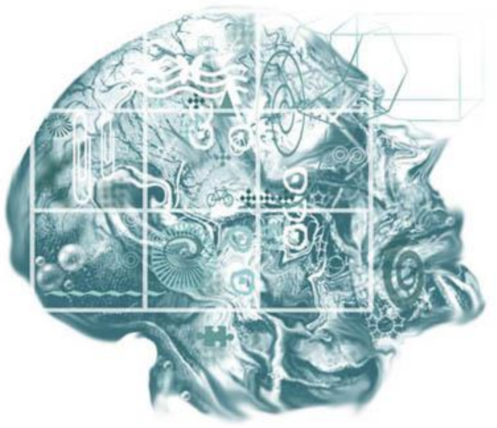


บทที่ 6 การประมวลผลภาษาธรรมชาติเชิงการใช้งาน

(Natural Language Processing)

นำเสนอโดย

ผู้ช่วยศาสตราจารย์จุฑาวุฒิ จันทรมาลี

หลักสูตรวิทยาการคอมพิวเตอร์

มหาวิทยาลัยสวนดุสิต

WHO IS THIS GUY?

CIKLUM — Data Science Team Lead

2KNOW Data Foresight Services — Sr. Data Scientist

eleks — Software Architect, R&D Engineer

I also teach Machine Learning: LITS | Lviv IT School; Київстар Big Data School

WHAT IS NLP?

Study of interaction between computers and human languages

NLP = Computer Science + AI + Computational Linguistics

COMMON NLP TASKS

Easy

- Chunking
- Part-of-Speech Tagging
- Named Entity Recognition
- Spam Detection
- Thesaurus

Medium

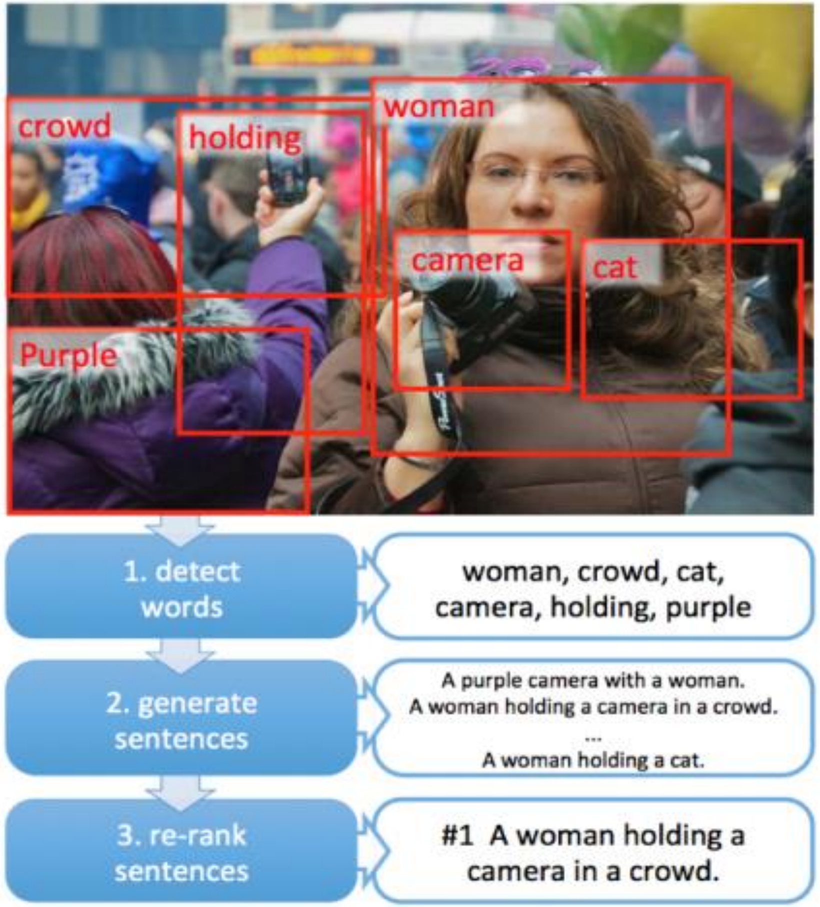
- Syntactic Parsing
- Word Sense Disambiguation
- Sentiment Analysis
- Topic Modeling
- Information Retrieval

Hard

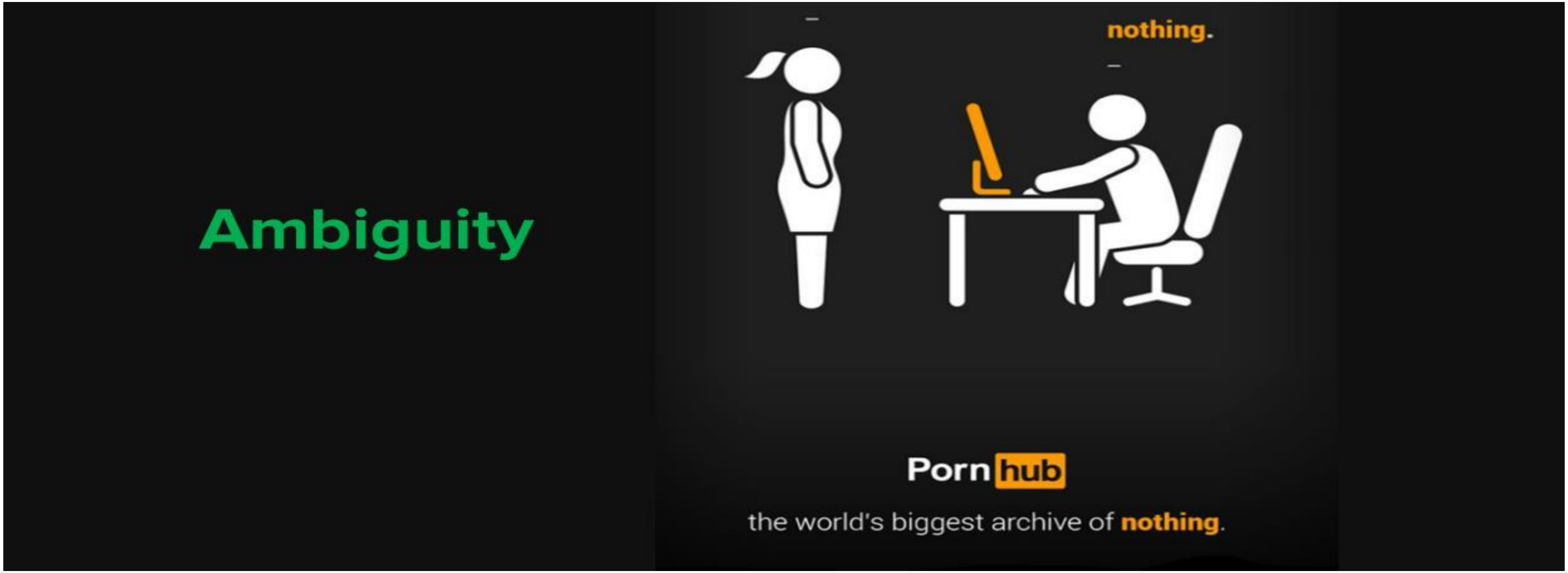
- Machine Translation
- Text Generation
- Automatic Summarization
- Question Answering
- Conversational Interfaces

INTERDISCIPLINARY TASKS: SPEECH-TO-TEXT

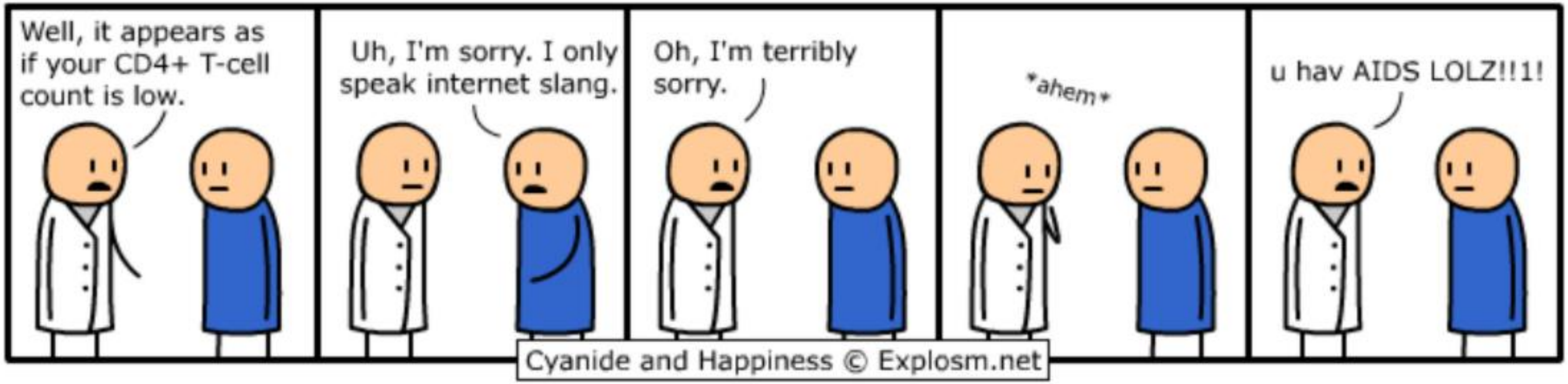
INTERDISCIPLINARY TASKS: IMAGE CAPTIONING



WHAT MAKES NLP SO HARD?



NON-STANDARD LANGUAGE



Also: neologisms, complex entity names, phrasal verbs/idioms

MORE COMPLEX LANGUAGES THAN ENGLISH

- **German:** Donaudampfschiffahrtsgesellschaftskapitän (5 “words”)
- **Chinese:** 50,000 different characters (2-3k to read a newspaper)
- **Japanese:** 3 writing systems
- **Thai:** Ambiguous word boundaries and sentence concepts
- **Slavic:** Different word forms depending on gender, case, tense

WRITE TRADITIONAL "IF-THEN-ELSE" RULES?

