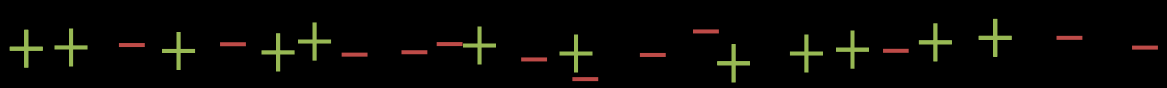
MACHINE LEARNING OVERVIEW

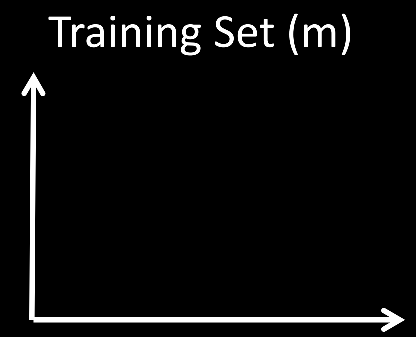
What if your model has a high error?

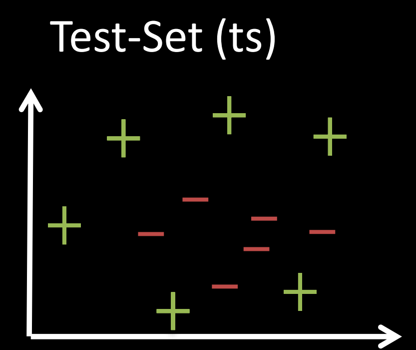
- Try getting more training examples
- Try smaller sets of features
- Try getting additional features
- Try creating features from existing features (kernels)
- Try decrease regularization
- Try increase regularization

