idea behind skip-gram embeddings



# Basic idea behind skip-gram embeddings

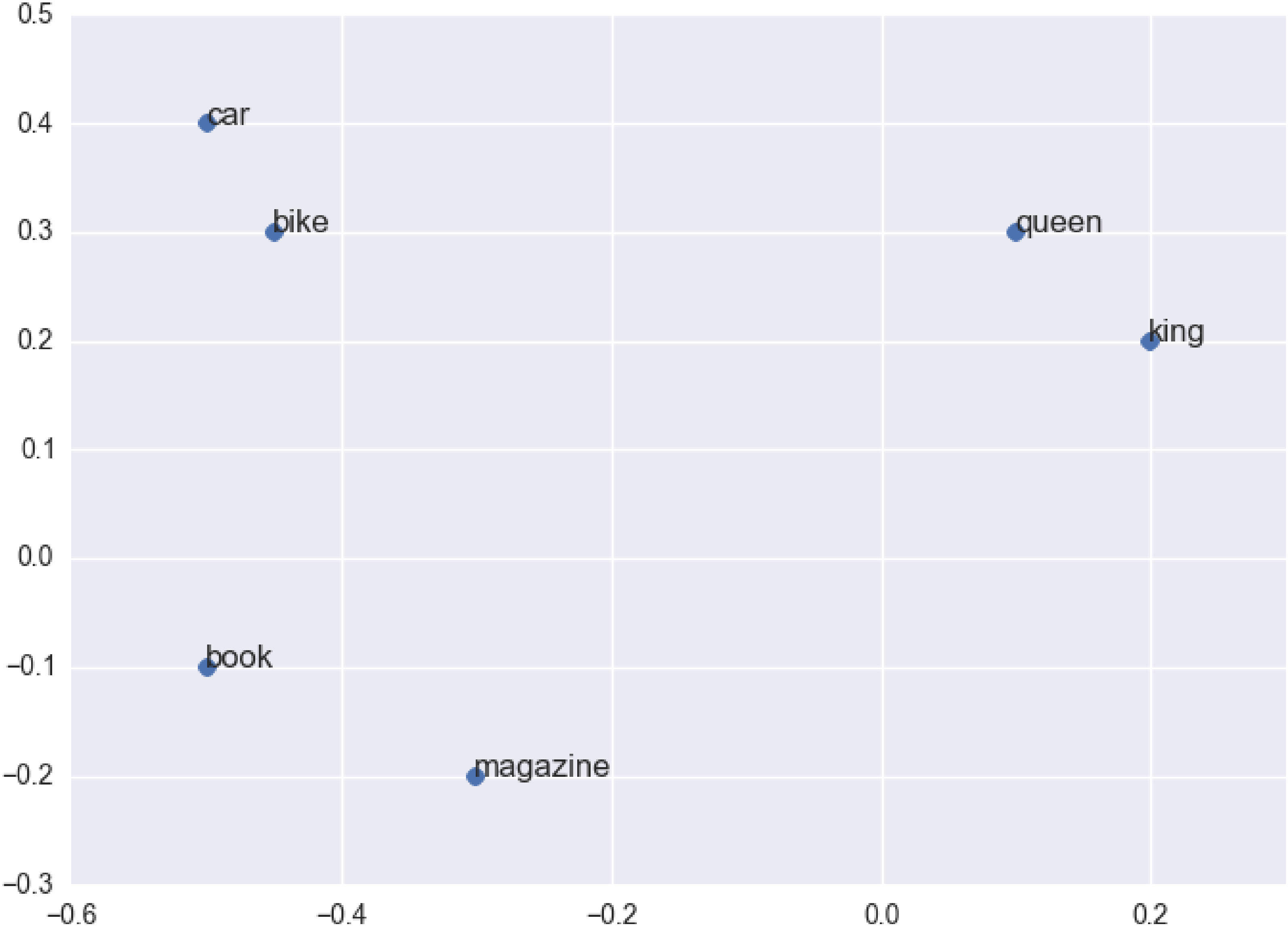
Training data: **positive** examples are pairs of words w(t), w(t+j) that co-occur