BIG NOPE!

Leads to very large and complex codebases.

Still struggles to capture trivial cases (for a human).

BETTER APPROACH: MACHINE LEARNING

“

- A computer program is said to learn from experience **E**
- with respect to some class of tasks **T** and performance measure **P**,
- if its performance at tasks in **T**, as measured by **P**,
- improves with experience **E**.

— Tom M. Mitchell

Before We Begin: Disclaimer

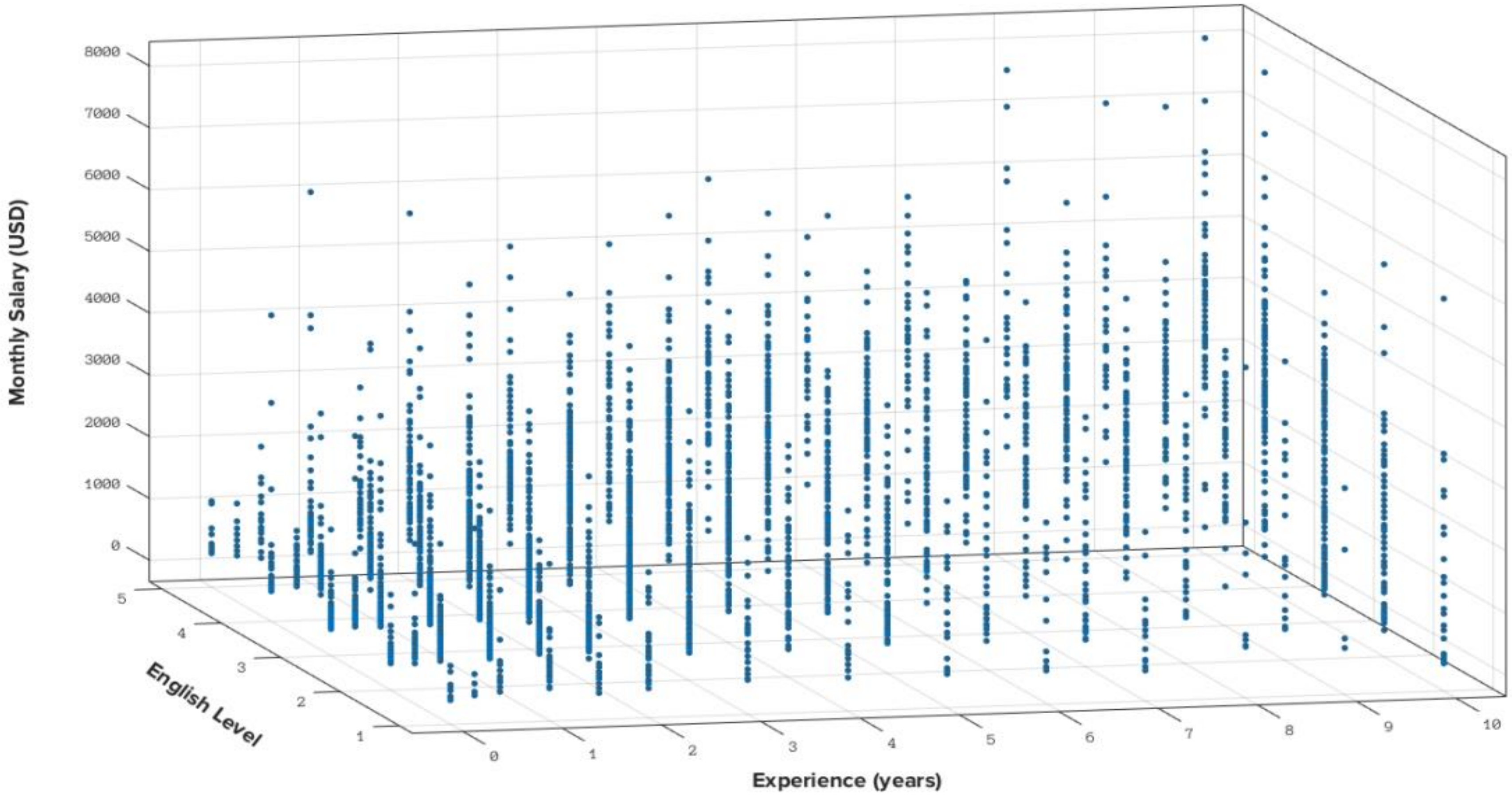
- This will be a very quick description of ML. By no means exhaustive.
- Only the essential background for what we'll have in Part 2.
- To fit everything into a small timeframe, I'll simplify some aspects.
- I encourage you to read ML books or watch videos to dig deeper.

COMMON ML TASKS

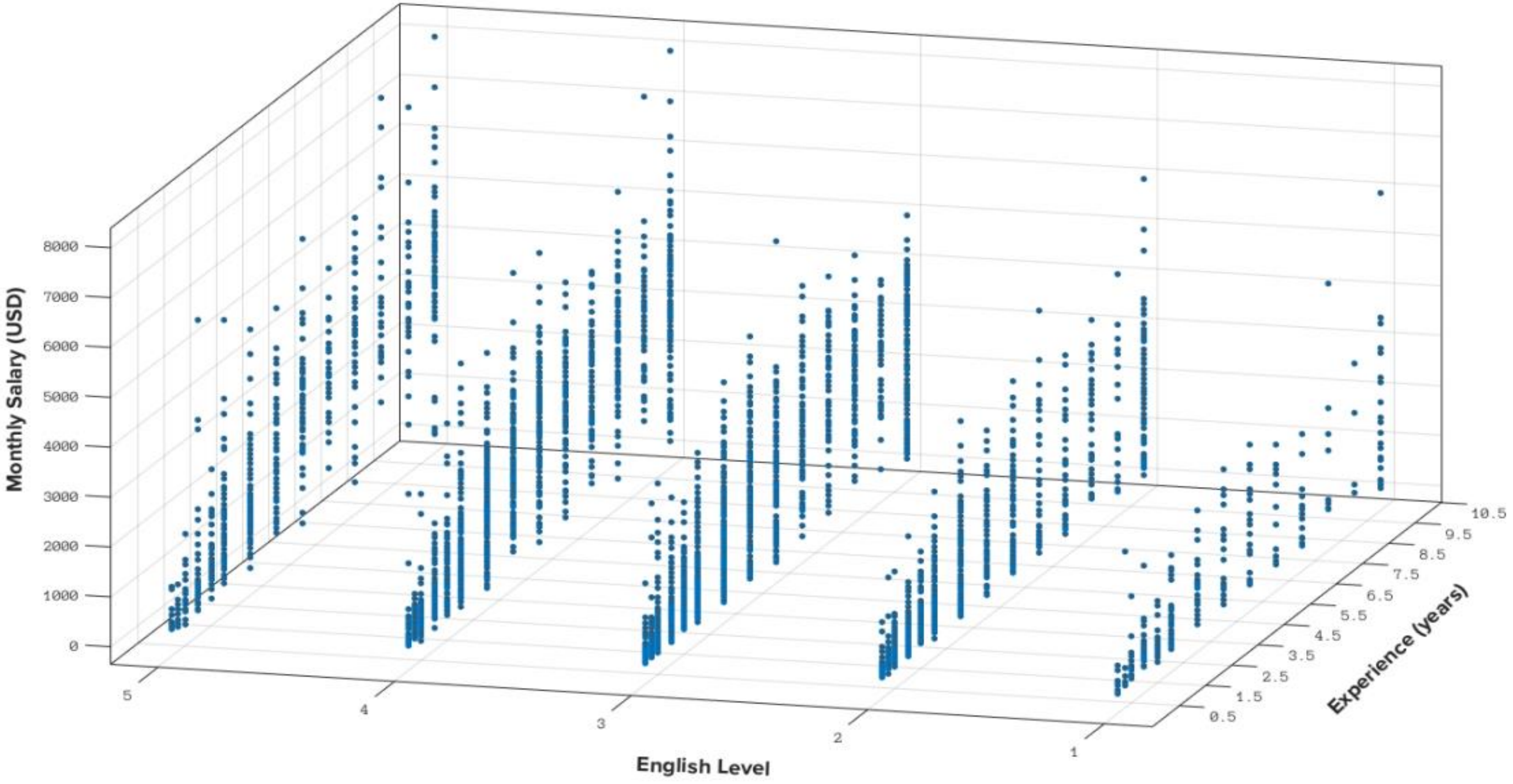
1. **Supervised Learning**
   - Regression
   - Classification (Binary or Multi-Class)

2. **Unsupervised Learning**
   - Clustering
   - Anomaly Detection
   - Latent Variable Models (Dimensionality Reduction, EM, ...)