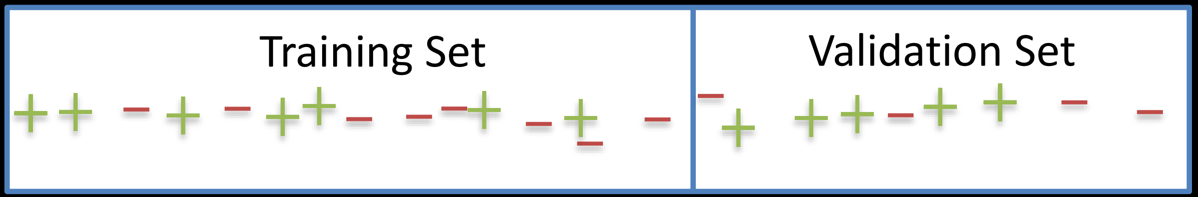
Bias and Variance



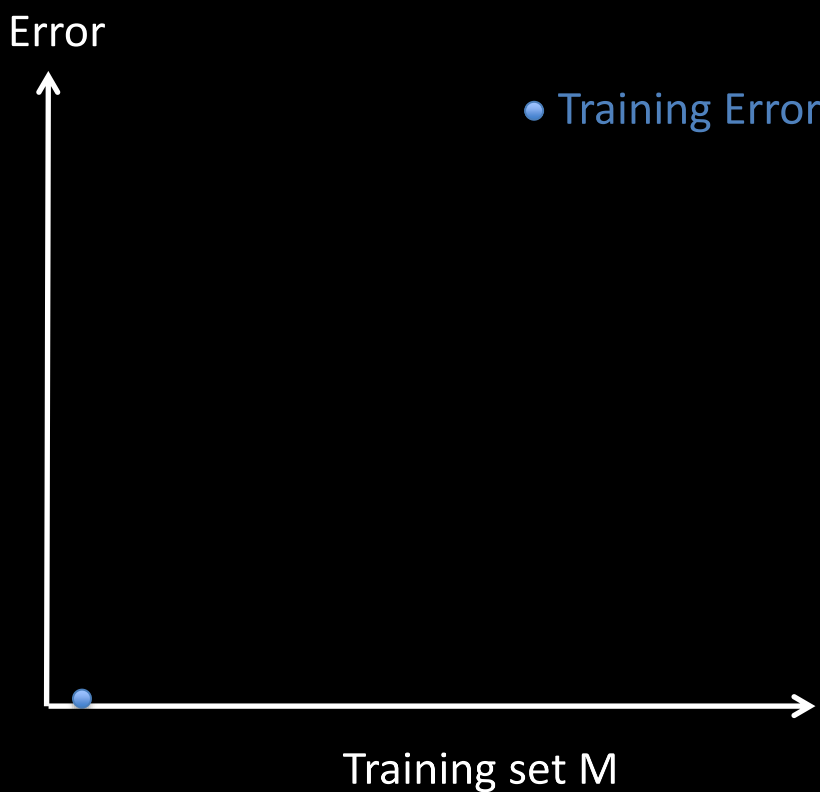
Bias and Variance

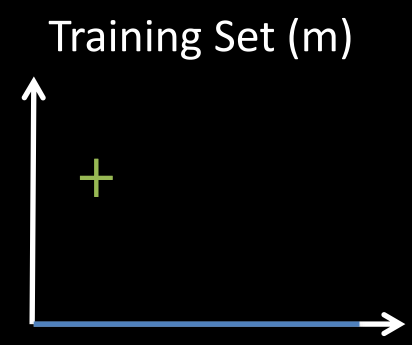




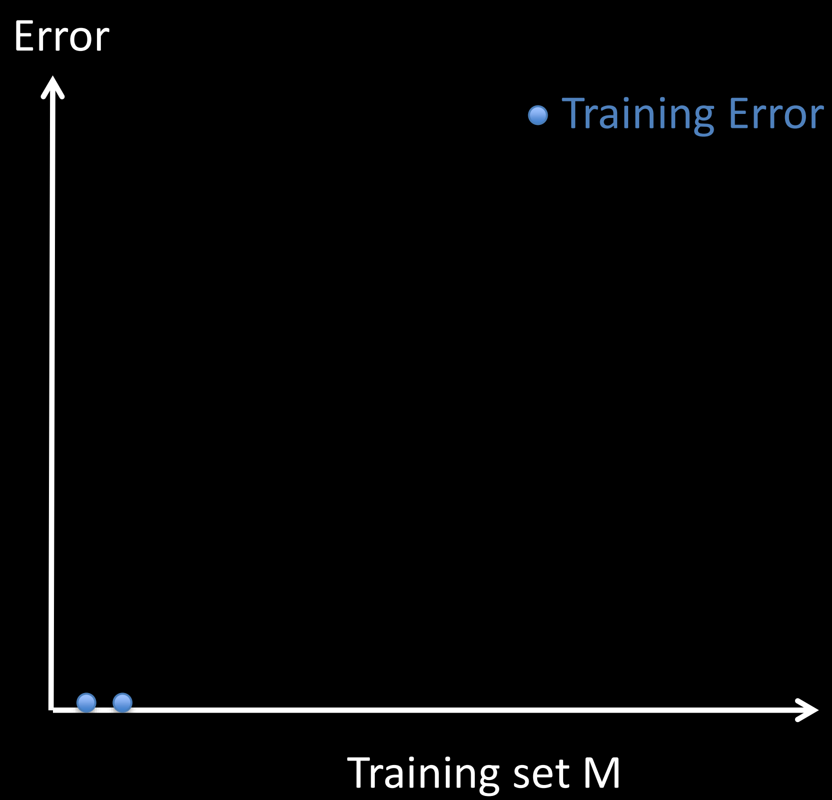


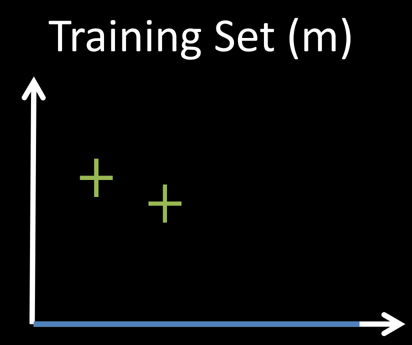
Bias and Variance

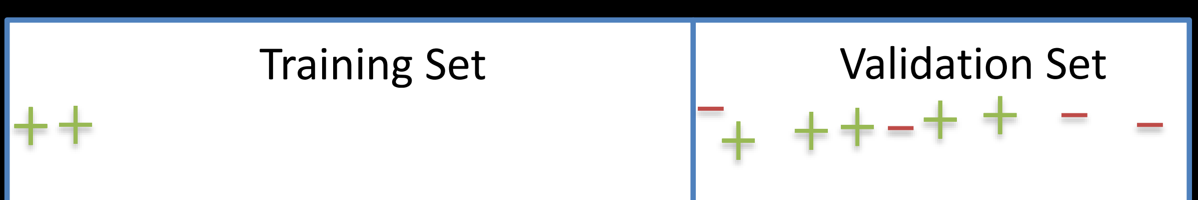




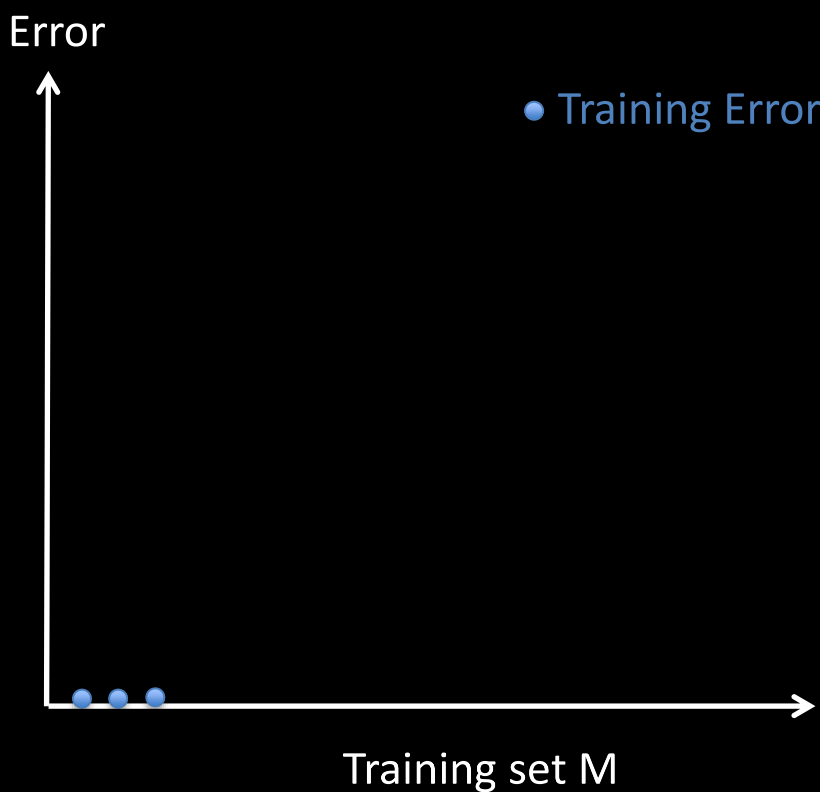
Bias and Variance

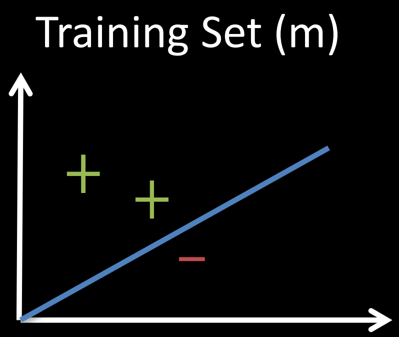




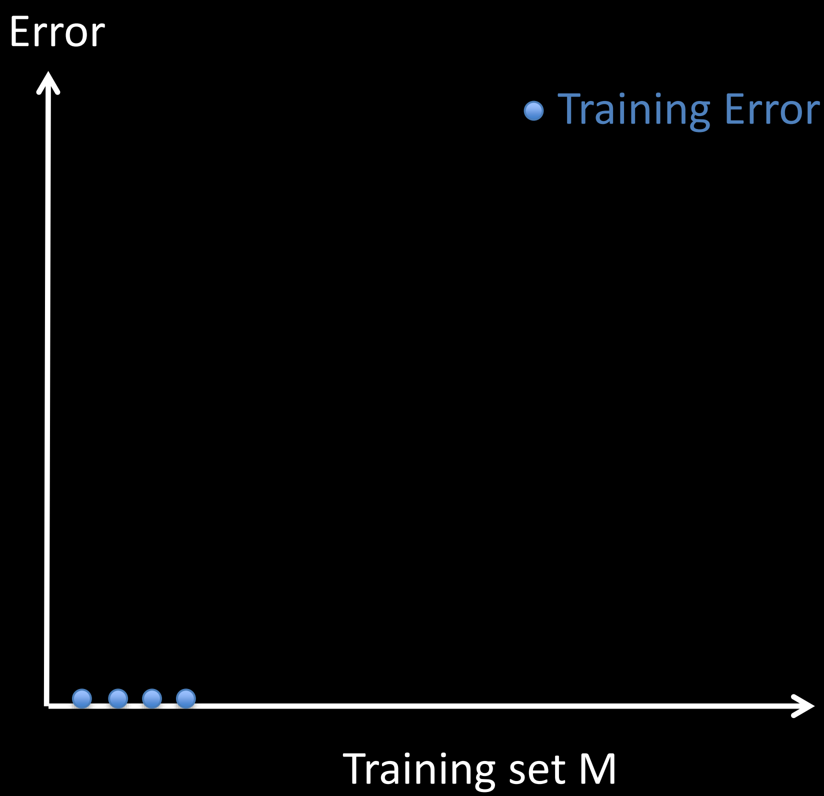


Bias and Variance

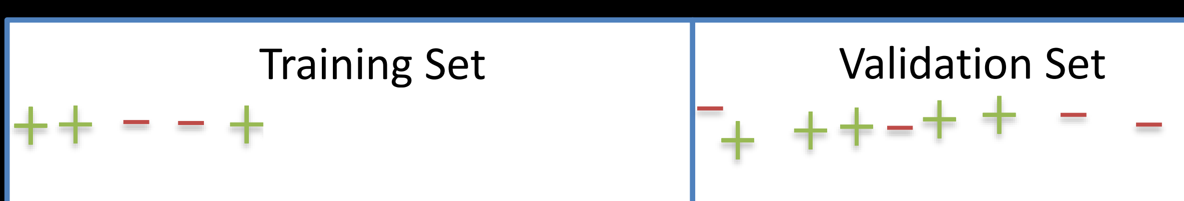




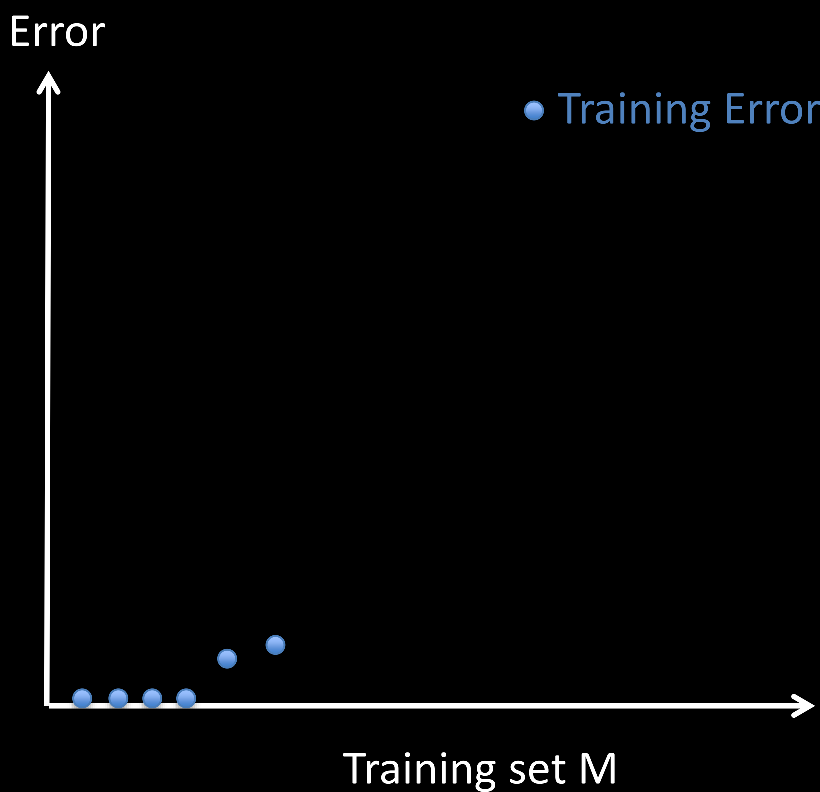
Bias and Variance

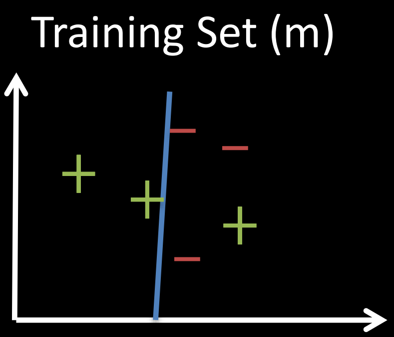


Bias and Variance

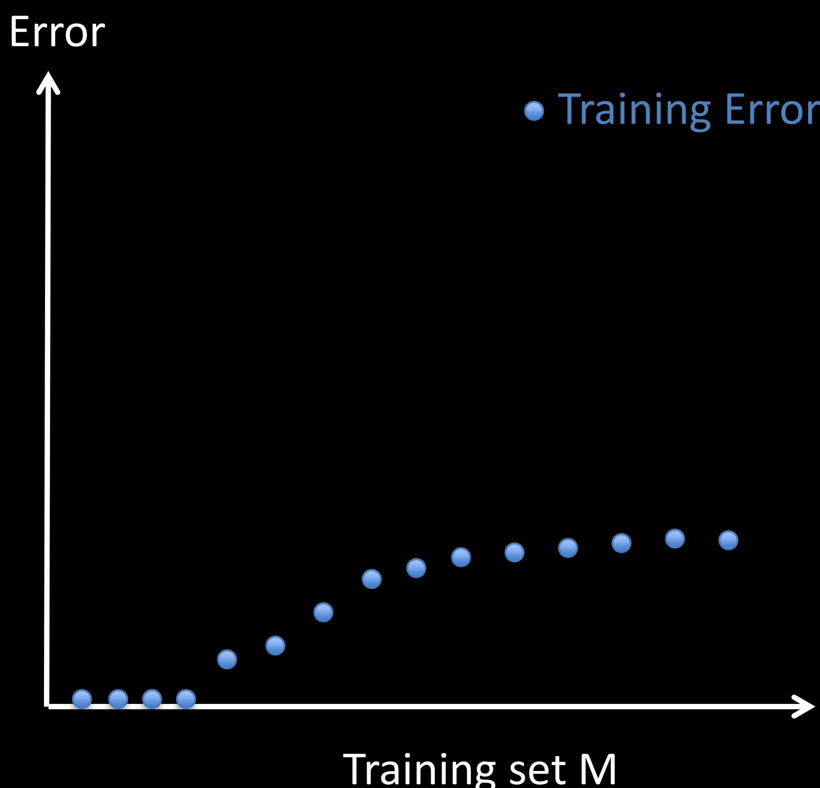


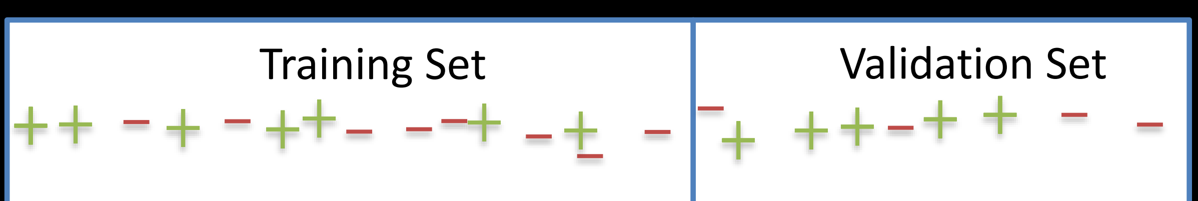
Bias and Variance



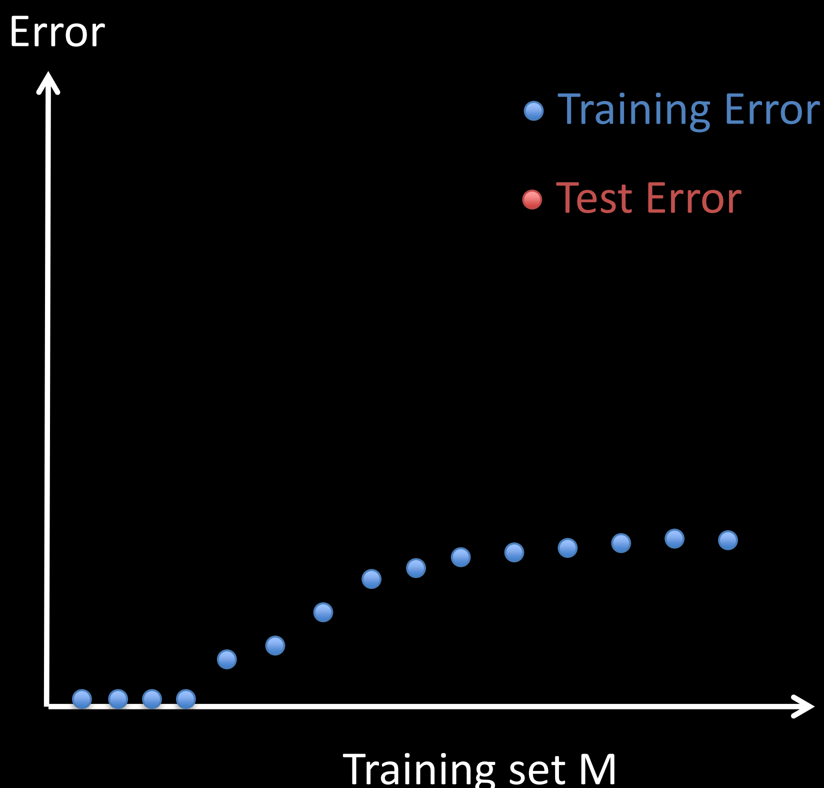


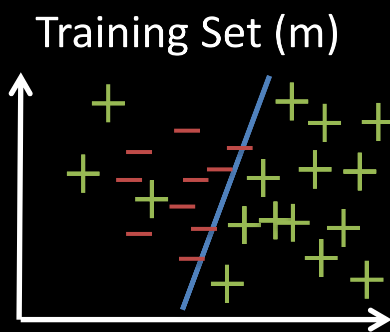
Bias and Variance





Bias and Variance

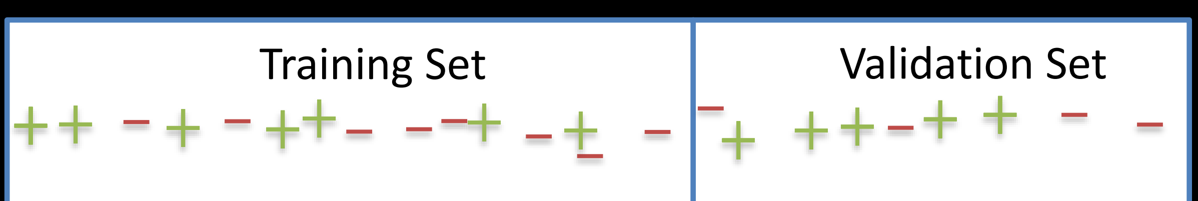




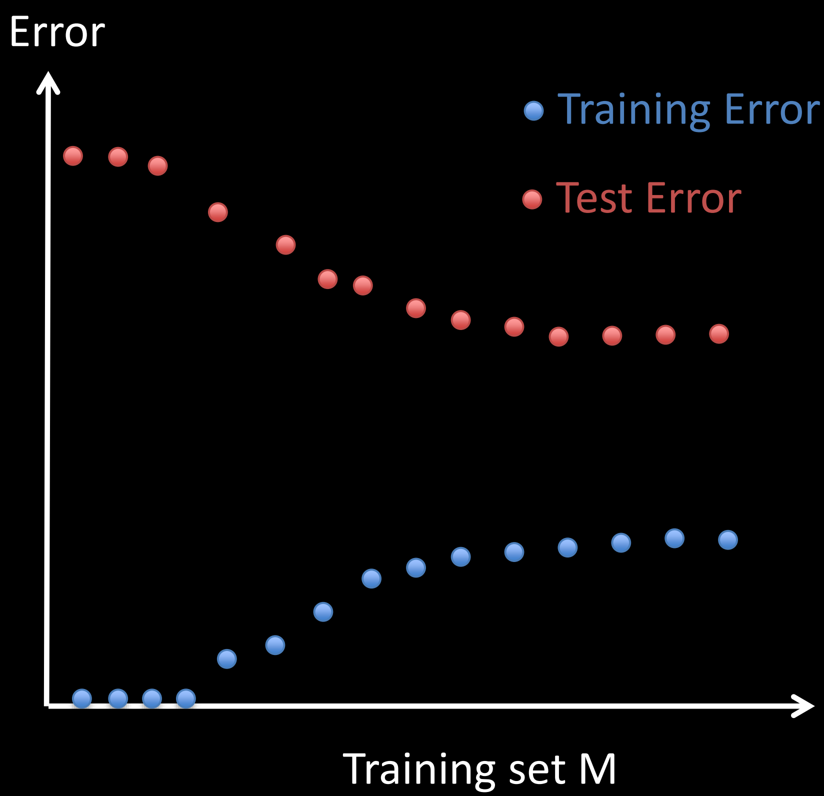
Clicker:

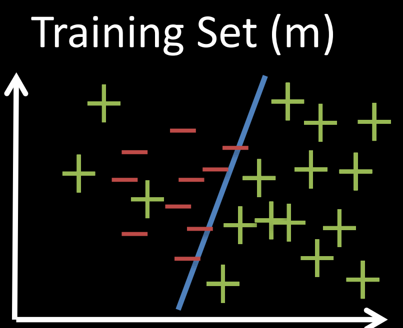
Test error

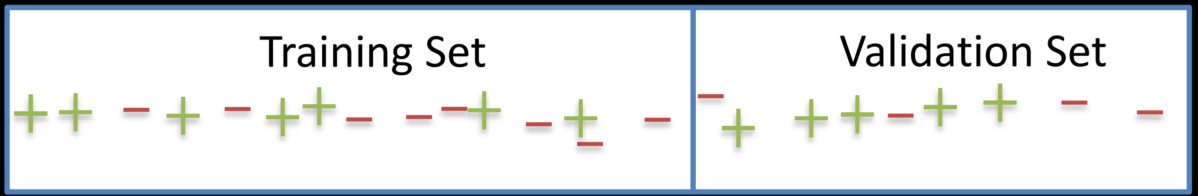
a) decreases with M
b) increases with M
c) stays constant



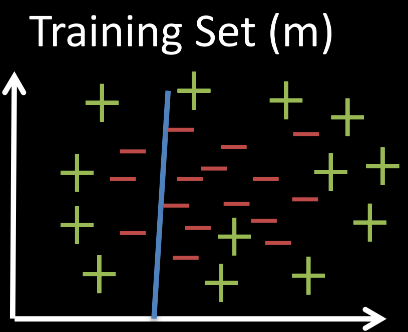
Bias and Variance







High Bias


Training Set (m)

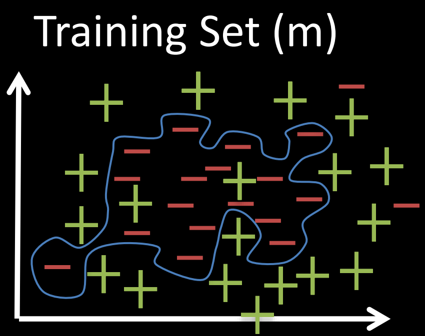

Error
Test Error
Training Error
Training set M

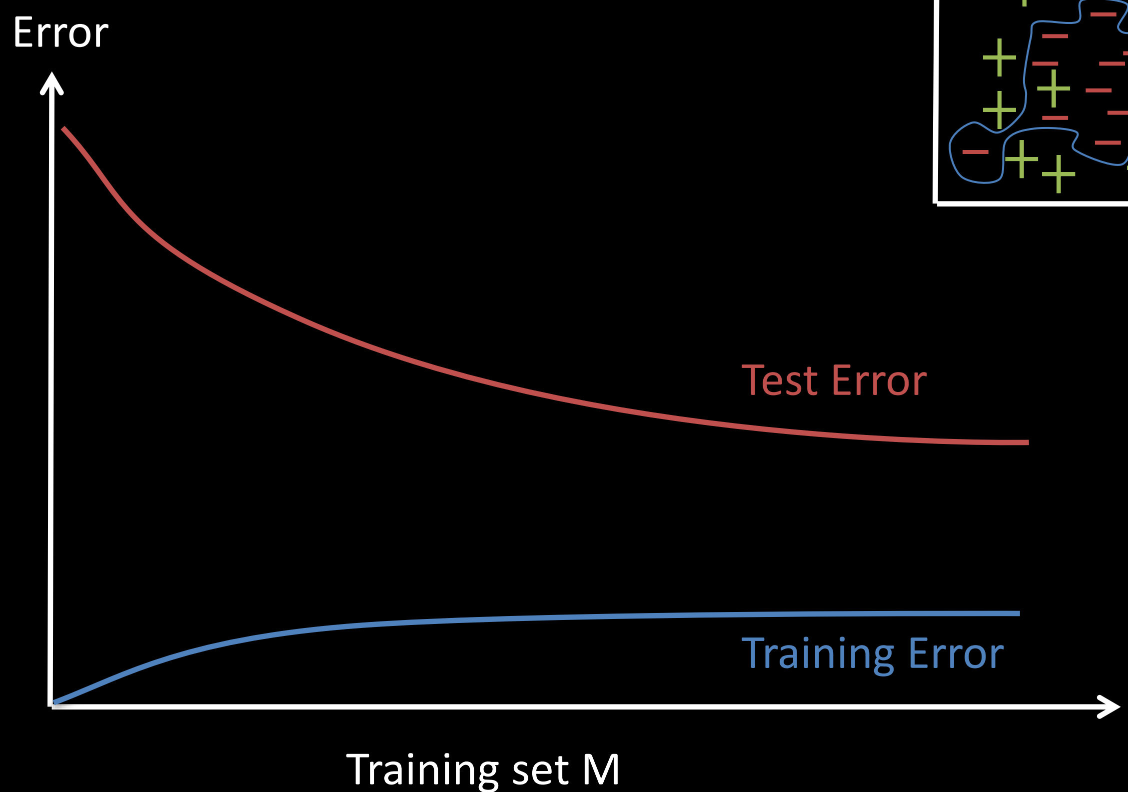
High Bias

Clicker: If you have high-bias, does more data help?

a) No
b) Yes

High Variance





Clicker: If you have high-variance, does more data help?

a) No
b) Yes

Clicker

1. **Get more training examples**
2. Try smaller sets of features
3. Try getting additional features
4. Try adding polynomial features (kernels)
5. Try increase regularization
6. Try decrease regularization

Helps with

A. High Variance
B. High Bias
C. Both
D. None

Clicker

1. Get more training examples
2. **Try smaller sets of features**
3. Try getting additional features
4. Try adding polynomial features (kernels)
5. Try increase regularization
6. Try decrease regularization

Helps with

A. High Variance
B. High Bias
C. Both
D. None

Clicker

1. Get more training examples
2. Try smaller sets of features
3. **Try getting additional features**
4. Try adding polynomial features (kernels)
5. Try increase regularization
6. Try decrease regularization

Helps with

A. High Variance
B. High Bias
C. Both
D. None

Clicker

1. Get more training examples
2. Try smaller sets of features
3. Try getting additional features
4. **Try adding polynomial features (kernels)**
5. Try increase regularization
6. Try decrease regularization

Helps with

A. High Variance
B. High Bias
C. Both
D. None

Clicker

1. Get more training examples
2. Try smaller sets of features
3. Try getting additional features
4. Try adding polynomial features (kernels)
5. **Try increase regularization**
6. Try decrease regularization

Helps with

A. High Variance
B. High Bias
C. Both
D. None

Clicker

1. Get more training examples
2. Try smaller sets of features
3. Try getting additional features
4. Try adding polynomial features (kernels)
5. Try increase regularization
6. **Try decrease regularization**

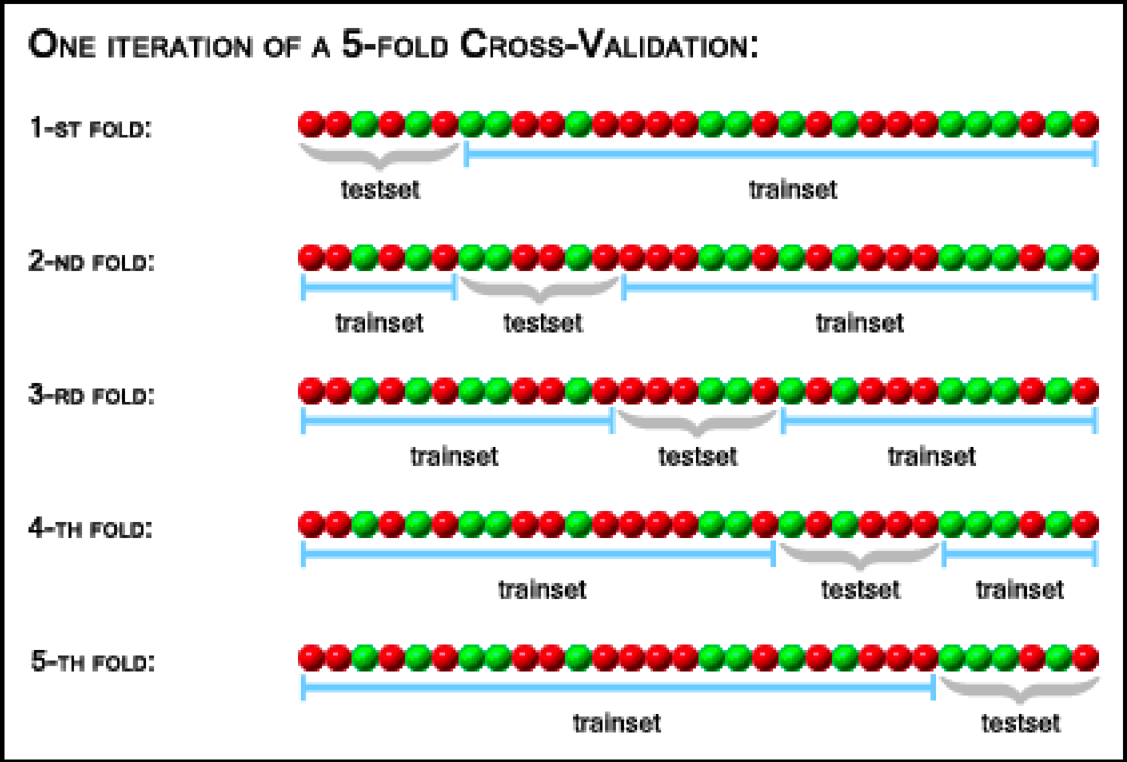
Helps with

A. High Variance
B. High Bias
C. Both
D. None

Cross-validation

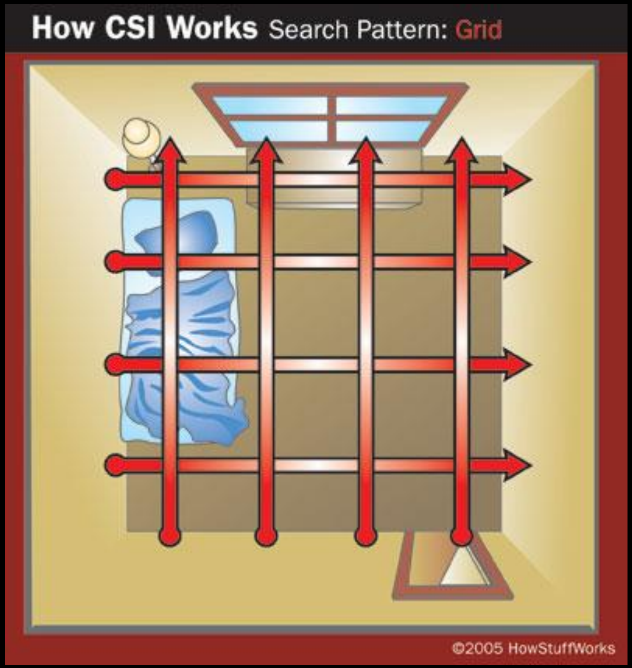
k-fold: split the data into k groups, train on every group except for one, which you test on.