**Skip-gram**

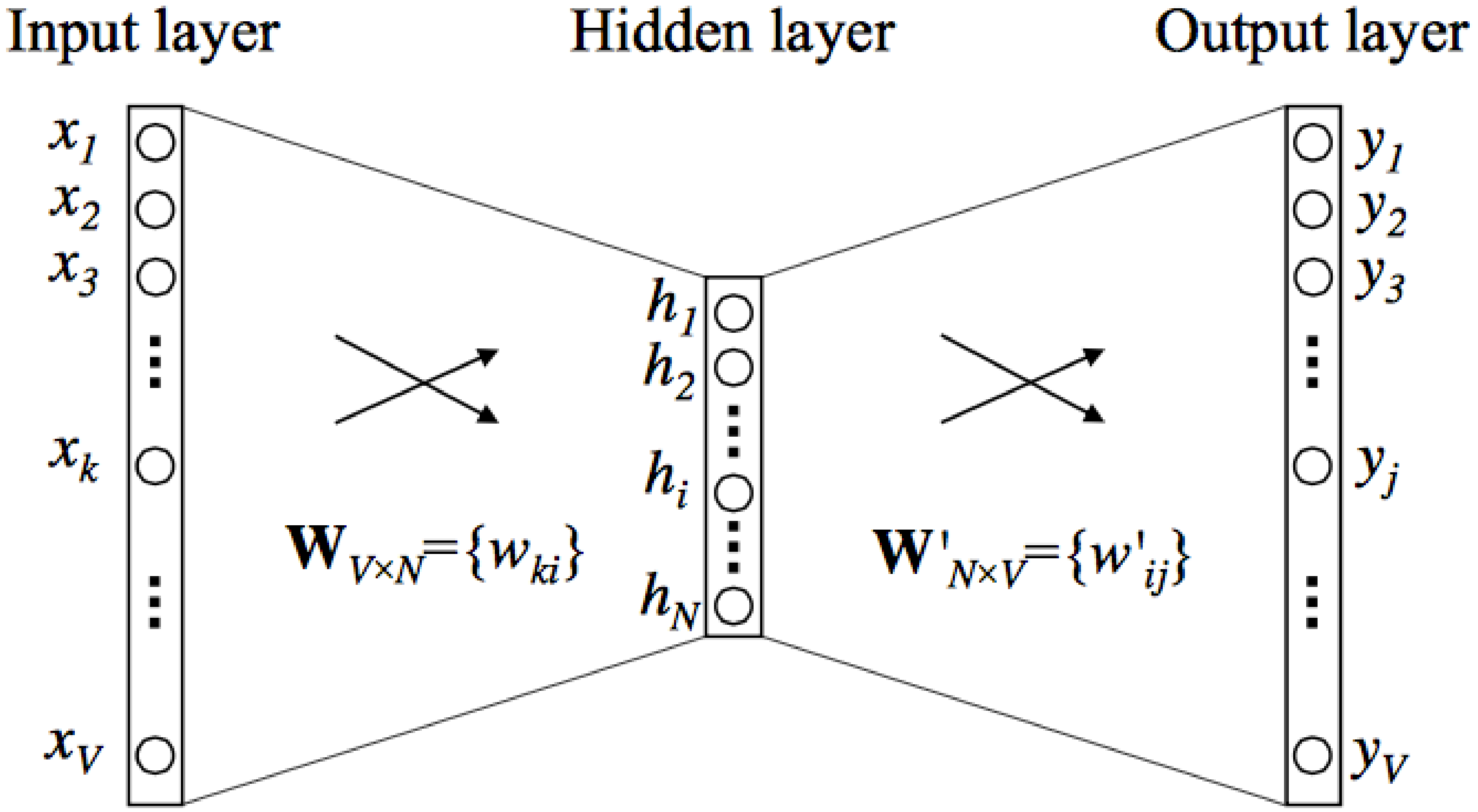
You want to train over a very large corpus (100M words+) and hundreds+ dimensions

Training data: negative examples are **samples** of pairs of words w(t), w(t+j) that don't co-occur



car
bike
queen
king
book
magazine
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.6
−0.4
−0.2
0.0
0.2

# network


Input layer
Hidden layer
Output layer
$x_1$
$x_2$
$x_3$
$x_k$
$x_V$
$h_1$
$h_2$
$h_i$
$h_N$
$y_1$
$y_2$
$y_3$
$y_j$
$y_V$
$\mathbf{W}_{V\times N}=\{w_{ki}\}$
$\mathbf{W}'_{N\times V}=\{w'_{ij}\}$

# Contextual word embedding

BERT

ERNIE


ElMO

# idea

meaning of words changes based on context.

examples :

- The plane took off at exactly nine o’clock.
- The **plane** surface is a must for any cricket pitch.
- **Plane** geometry is fun to study.

# Transformer *in* Language model = BERT