COMMON ML TASKS



COMMON ML TASKS



REGRESSION

Predict a continuous dependent variable
based on independent predictors

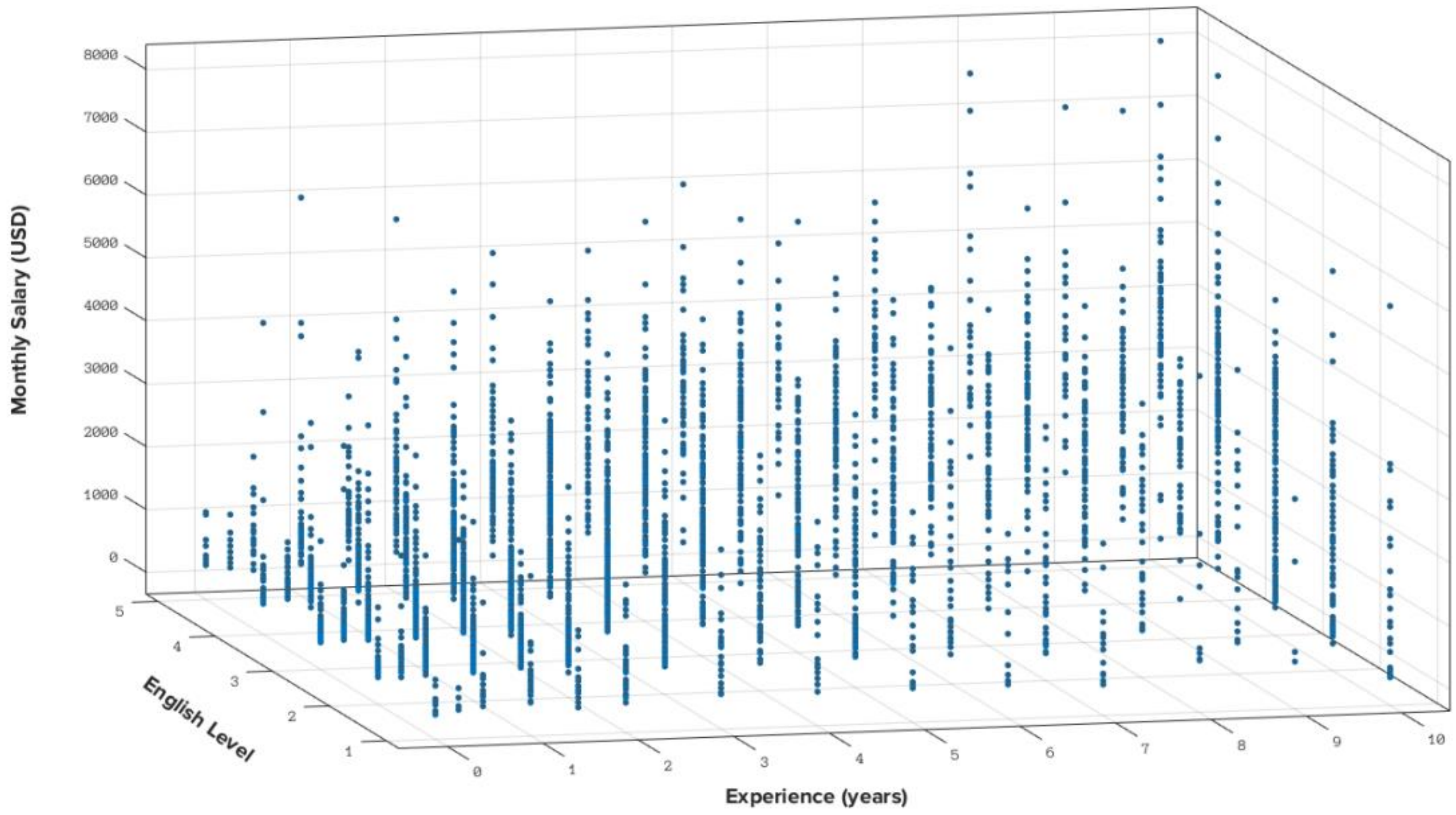
REGRESSION

$$\text{Salary} = \theta_1 \cdot \text{Experience} + \theta_2 \cdot \text{EnglishLevel} + \theta_3$$

$$y = \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \theta_3 = \theta^T x$$

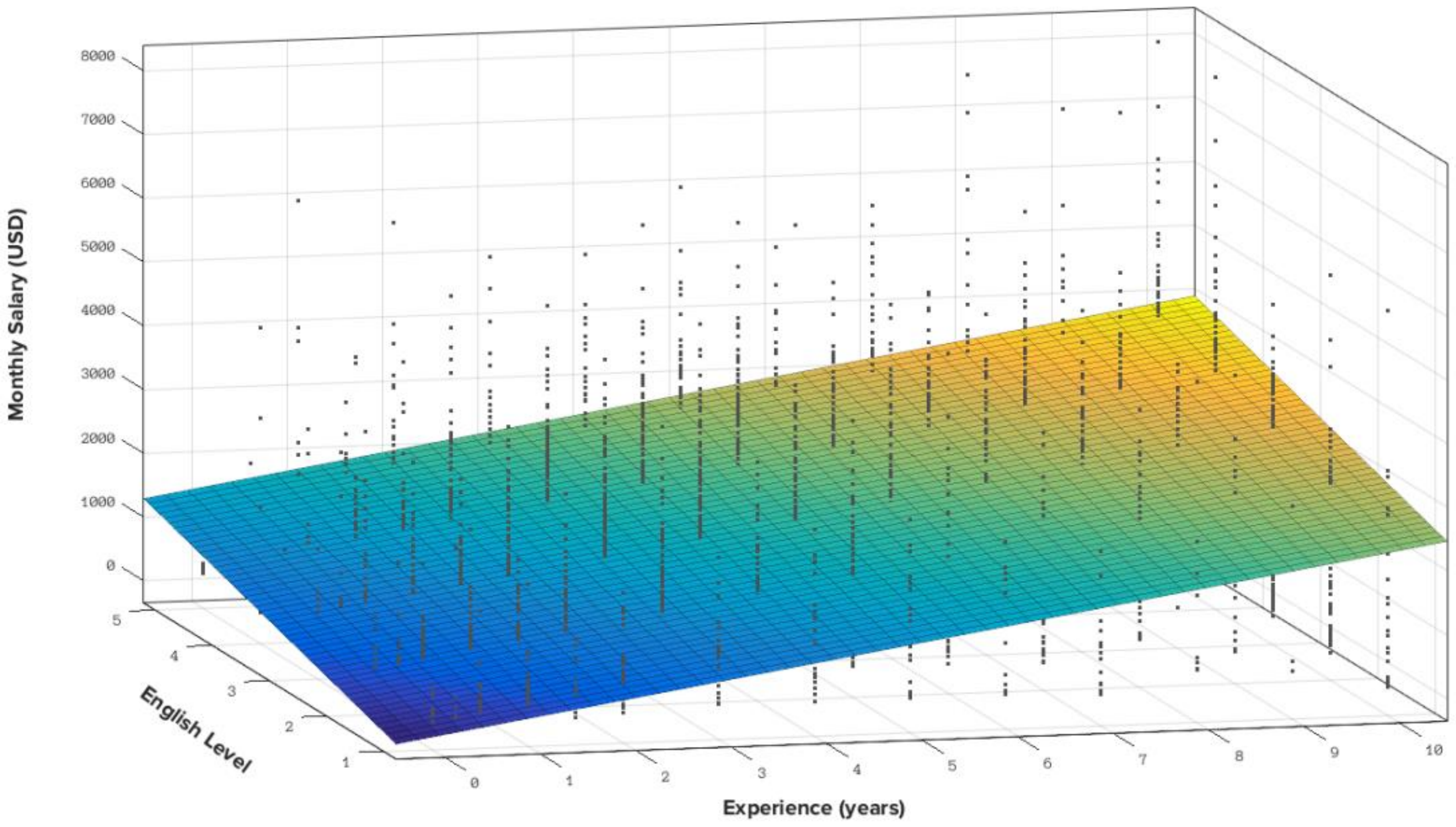
$$h_\theta(x) = \theta^T x$$

REGRESSION



Monthly Salary (USD)
8000
7000
6000
5000
4000
3000
2000
1000
0
English Level
5
4
3
2
1
Experience (years)
0
1
2
3
4
5
6
7
8
9
10

REGRESSION



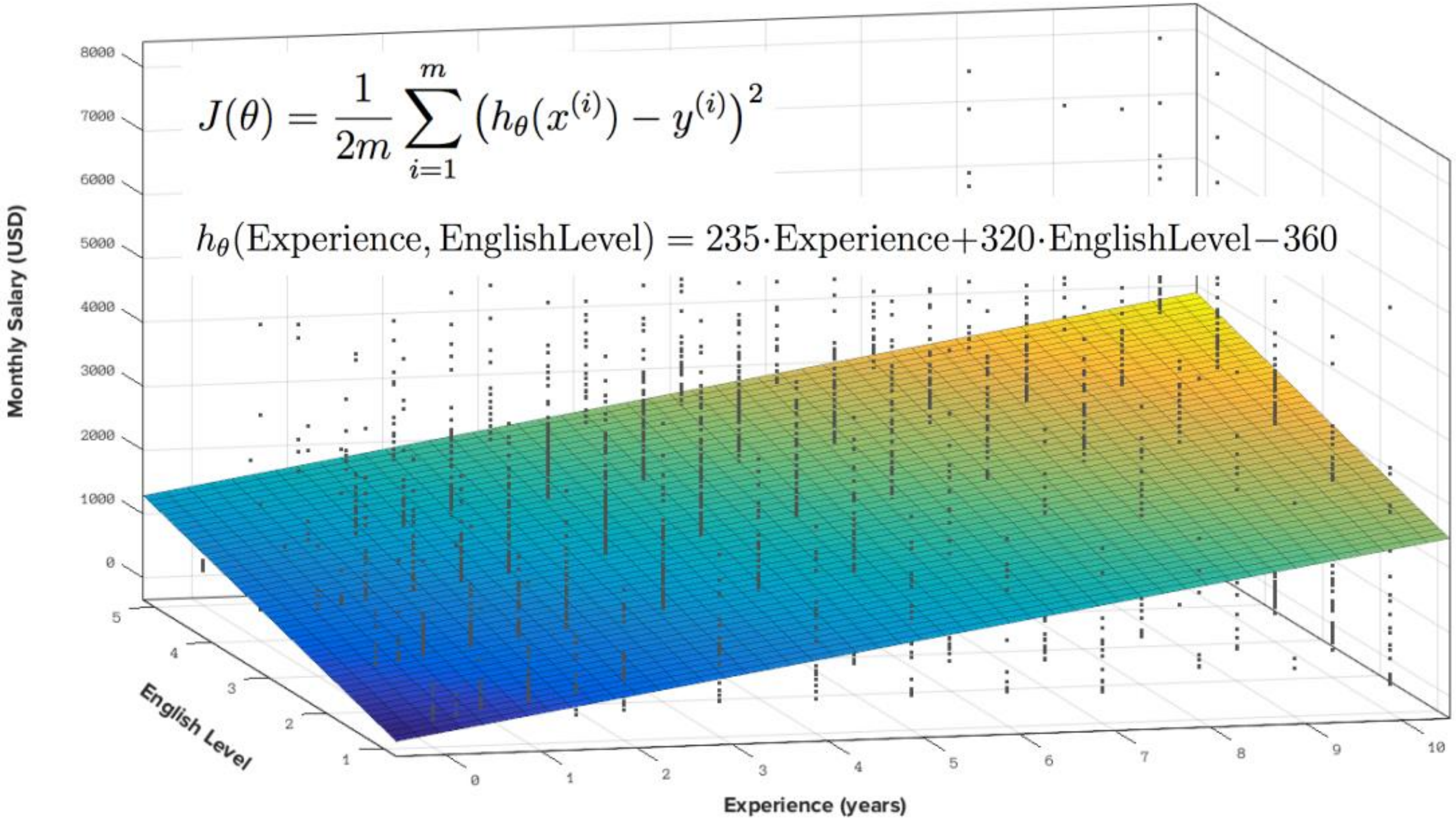
Monthly Salary (USD)
8000
7000
6000
5000
4000
3000
2000
1000
0
English Level
5
4
3
2
1
Experience (years)
0
1
2
3
4
5
6
7
8
9
10