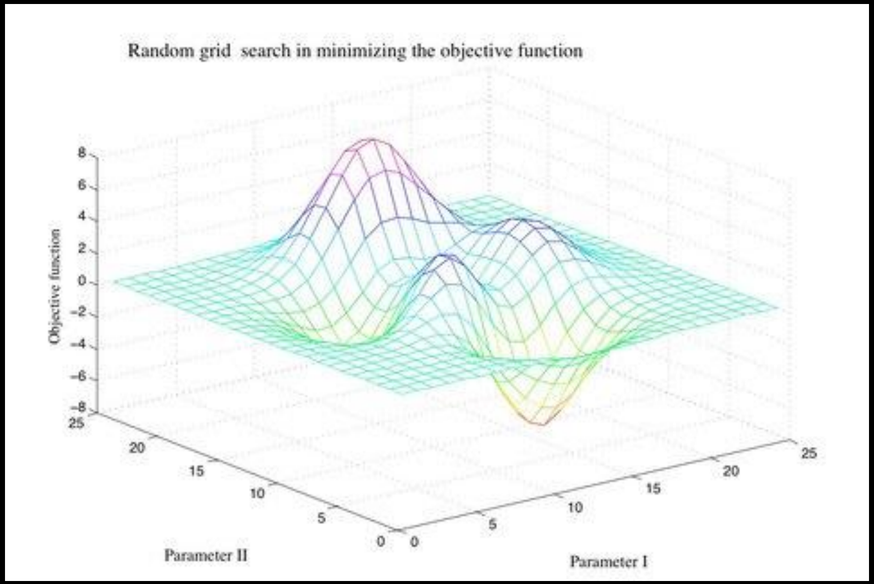
Repeat for all groups



Parameter Tuning

Grid Search





How to speed-up tuning?

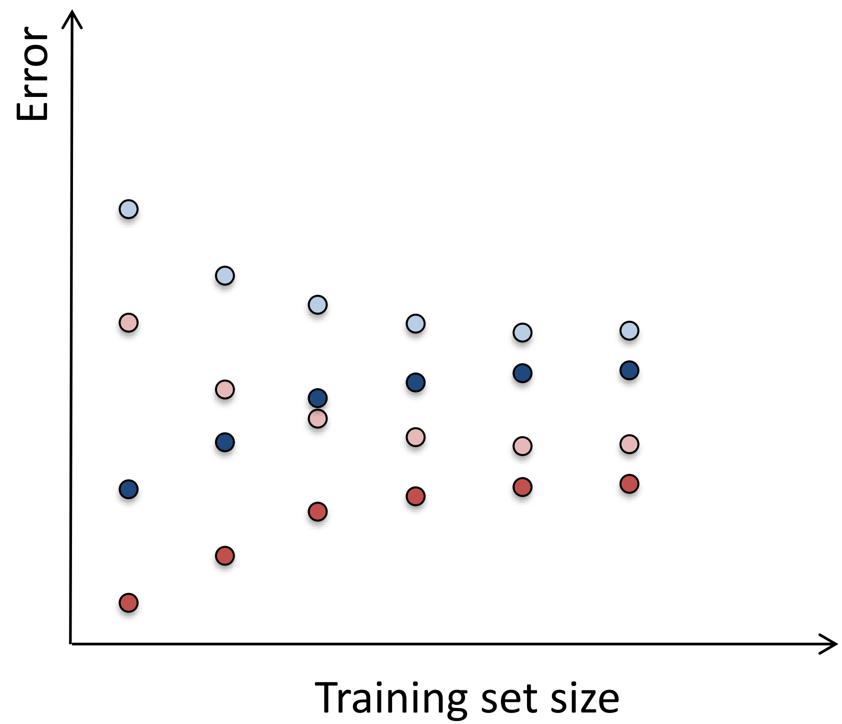
Can we use sampling?

Algorithm 1:

Training Validation

Algorithm 2:

Training Validation



How to speed-up tuning?

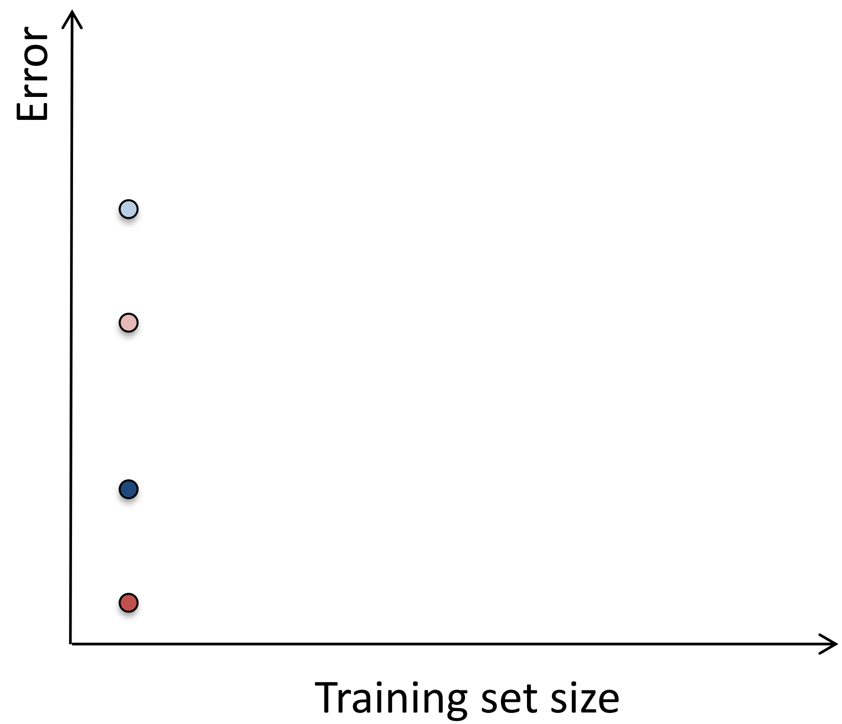
Can we use sampling?

Algorithm 1:

Training Validation

Algorithm 2:

Training Validation



How to speed-up tuning?

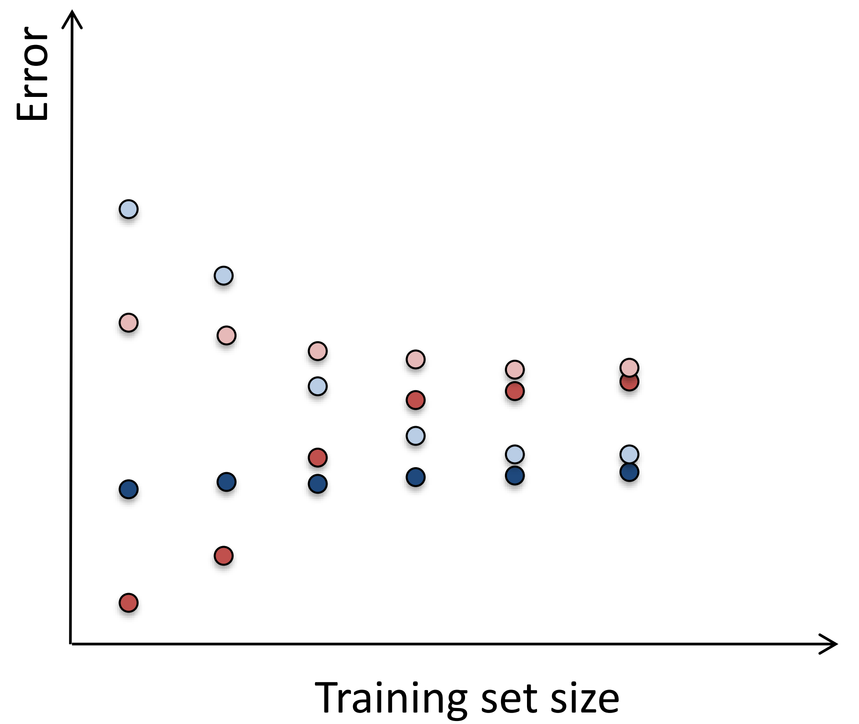
Can we use sampling?

Algorithm 1:

Training Validation

Algorithm 2:

Training Validation



How to speed-up tuning?

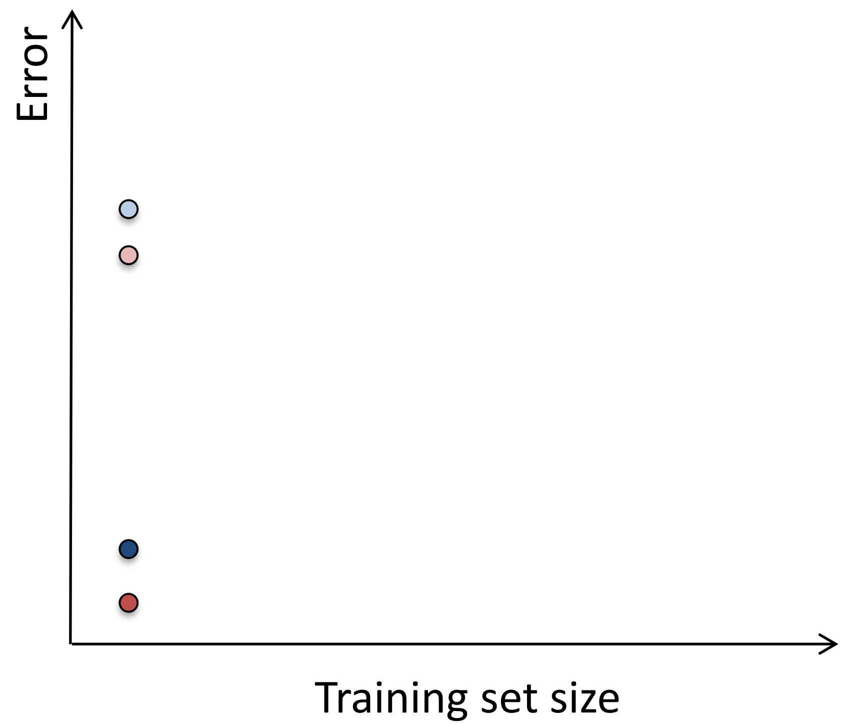
Can we use sampling?

Algorithm 1:

- Training
- Validation

Algorithm 2:

- Training
- Validation



How to speed-up tuning?

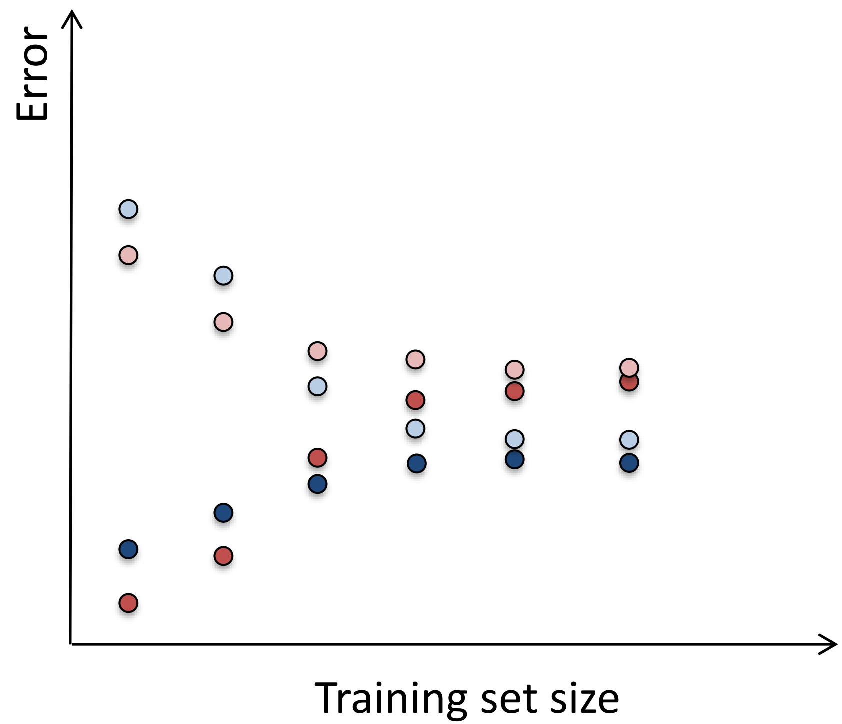
Can we use sampling?

Algorithm 1:

Training Validation

Algorithm 2:

Training Validation



How to speed-up tuning?

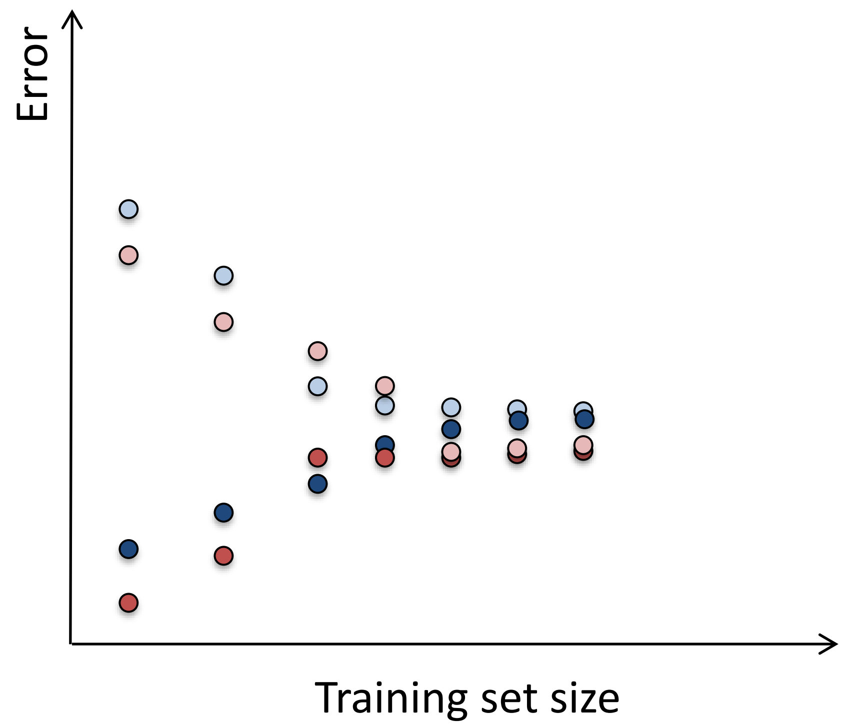
Can we use sampling?

Algorithm 1:

● Training ○ Validation

Algorithm 2:

● Training ○ Validation



Can we prune now?

How to speed-up tuning?

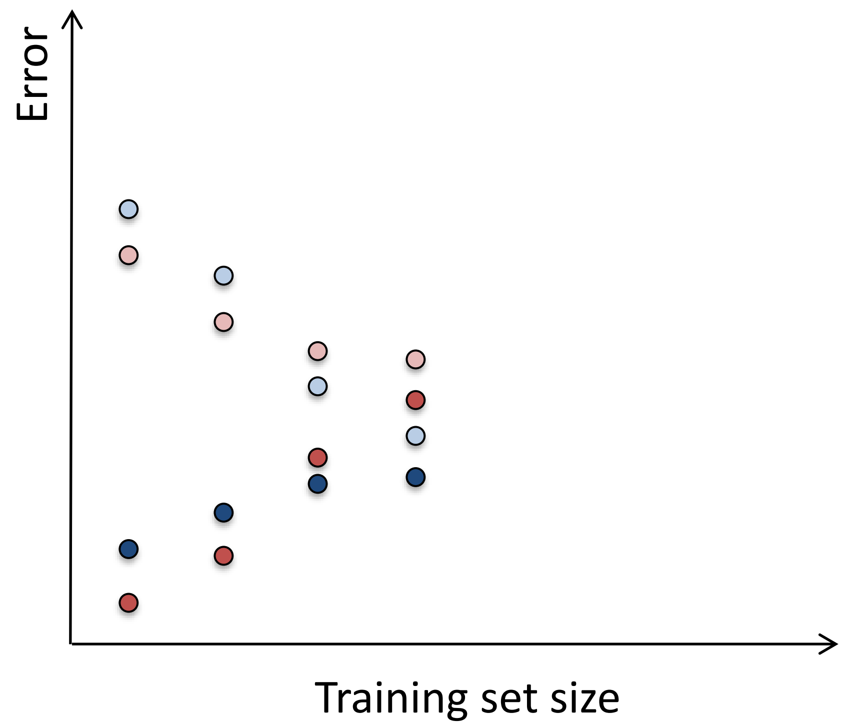
Can we use sampling?

Algorithm 1:

- Training
- Validation

Algorithm 2:

- Training
- Validation

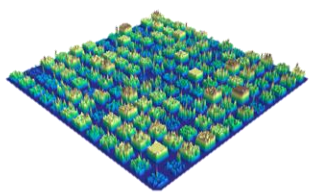


Algorithm 1 training error > Algorithm 2 validation error

Northstar's (now Einblick) AutoML

Built for *interactive results*,
unlike all other Auto-ML tools, which can take hours to produce results

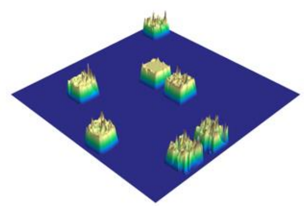
What modeling options do I have?



Rule-based Search Space Expansion

ML pipeline search space visualization

What should I try first?



Use past experience to optimize for expected quality per time-unit

How can I get some quick results?

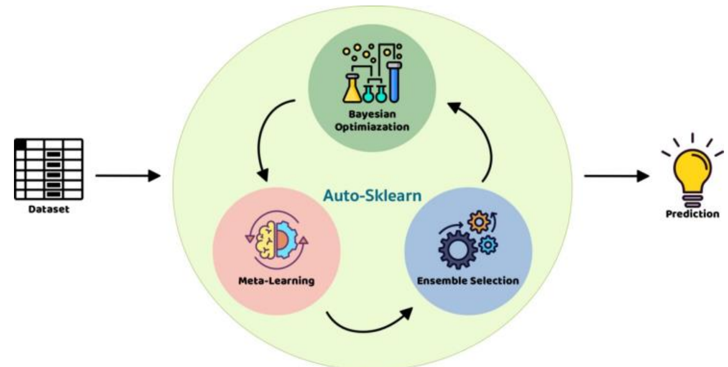
Adaptive sampling-based pruning and transfer learning

improve based on results

Zeyuan Shang, Emanuel Zgraggen, Benedetto Buratti, Ferdinand Kossmann, Philipp Eichmann, Yeounoh Chung, Carsten Binnig, Eli Upfal, Tim Kraska:
Democratizing Data Science through Interactive Curation of ML Pipelines.
SIGMOD Conference 2019: 1171-1188

Free to use for MIT students: https://einblick.ai/

AutoML Tools



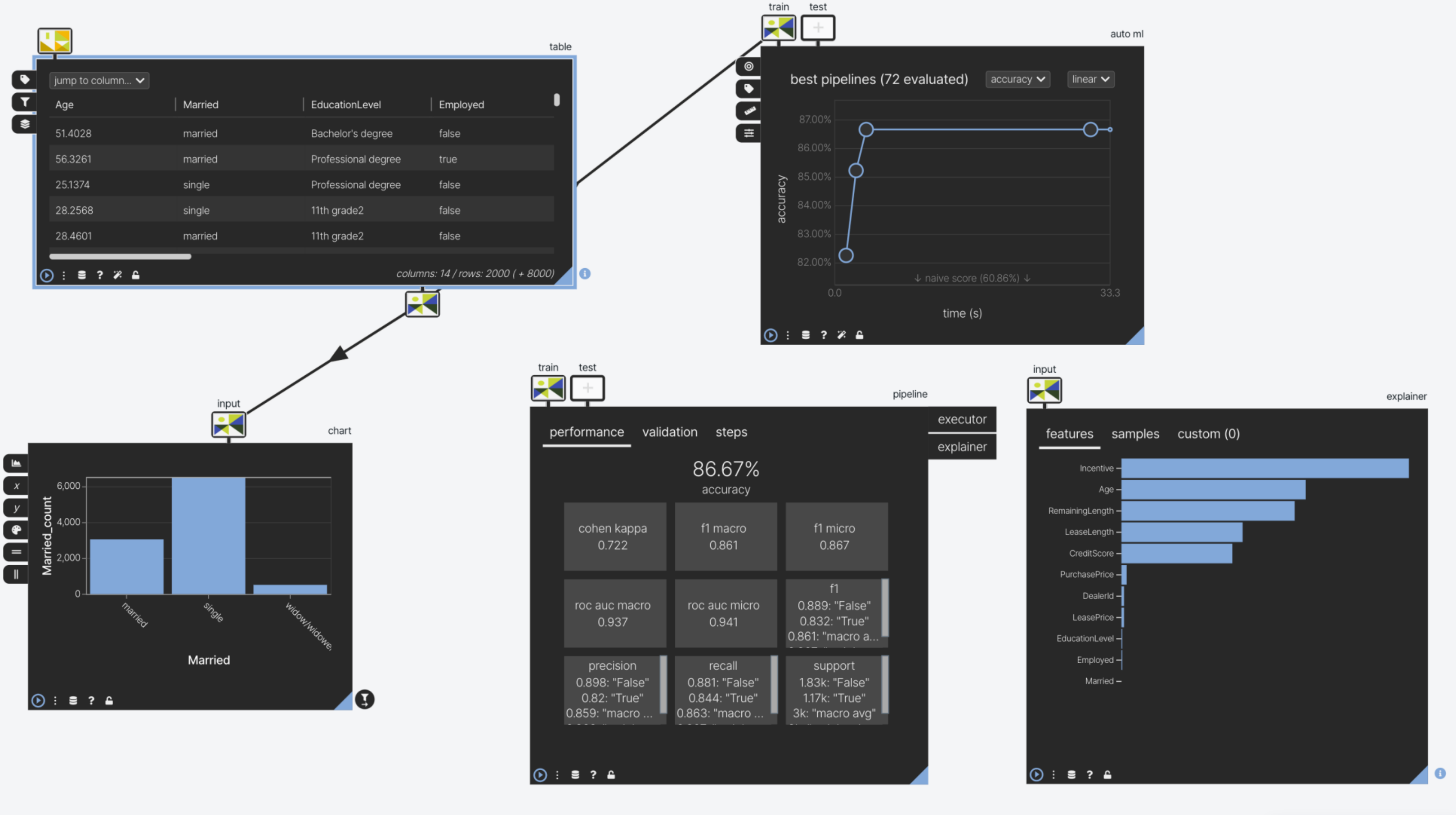
AutoKeras

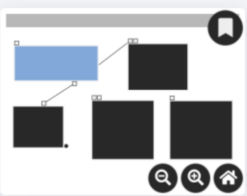
TPOT

H_2O AutoML

einblick

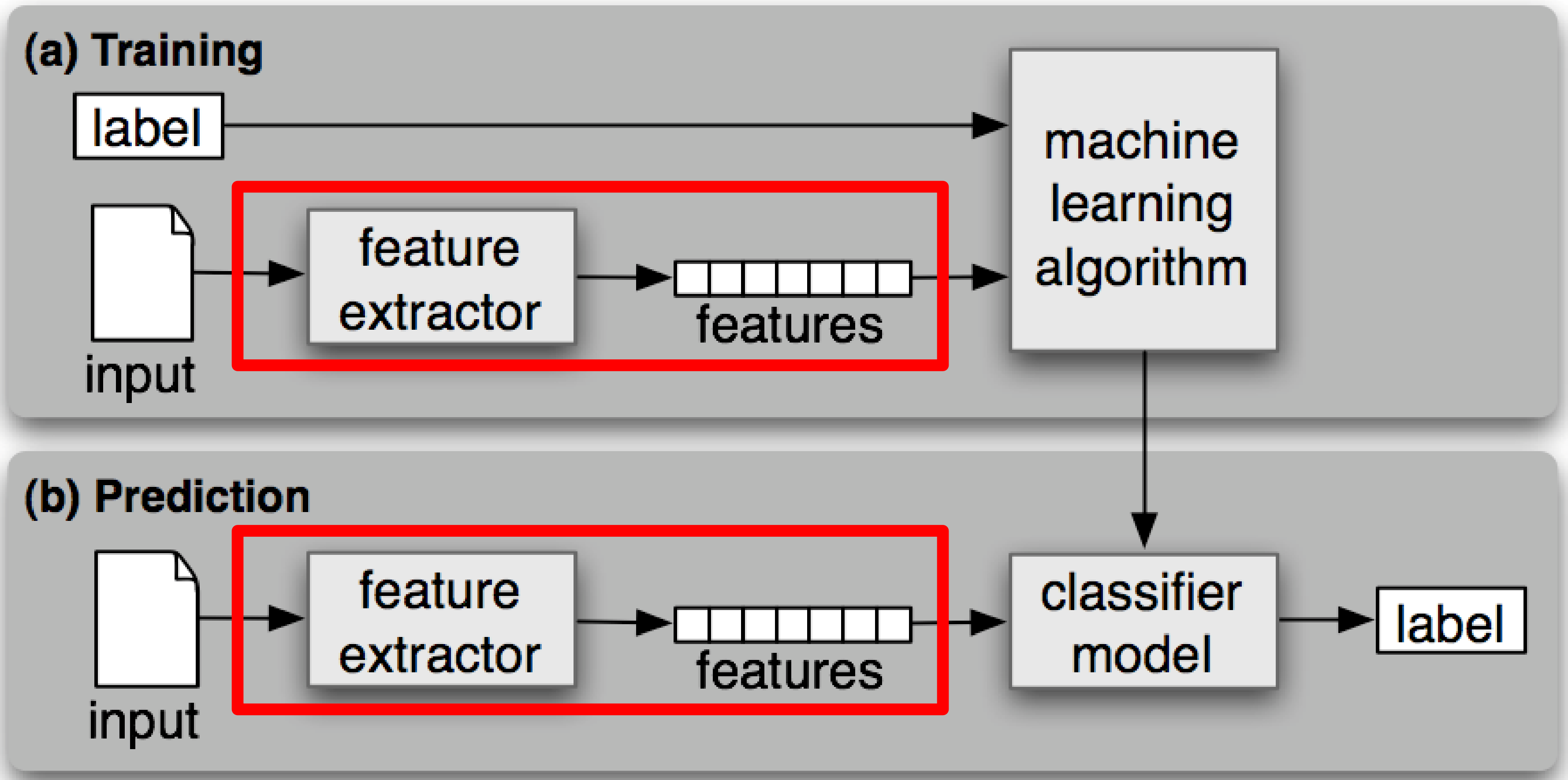
....