LINEAR REGRESSION

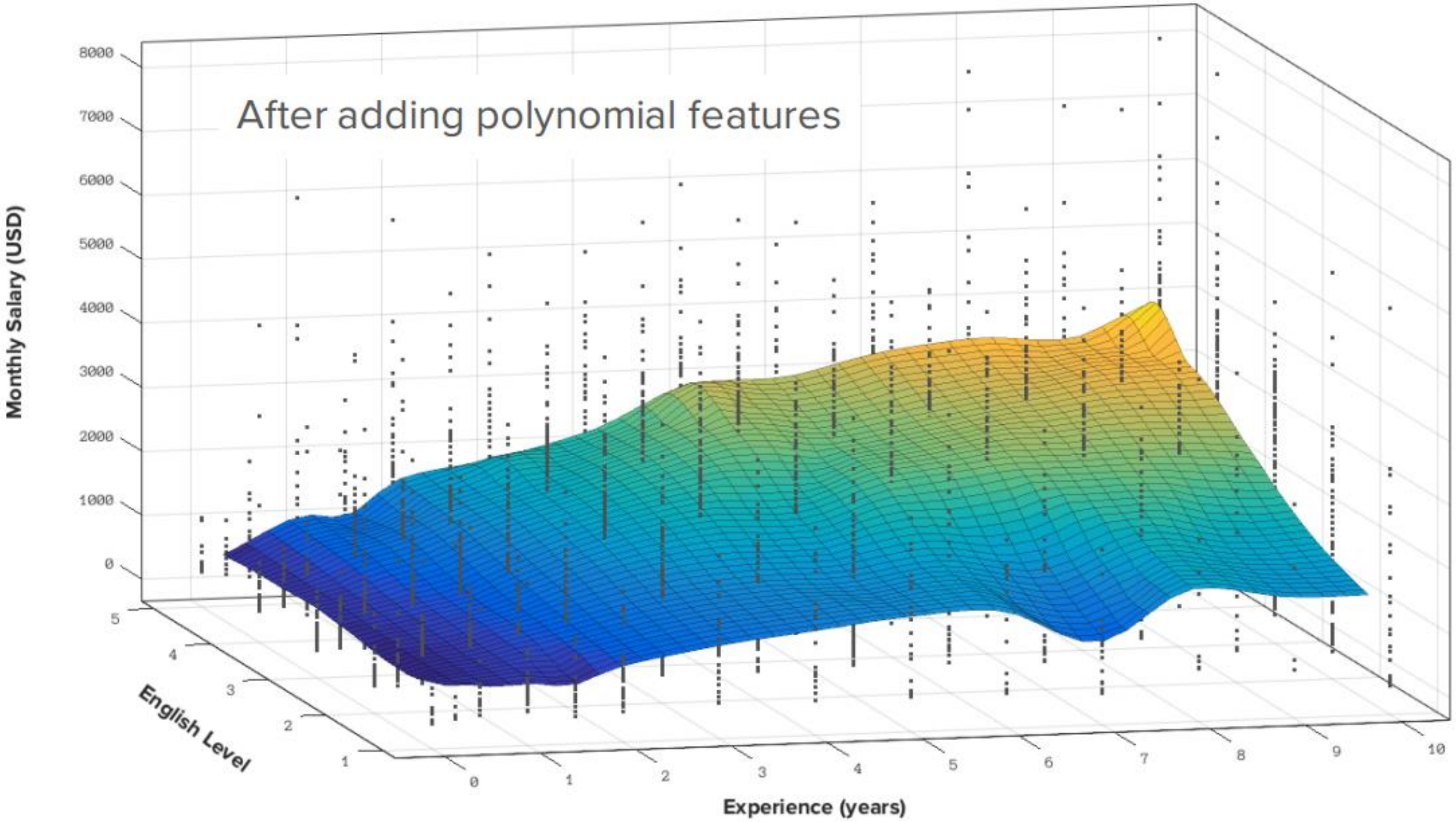
$$h_\theta(x) = \theta^T x$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

LINEAR REGRESSION



LINEAR REGRESSION



LINEAR REGRESSION

$$X = \begin{bmatrix} \text{AliceExperience} & \text{AliceEnglishLevel} & \ldots \\ \text{BobExperience} & \text{BobEnglishLevel} & \ldots \\ \text{CharlieExperience} & \text{CharlieEnglishLevel} & \ldots \\ \ldots & \ldots & \ldots \end{bmatrix} \qquad y = \begin{bmatrix} \text{AliceSalary} \\ \text{BobSalary} \\ \text{CharlieSalary} \\ \ldots \end{bmatrix}$$

Learn model parameters (θ) by optimizing the objective function $J(\theta)$

$$y_{unknown} = h_\theta(X_{unknown}) = X_{unknown}\theta$$

CLASSIFICATION

Assign an observation to some category
from a known discrete list of categories

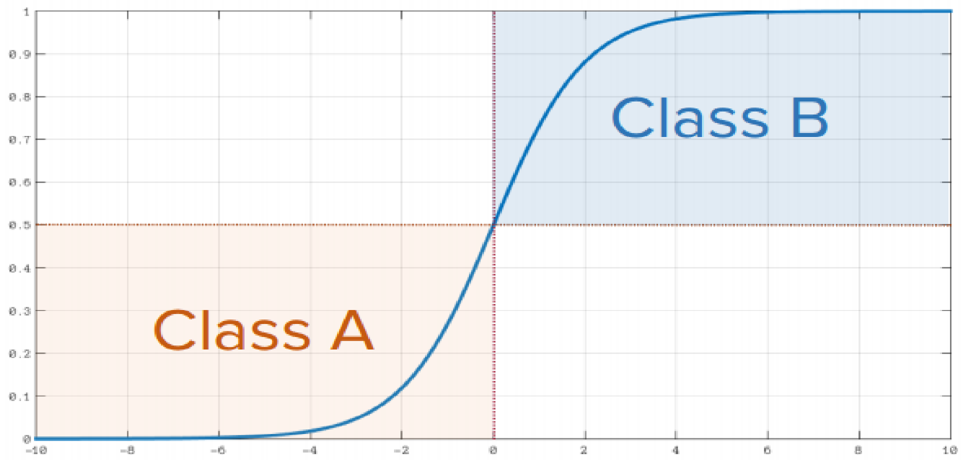
CLASSIFICATION

Logistic Regression

(Multi-class extension = Softmax Regression)

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$$h_\theta(x) = \sigma(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

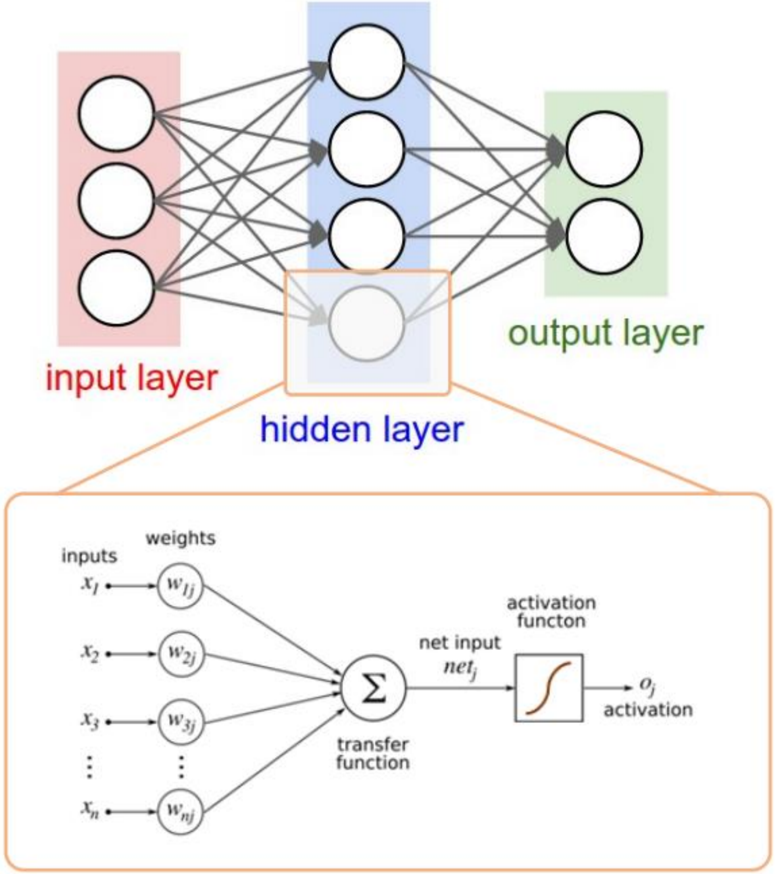


$$J(\theta) = \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right]$$

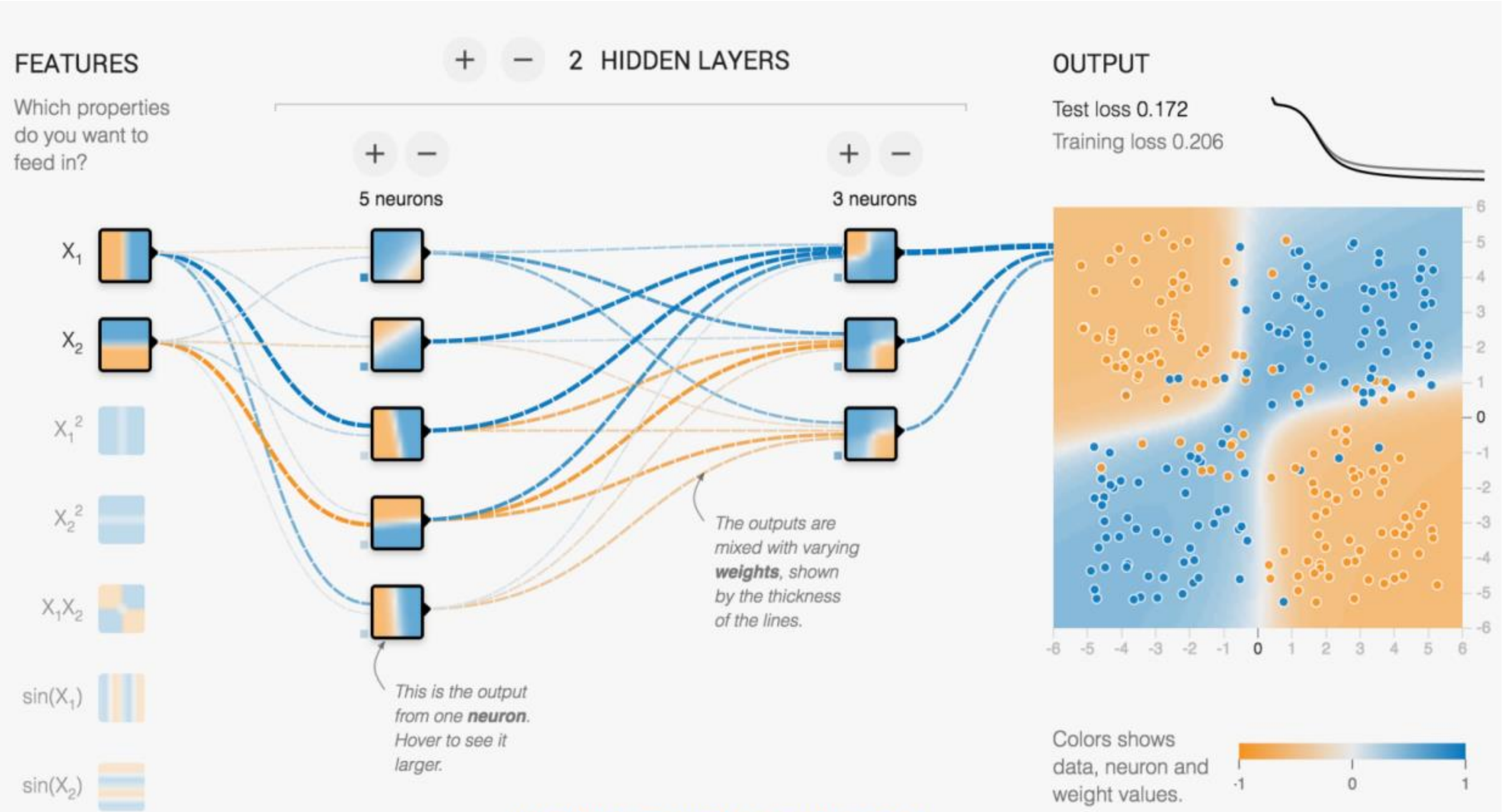
CLASSIFICATION

Neural Networks

and Backpropagation Algorithm



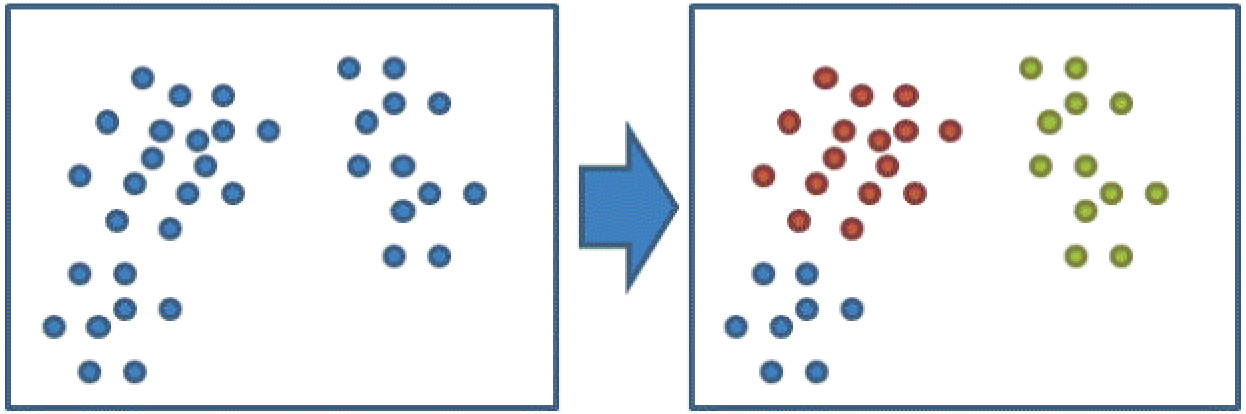
CLASSIFICATION



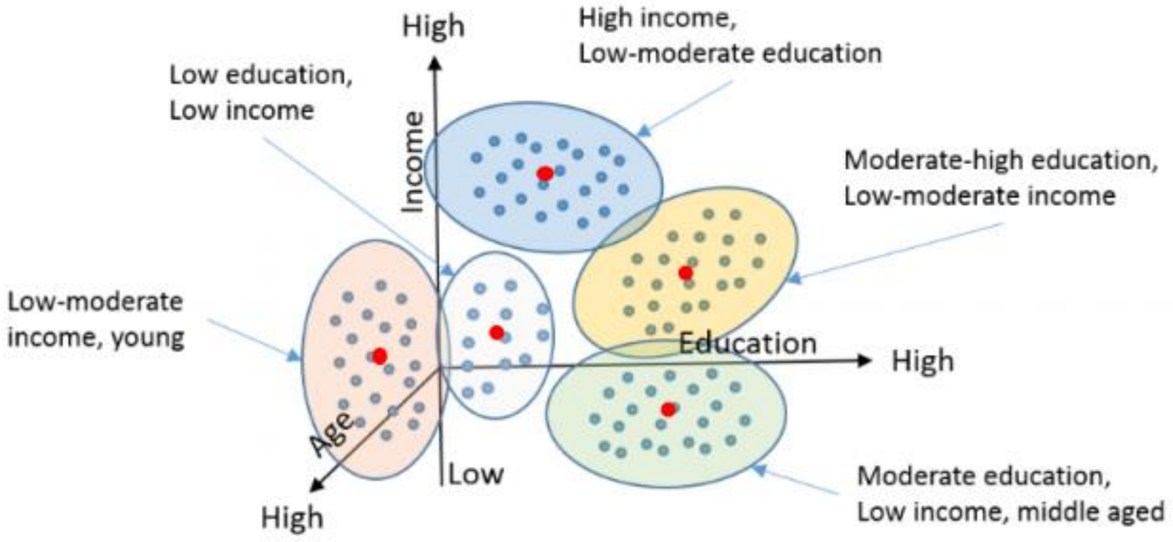
http://playground.tensorflow.org/

CLUSTERING

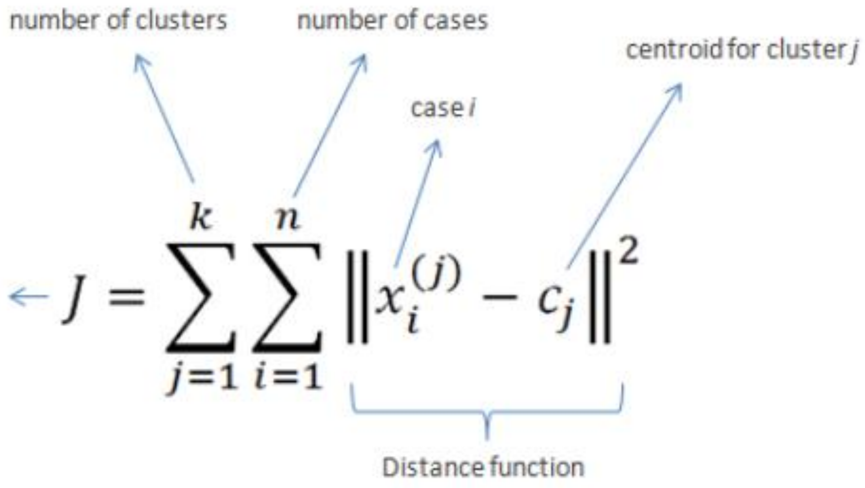
Group objects in such a way
that objects in the **same** group are **similar**,
and objects in the **different** groups are **not**



K-Means Clustering



$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$



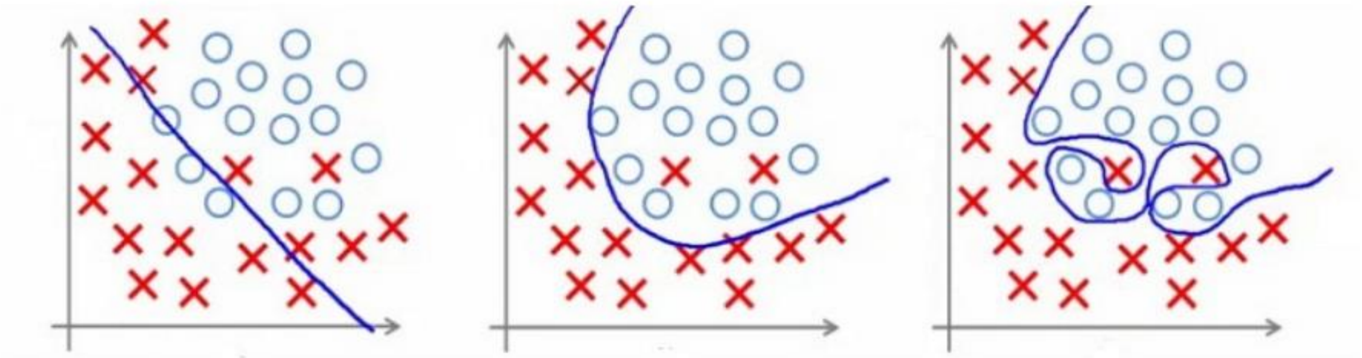
EVALUATION

How do we know if an ML model is good?