https://einblick.ai/

Feature Engineering



Class Task: Feature Engineering

How would you predict the unemployment rate before the official numbers come out?

https://www.washingtonpost.com/news/wonk/wp/2014/04/07/twitter-is-surprisingly-accurate-at-predicting-unemployment/

Feature engineering

- Dropping features
  - Remove duplicates
  - Highly correlated values (Zip code, Lon/Lat)

- Feature creation
  - Feature crosses: Cost per square feet
  - Creating special features ("I lost my job")
  - Row statistics
    - Number of 0, nulls, negative value, mean, max, min,…
  - Projection to circle
    - Turn a single feature (like day_of_week) into two coordinates on a circle
    - Ensures that distance between Monday and Sunday etc is the same
  - Spatial
    - GPS encoding
    - Categorized locations (e.g., close to city, rural, nearby hospital, etc.)
  - Use embeddings from other models (more on that later)
  - Discretization (date → weekend/weekday)
  - …

Transformations

- Rounding
  - Lossy
  - Precision can just be noise -> might improve training
  - Log transform before rounding often useful
- Binning
  - Removes information
  - Can work gracefully with variables outside of ranges seen in the train set
- Scaling
  - Sandard (Z) Scaling
  - MinMax Scaling
  - Root Scaling
  - Log Scaling
- Outlier removal
- Imputation (mean, median, …)
- Interaction encoding : Specifically encodes the interaction between two numerical variables
  - Substraction, Addition, Multiplication….
  - **Polynomial encoding**
    - Linear algorithms can not solve XOR problem
    - A polynomial kernel can solve XOR

Encodings

- **One-hot**
- NaN, null, etc → create explicit encoding
- **Hash-encoding** (careful might introduce collisions)
- **Count encoding**: replace categorical value with their count
  - Useful for both linear and non-linear algorithms
  - Sensitive to outliers
  - Might create collisions
- **Rank encoding**: Rank categorical variables by count in train set
  - Useful for both linear and non-linear algorithms
  - Not sensitive to outliers
  - Won't give same encoding to different variables
- **Target encoding**: Encode categorical variables by their ratio of target (binary classification) in train set
  - Be careful to avoid overfit
  - Add smoothing to avoid setting variable encoding to 0
  - Add random noise?
  - Can work extremely well when done right
- **Consolidation/expansion encoding**: map different categorical variables to the same
  - Spelling errors, slightly different job descriptions, abbreviations

Example: Customer Conversion

Questions:

- What sequence of emails, phone calls, showing ads in the platform, etc. leads to the highest conversion rate (e.g., open an account)
- How many emails/phone calls are too much?
- What should the timing be?
- How can we annotate the data with custom information we have available (notes from 1on1 conversations, portal logins, current accounts, etc.)?
- How do we quickly adjust to changing conditions (e.g., covid happens, increasing interest rates, inflation,…)?

→ This is not a standardized process. Rather it requires to quickly experiment with new models

The data

Customer	Date	Event	Info
Tim	2022/01/03	E11	Promotion email to savings bank account
Paul	2022/01/03	E11	Promotion email to savings bank account
Tim	2022/01/04	P10	Phone call.
Mark	2022/01/05	E11	Promotion email to savings bank account
Tim	2022/01/05	E11	Promotion email to start saving
Tim	2022/01/06	D1	Display ad regarding savings account
Paul	2022/01/07	D1	Display ad regarding savings account
Tim	2022/01/08	B1	User opens saving account
…	…	…	…

Customer	Event	Log
Tim	1on1 Meeting	Had a good meeting. Expressed some interest in savings account but was worried about inflation
Paul	1on1 Meeting	Good conversation, but expressed no interest in savings accounts
Mark	Phone call	Talked about insurance and mortgages
…	…	…

Customer	Account	Balance
Tim	Checking	10k
Paul	Checking	20k
Tim	Trading	40k
Mark	Checking	15k
…	…	…

…

Example: Customer Conversion

The data

Customer	Date	Event	Info
Tim	2022/01/03	E11	Promotion email to savings bank account
Paul	2022/01/03	E11	Promotion email to savings bank account
Tim	2022/01/04	P10	Phone call.
Mark	2022/01/05	E11	Promotion email to savings bank account
Tim	2022/01/05	E11	Promotion email to start saving
Tim	2022/01/06	D1	Display ad regarding savings account
Paul	2022/01/07	D1	Display ad regarding savings account
Tim	2022/01/08	B1	User opens saving account
…	…	…	…

How would you build a model
over this data to predict
if a user opens a a new savings account?

Customer	Event	Log
Tim	1on1 Meeting	Had a good meeting. Expressed some interest in savings account but was worried about inflation
Paul	1on1 Meeting	Good conversation, but expressed no interest in savings accounts
Mark	Phone call	Talked about insurance and mortgages
…	…	…

Customer	Account	Balance
Tim	Checking	10k
Paul	Checking	20k
Tim	Trading	40k
Mark	Checking	15k
…	…	…

…

Example: Customer Conversion

Customer	Date	Event	Info
Tim	2022/01/03	E11	Promotion email to savings bank account
Paul	2022/01/03	E11	Promotion email to savings bank account
Tim	2022/01/04	P10	Phone call.
Mark	2022/01/05	E11	Promotion email to savings bank account
Tim	2022/01/05	L1	User logged into platform
Tim	2022/01/06	D1	Display ad regarding savings account
Paul	2022/01/07	L1	Display ad regarding savings account
Tim	2022/01/08	B1	User opens saving account
…	…	…	…

Customer	Event	Log
Tim	1on1 Meeting	Had a good meeting. Expressed some interest in savings account but was worried about inflation
Paul	1on1 Meeting	Good conversation, but expressed no interest in savings accounts
Mark	Phone call	Talked about insurance and mortgages
…	…	…

Customer	Account	Balance
Tim	CK1	10k
Paul	CK2	20k
Tim	TD	40k
Mark	CK1	15k
…	…	…

…

Customer	Checking Account	Trading Account	Looking for Savings account	Last Phone outreach	Last Email	Last Email Encoding	Previous Email	Previous Email Encoding	Conversation Encoding	Next Action	Result open account
Tim	Yes	Yes	Yes	2022/01/04	2022/01/05	32423423	2022/01/03	084308934	7897979	$100 Promotion Email	80%
Paul	Yes	No	No	-	2022/01/03	234234234	-	23947829374	797897	Email promotion	2%
Mark	Yes	No	unknown	-	2022/01/05	23423424	-	2349729347	7789789	Phone call	3%
…	…	…	…	…	…	…	…	…	…	…	…

Other time-related feature engineering tricks

- Trendlines
  - Instead of encoding: total spend, encode things like: spend in last week, spend last month, spend in last year
  - Gives a trend to the algorithm

- Closeness to major events
  - Hardcode categorical features like date_3_days_before_holidays
  - Try national holidays, major sport events, weekends, end of quarter, etc. → All can have impact on spending behavior

- Projection to circle

Word embeddings

- **Idea:** learn an embedding from words into vectors

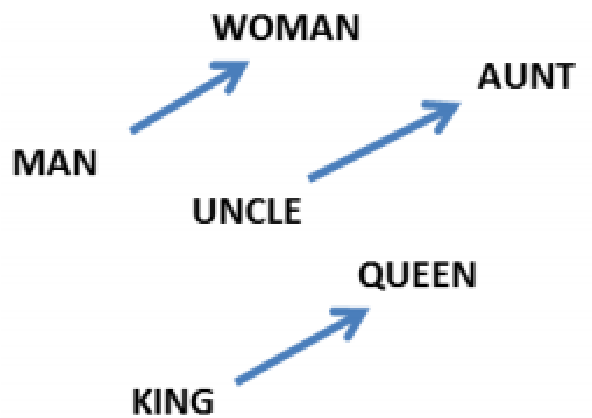
  Cat – {0.002, 0.244, 0.546, …, 0.345}

- Need to have a function W(word) that returns a vector encoding that word.

- Applications: ???

Word embeddings: properties

Relationships between words correspond to difference between vectors.



$$W(\text{``woman"}) - W(\text{``man"}) \simeq W(\text{``aunt"}) - W(\text{``uncle"})$$

$$W(\text{``woman"}) - W(\text{``man"}) \simeq W(\text{``queen"}) - W(\text{``king"})$$

http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

Word embeddings: questions

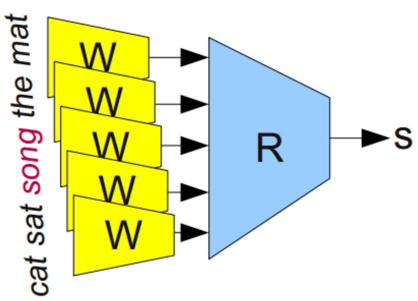
- How big should the embedding space be?
  - Trade-offs like any other machine learning problem – greater capacity versus efficiency and overfitting.

- How do we find W?
  - Often as part of a prediction or classification task involving neighboring words.