What do we do if something goes wrong?

EVALUATION

Underfitting & Overfitting



Under-fitting

(too simple to explain the variance)

Appropriate-fitting

Over-fitting

(forcefitting -- too good to be true)

EVALUATION

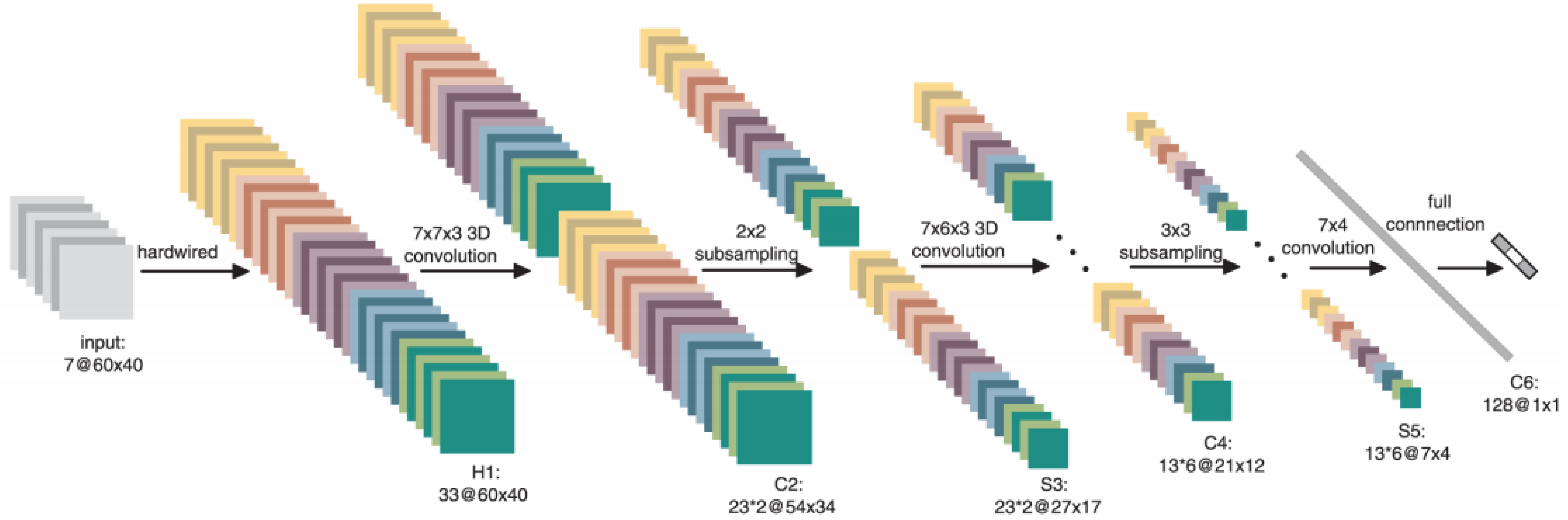
Development & Troubleshooting

- Picking the right metric: MAE, RMSE, AUC, Cross-Entropy, Log-Loss
- Training Set / Validation Set / Test Set split
- Picking hyperparameters against Validation Set
- Regularization to prevent OF
- Plotting learning curves to check for UF/OF

DEEP LEARNING

- Core idea: instead of hand-crafting complex features, use increased computing capacity and build a deep computation graph that will try to **learn feature representations on its own**.
  End-to-end learning rather than a cascade of apps.
- Works best with lots of **homogeneous**, **spatially related** features (image pixels, character sequences, audio signal measurements).
  Usually works **poorly otherwise**.
- State-of-the-art and/or superhuman performance on many tasks.
- Typically requires **massive** amounts of data and training resources.
- But: a very young field. Theories not strongly established, views change.

EXAMPLE: CONVOLUTIONAL NEURAL NETWORK



"Classical" NLP Pipeline

Tokenization ↓	Break text into sentences and words, lemmatize
Morphology ↓	Part of speech (POS) tagging, stemming, NER
Syntax ↓	Constituency/dependency parsing
Semantics ↓	Coreference resolution, wordsense disambiguation
Discourse	Task-dependent (sentiment, …)

Often Relies on Language Banks

- WordNet (ontology, semantic similarity tree)
- Penn Treebank (POS, grammar rules)
- PropBank (semantic propositions)
- ...Dozens of them!

PART 2 NLP CHALLENGES AND APPROACHES

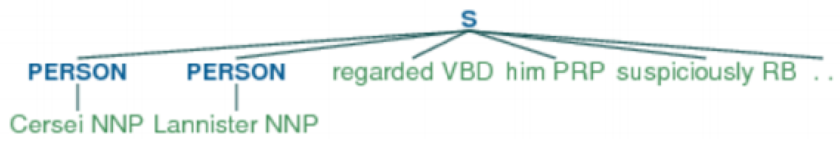
Tokenization & Stemming

```
In [1]: import nltk

In [2]: nltk.download(["punkt", "averaged_perceptron_tagger", "treebank",
                       "maxent_ne_chunker", "words", "wordnet"])
        ...

In [3]: text = open("data/clean/asoiaf01.txt").read()

In [4]: sentences = nltk.tokenize.sent_tokenize(text)

In [5]: sentences[5313]

Out[5]: 'Cersei Lannister regarded him suspiciously.'

In [6]: sentences = [nltk.tokenize.word_tokenize(sentence) for sentence in sentences]

In [7]: sentences[5313]

Out[7]: ['Cersei', 'Lannister', 'regarded', 'him', 'suspiciously', '.']

In [8]: stemmer = nltk.stem.SnowballStemmer(language="english")
        [stemmer.stem(word) for word in sentences[5313]]

Out[8]: ['cersei', 'lannist', 'regard', 'him', 'suspici', '.']
```

PART 2 NLP CHALLENGES AND APPROACHES

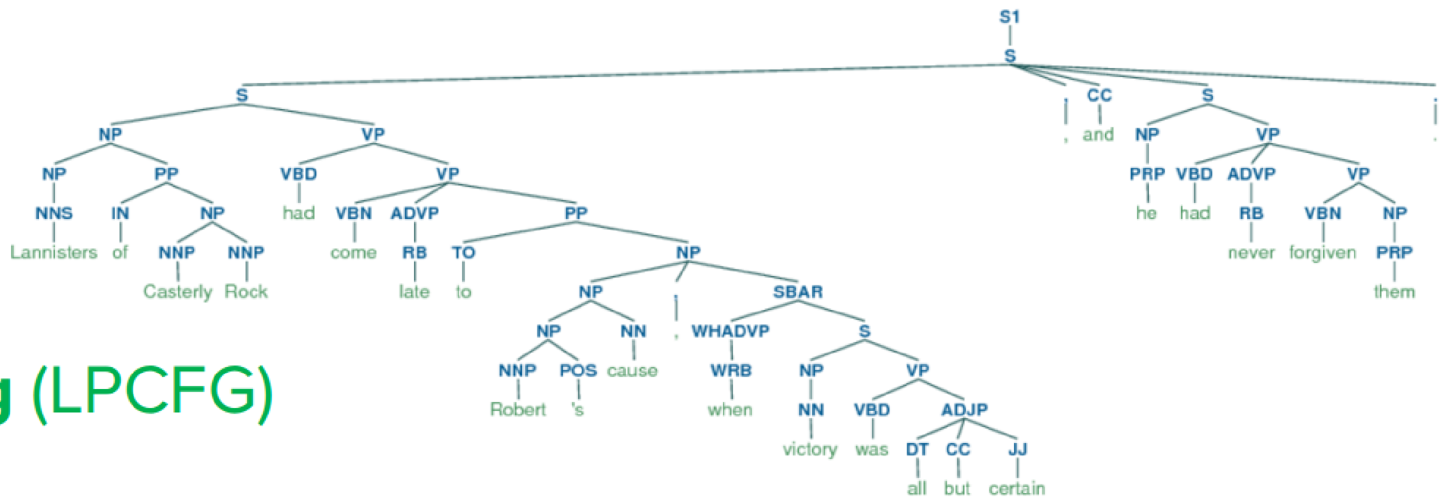
POS/NER Tagging

```
In [10]: tagged_sentence = nltk.pos_tag(sentences[5313])

In [11]: tagged_sentence

Out[11]: [('Cersei', 'NNP'),
          ('Lannister', 'NNP'),
          ('regarded', 'VBD'),
          ('him', 'PRP'),
          ('suspiciously', 'RB'),
          ('.', '.')]

In [12]: lemmatizer = nltk.stem.wordnet.WordNetLemmatizer()
         [lemmatizer.lemmatize(word, pos=penn_to_wn(tag)) for word, tag in tagged_sentence]

Out[12]: ['Cersei', 'Lannister', 'regard', 'him', 'suspiciously', '.']

In [13]: tree = nltk.ne_chunk(tagged_sentence)

In [14]: tree.draw()
```



PART 2 NLP CHALLENGES AND APPROACHES

```
In [1]: import bllipparser

In [2]: model_dir = "/home/yuriyguts/nltk_data/models/bllip_wsj_no_aux"
        parser = bllipparser.RerankingParser.from_unified_model_dir(model_dir)

In [3]: sentence = "Lannisters of Casterly Rock had come late to Robert's cause, "\
                   "when victory was all but certain, and he had never forgiven them."

In [4]: best_parse_tree = parser.parse(sentence)[0].ptb_parse

In [5]: best_parse_tree.as_nltk_tree().draw()
```



Parsing (LPCFG)

“CLASSICAL” WAY: TRAINING A NER TAGGER

Task: Predict whether the word is a **PERSON**, **LOCATION**, **DATE** or **OTHER**.