Learning word embeddings

https://arxiv.org/ftp/arxiv/papers/1102/1102.1808.pdf

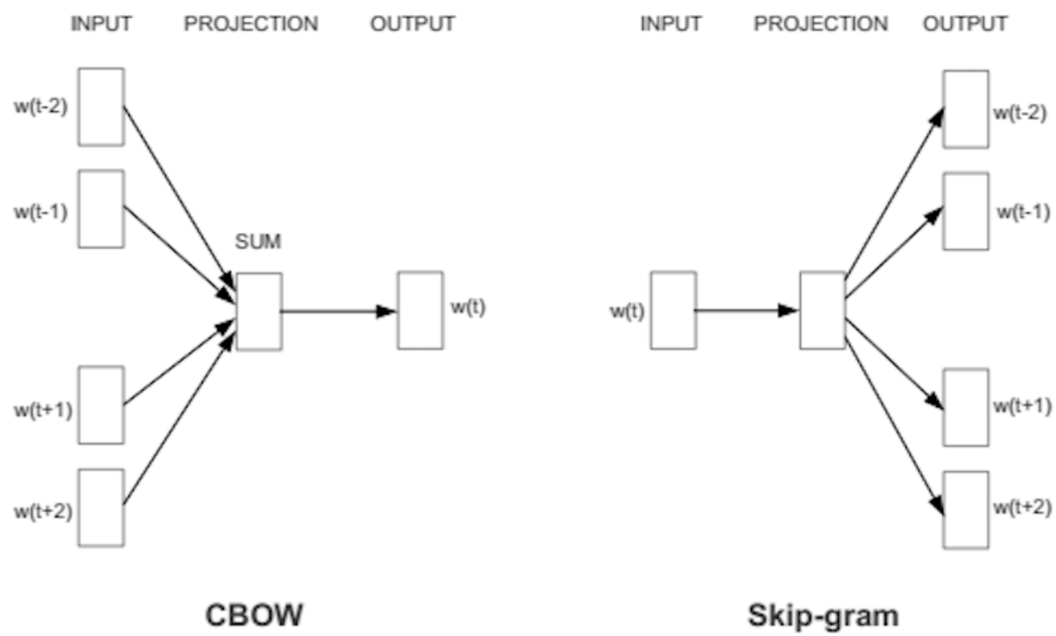
- First attempt:
  - Input data is sets of 5 words from a meaningful sentence. E.g., "one of the best places". Modify half of them by replacing middle word with a random word. "one of function best places"
  - W is a map (depending on parameters, Q) from words to 50 dim'l vectors. E.g., a look-up table or an RNN.
  - Feed 5 embeddings into a module R to determine 'valid' or 'invalid'
  - Optimize over Q to predict better



http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

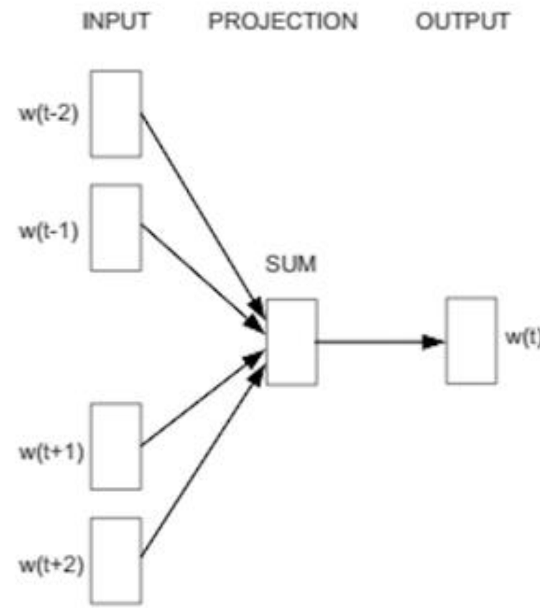
Word-Embeddings: word2vec

- Predict words using context
- Two versions: CBOW (continuous bag of words) and Skip-gram



Continuous Bag of words (CBOS

- Bag of words (BOW)
  - Gets rid of word order. Used in discrete case using counts of words that appear.
- CBOW
  - Takes vector embeddings of n words before target and n words after and adds them (as vectors).
  - Also removes word order, but the vector sum is meaningful enough to deduce missing word.



CBOW

Continuous Bag of Words - Window Size 2

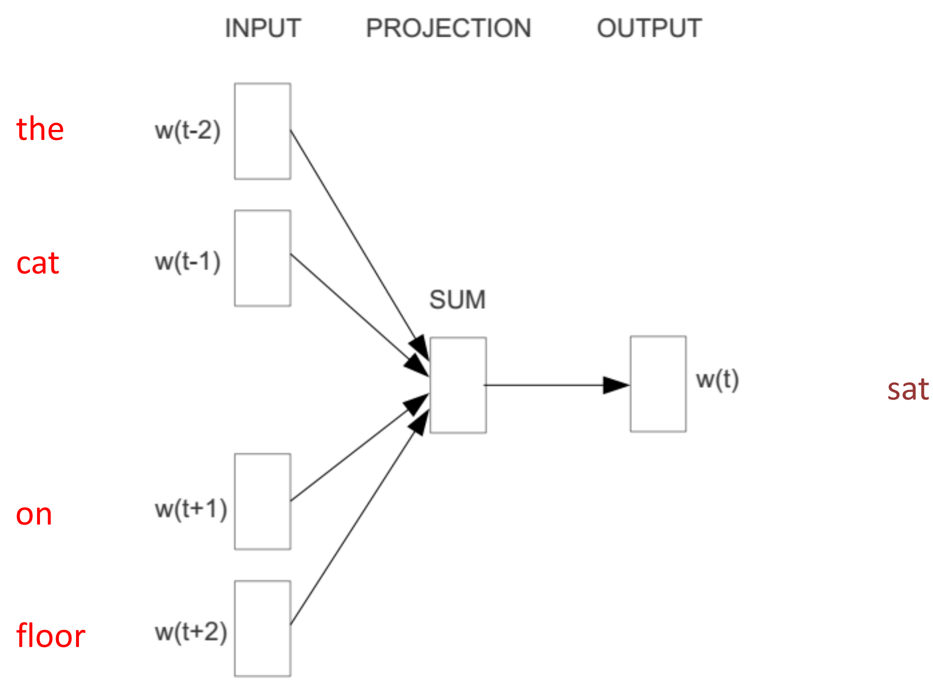
Jay was hit by a ______ bus in...