Could be more than 3 NER tags (e.g. MUC-7 contains 7 tags).

Features:

1. Current word.
2. Previous, next word (context).
3. POS tags of current word and nearby words.
4. NER label for previous word.
5. Word substrings (e.g. ends in “burg”, contains “oxa” etc.)
6. Word shape (internal capitalization, numerals, dashes etc.).
7. …on and on and on…

FEATURE REPRESENTATION: BAG OF WORDS

A single word is a one-hot encoding vector with the size of the dictionary :(

FEATURE REPRESENTATION: BAG OF WORDS

Problem

- Manually designed features are often over-specified, incomplete, take a long time to design and validate.
- Often requires PhD-level knowledge of the domain.
- Researchers spend literally decades hand-crafting features.
- Bag of words model is very high-dimensional and sparse, cannot capture semantics or morphology.

Maybe **Deep Learning** can help?

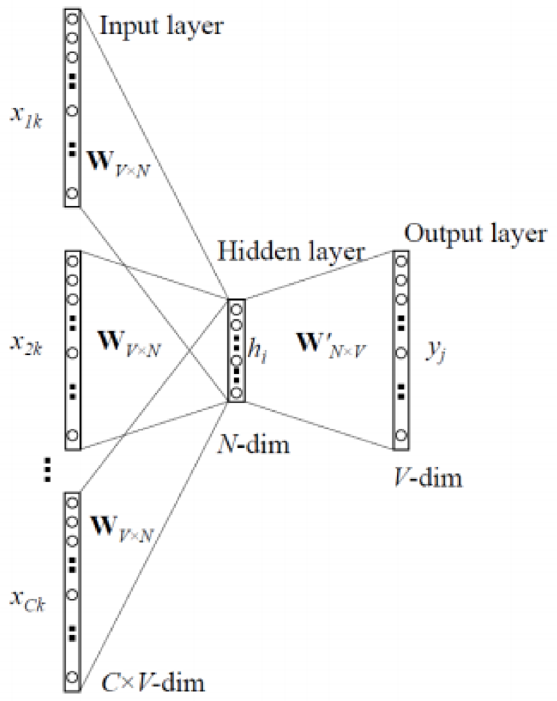
DEEP LEARNING FOR NLP

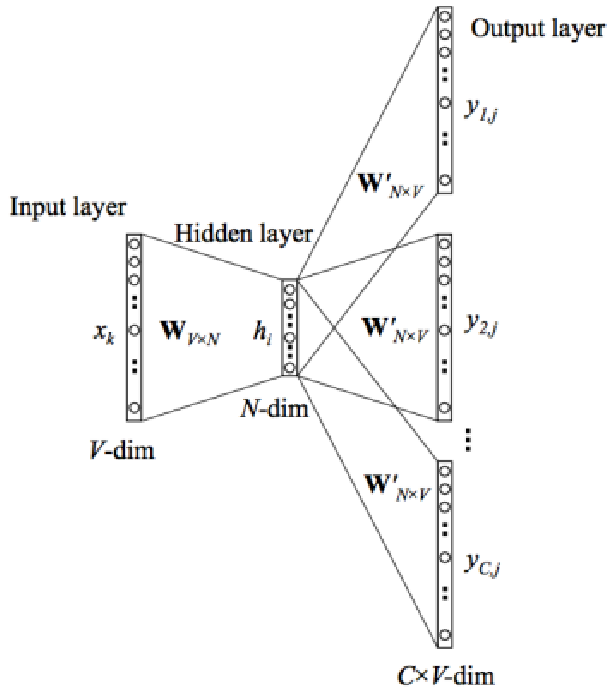
- Core enabling idea: represent words as **dense vectors**

  ~~[0 1 0 0 0 0 0 0 0]~~ [0.315 0.136 0.831]

- Try to capture semantic and morphologic similarity so that the features for “similar” words are “similar”
  (e.g. closer in Euclidean space).
- Natural language is context dependent: use context for learning.
- Straightforward (but slow) way: build a co-occurrence matrix and SVD it.

EMBEDDING METHODS: WORD2VEC





CBoW version: predict center word from context

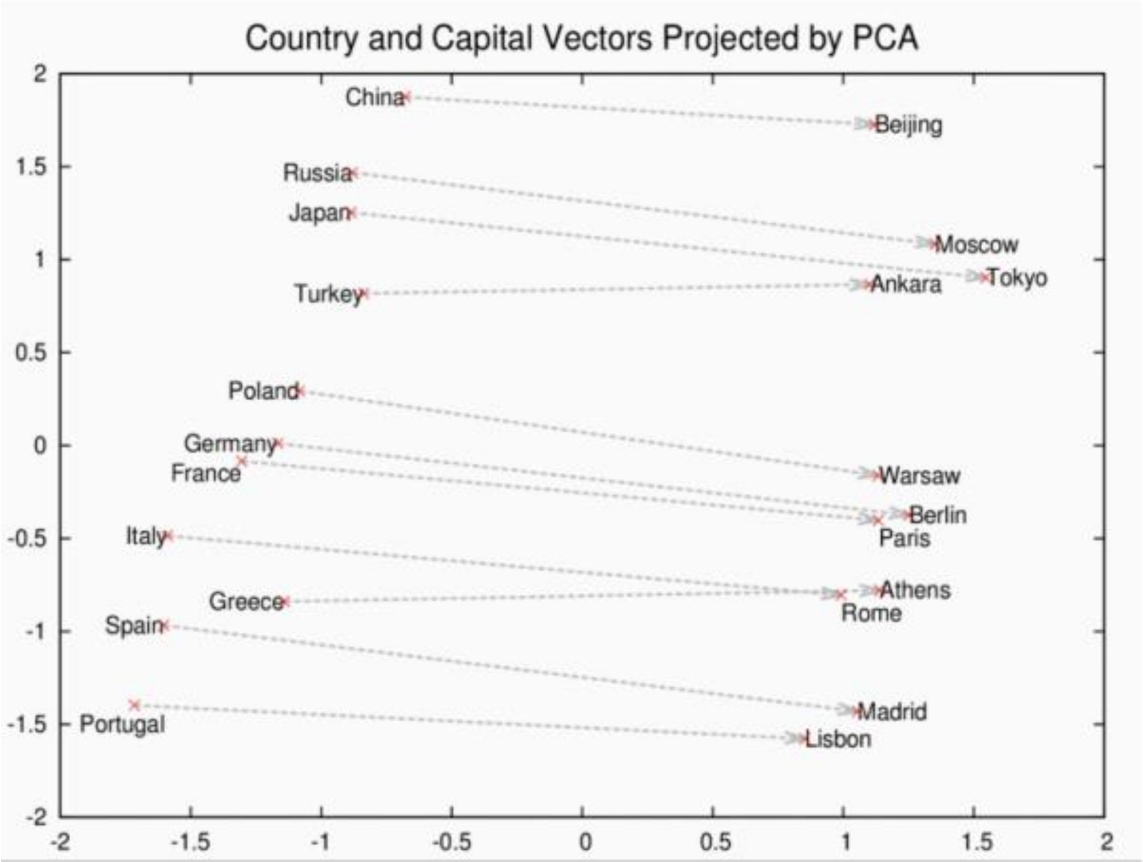
Skip-gram version: predict context from center word

EMBEDDING METHODS: WORD2VEC

Benefits

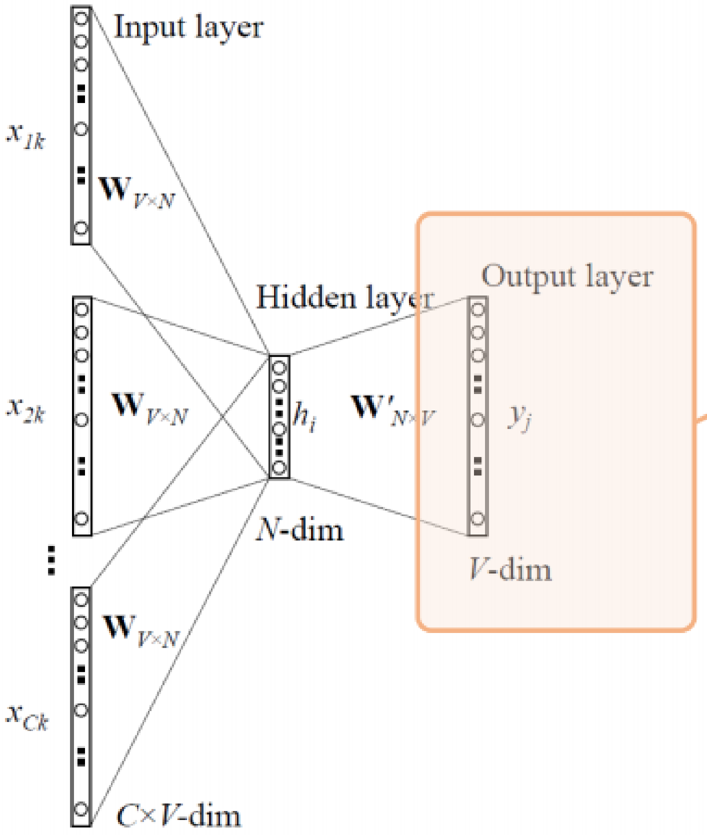
- Learns features of each word on its own, given a text corpus.
- No heavy preprocessing is required, just a corpus.
- Word vectors can be used as features for lots of supervised learning applications: POS, NER, chunking, semantic role labeling. All with pretty much the same network architecture.
- **Similarities and linear relationships between word vectors**.
- A bit more modern representation: GloVe, but requires more RAM.

LINEARITIES



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

TRAINING A NER TAGGER: DEEP LEARNING



Just replace this with NER tag
(or POS tag, chunk end, etc.)

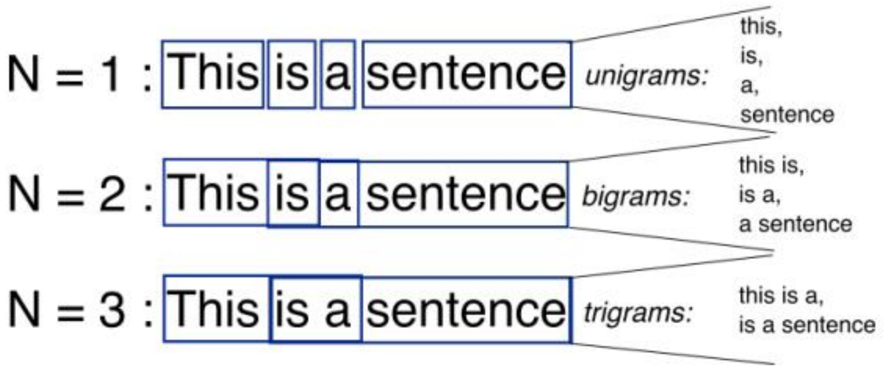
LANGUAGE MODELING

Assign high probabilities to well-formed sentences

(crucial for text generation, speech recognition, machine translation)

LANGUAGE MODELING

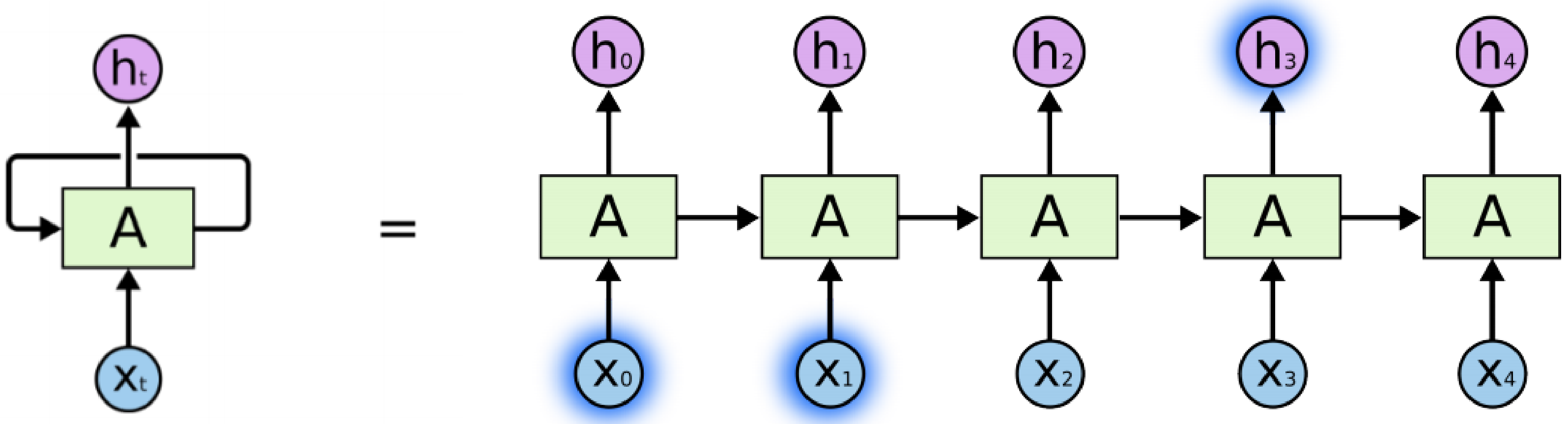
"Classical" Way: N-Grams



$$P(w_i|w_{i-k}^{i-1}) = \frac{count(w_{i-k}^{i}) + \alpha}{count(w_{i-k}^{i-1}) + \alpha V}$$

Problem: doesn't scale well to bigger N. N = 5 is pretty much the limit.

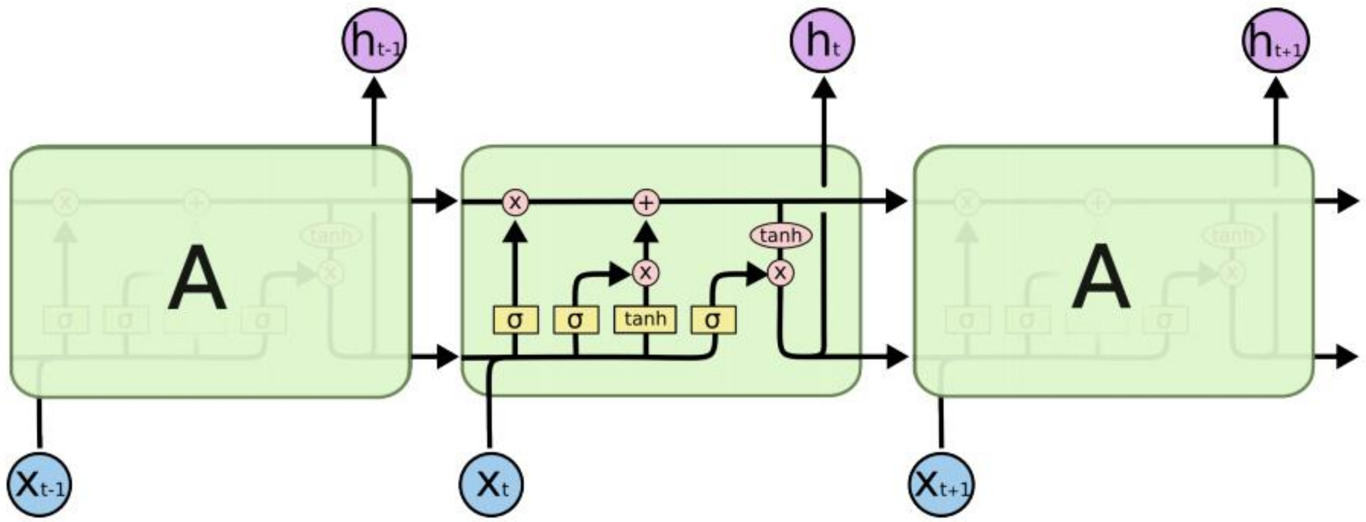
Deep Learning Way: Recurrent NN (RNN)



Can use past information without restricting the size of the context.
But: in practice, can't recall information that came in a long time ago.

LANGUAGE MODELING

Long Short Term Memory Network (LSTM)

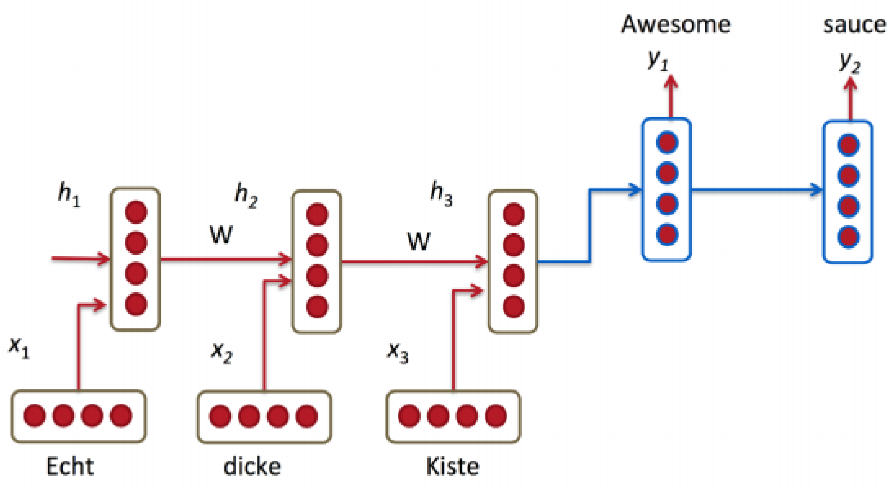


Contains gates that control forgetting, adding, updating and outputting information.
Surprisingly amazing performance at language tasks compared to vanilla RNN.

Tackling Hard Tasks

Deep Learning enables **end-to-end learning** for Machine Translation, Image Captioning, Text Generation, Summarization:

NLP tasks which are inherently very hard!



RNN for Machine Translation

HOTTEST CURRENT RESEARCH

- Attention Networks
- Dynamic Memory Networks

(see ICML 2016 proceedings)

HOTTEST CURRENT RESEARCH

Tools I Used

- NLTK (Python)
- Gensim (Python)
- Stanford CoreNLP (Java with bindings)
- Apache OpenNLP (Java with bindings)

Deep Learning Frameworks with GPU Support:

- Torch (Torch-RNN) (Lua)
- TensorFlow, Theano, Keras (Python)

HOTTEST CURRENT RESEARCH

NLP Progress for Ukrainian

- Ukrainian lemma dictionary with POS tags
  https://github.com/arysin/dict_uk
- Ukrainian lemmatizer plugin for ElasticSearch
  https://github.com/mrgambal/elasticsearch-ukrainian-lemmatizer
- **lang-uk project** (1M corpus, NER, tokenization, etc.)
  https://github.com/lang-uk

HOTTEST CURRENT RESEARCH

Demo 1: Exploring Semantic Properties Of ASOIAF ("Game of Thrones")

Demo 2: Topic Modeling for DOU.UA Comments

GitHub Repos with IPython Notebooks

- https://github.com/YuriyGuts/thrones2vec
- https://github.com/YuriyGuts/dou-topic-modeling

THANK YOU
FOR
YOUR ATTENTION

จบบทที่ 6