by	a	red	bus	in

input 1	input 2	input 3	input 4	output
by	a	bus	in	red

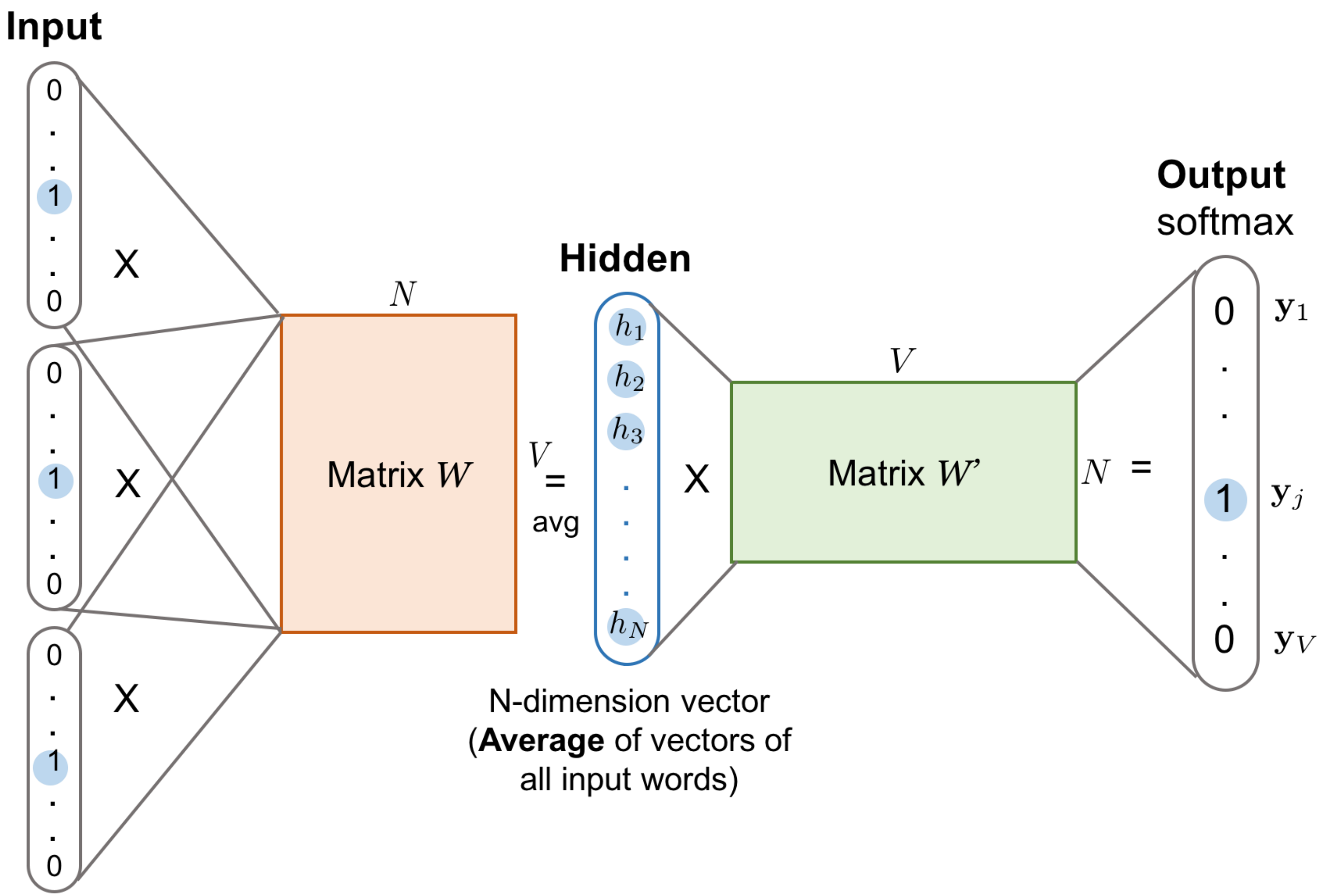
Continuous Bag of Word

- E.g. "The cat sat on floor"
  - Window size = 2

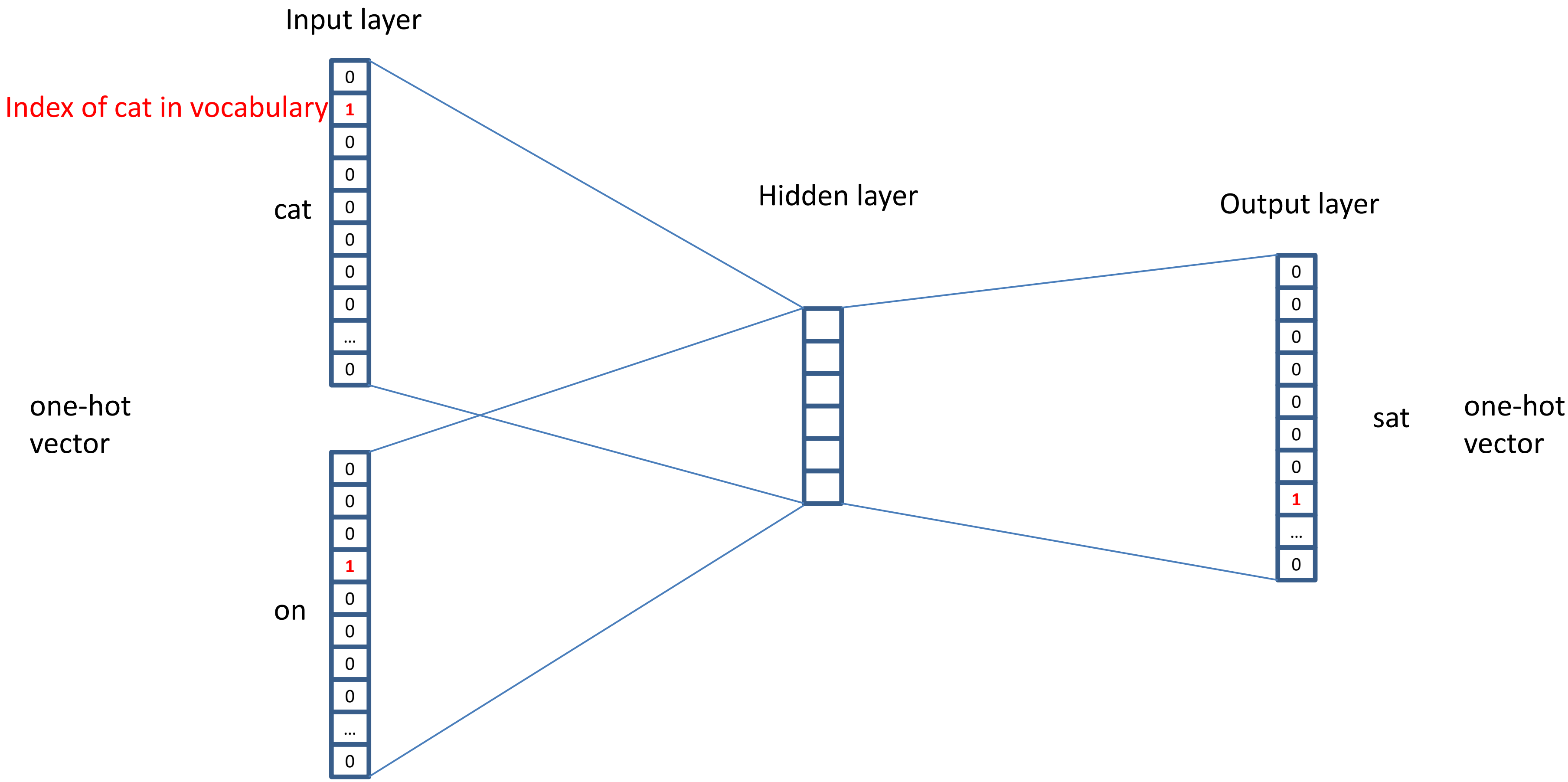


www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx

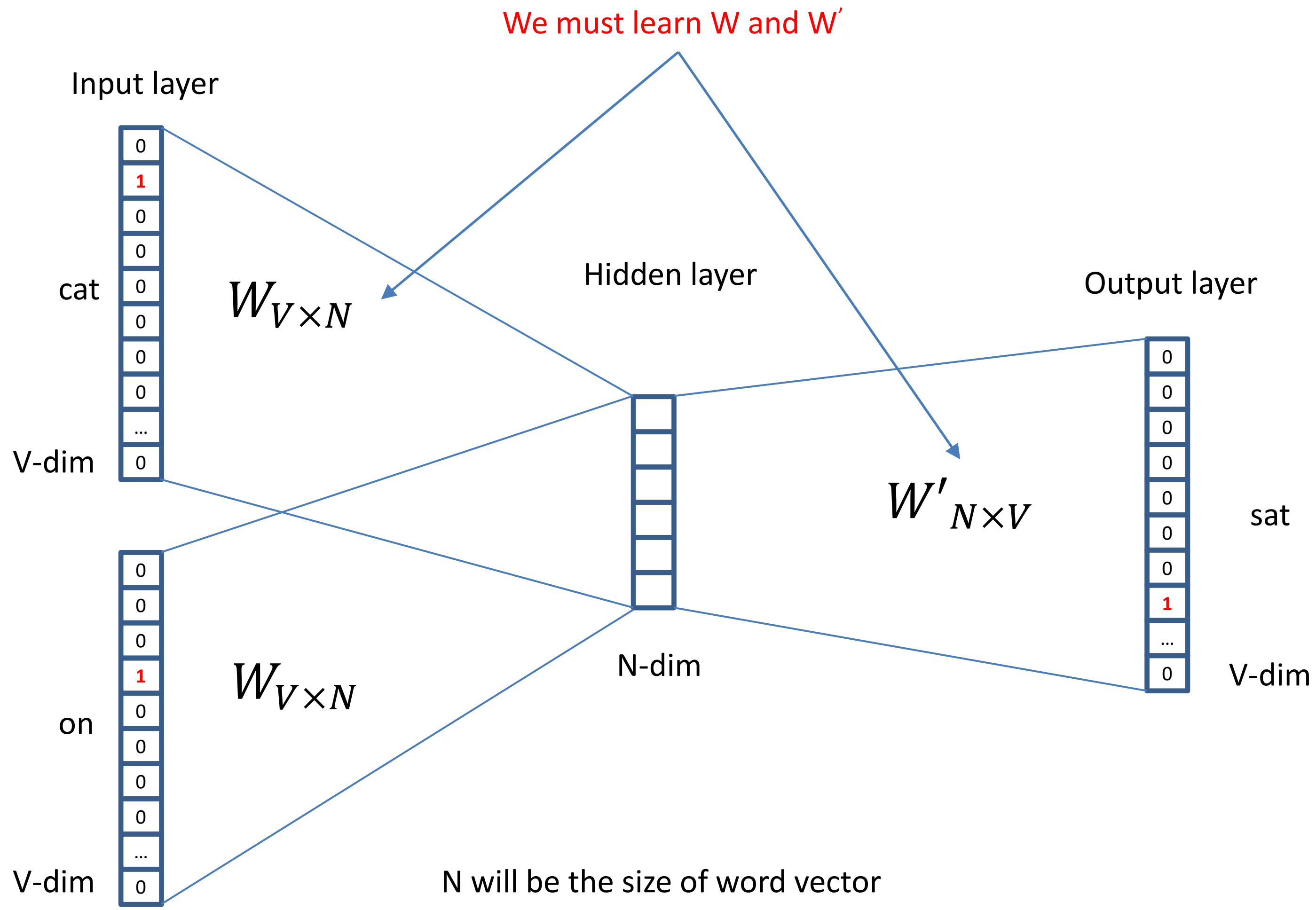
COBW



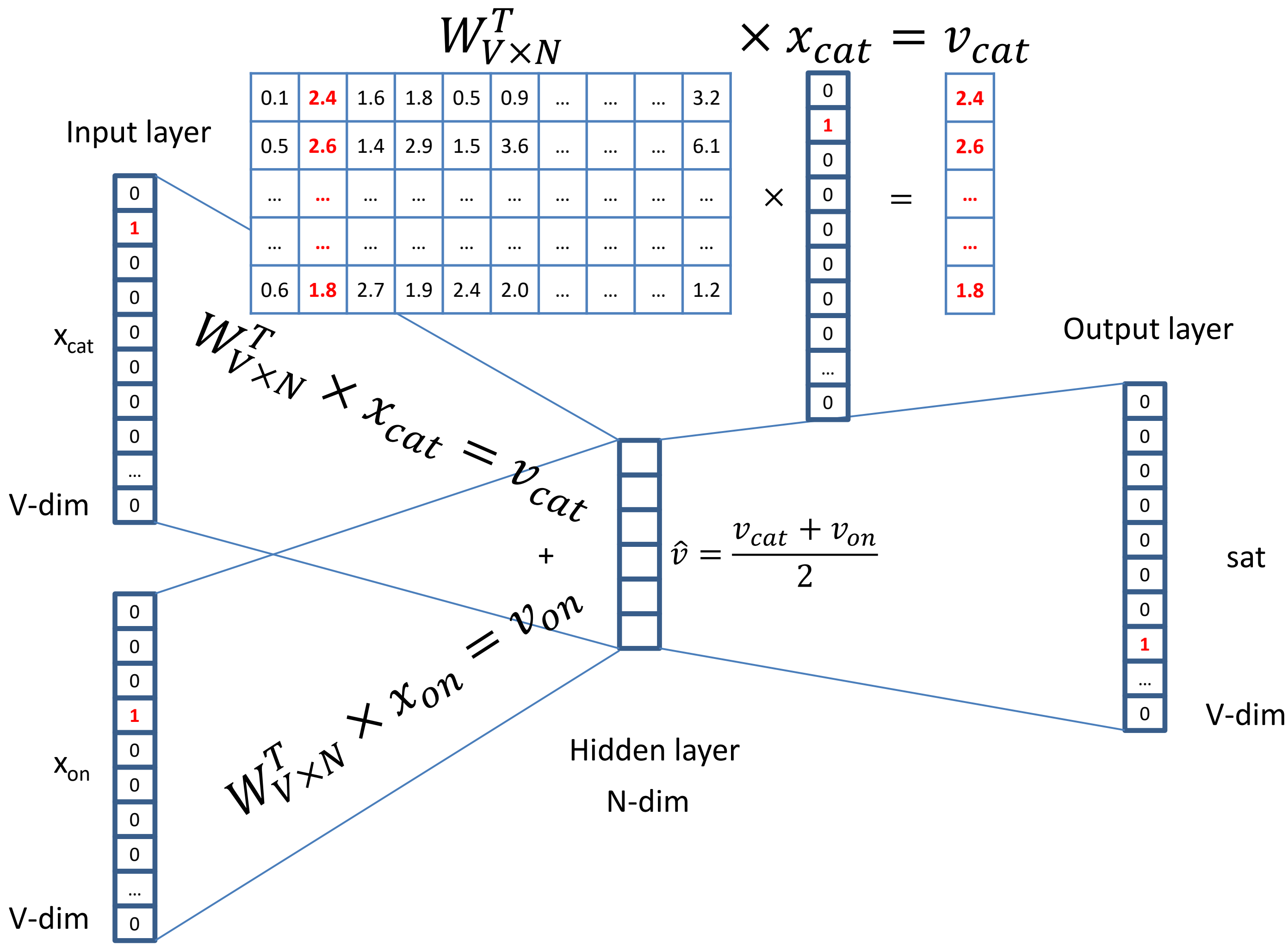
Input
0
.
.
1
.
.
0
X
0
.
.
1
.
.
0
X
0
.
.
1
.
.
0
X
N
Matrix W
V
=
avg
Hidden
h_1
h_2
h_3
h_N
X
V
Matrix W'
N
=
Output
softmax
0
1
0
$\mathbf{y}_1$
$\mathbf{y}_j$
$\mathbf{y}_V$
N-dimension vector
(**Average** of vectors of
all input words)



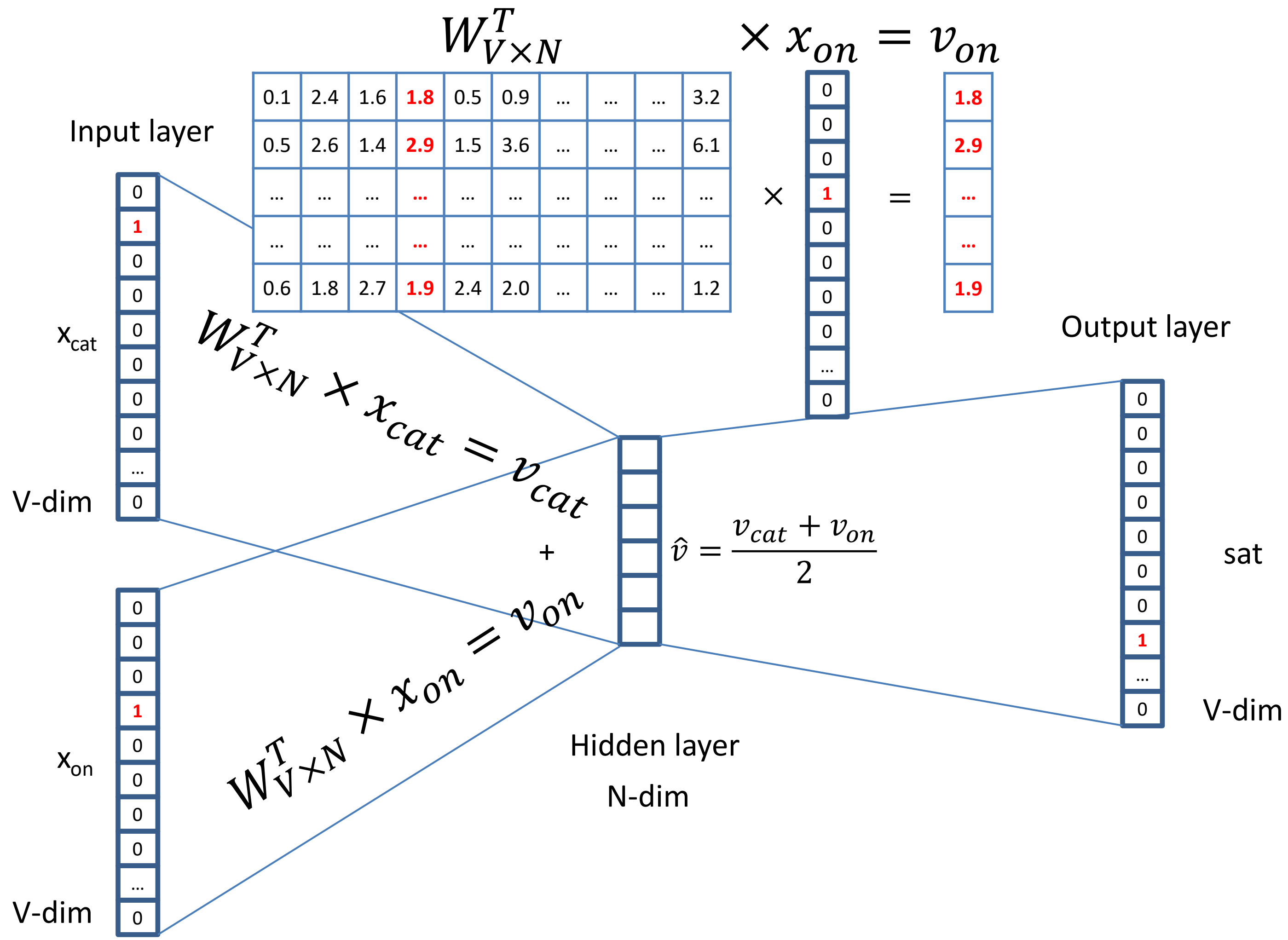
www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx



www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx



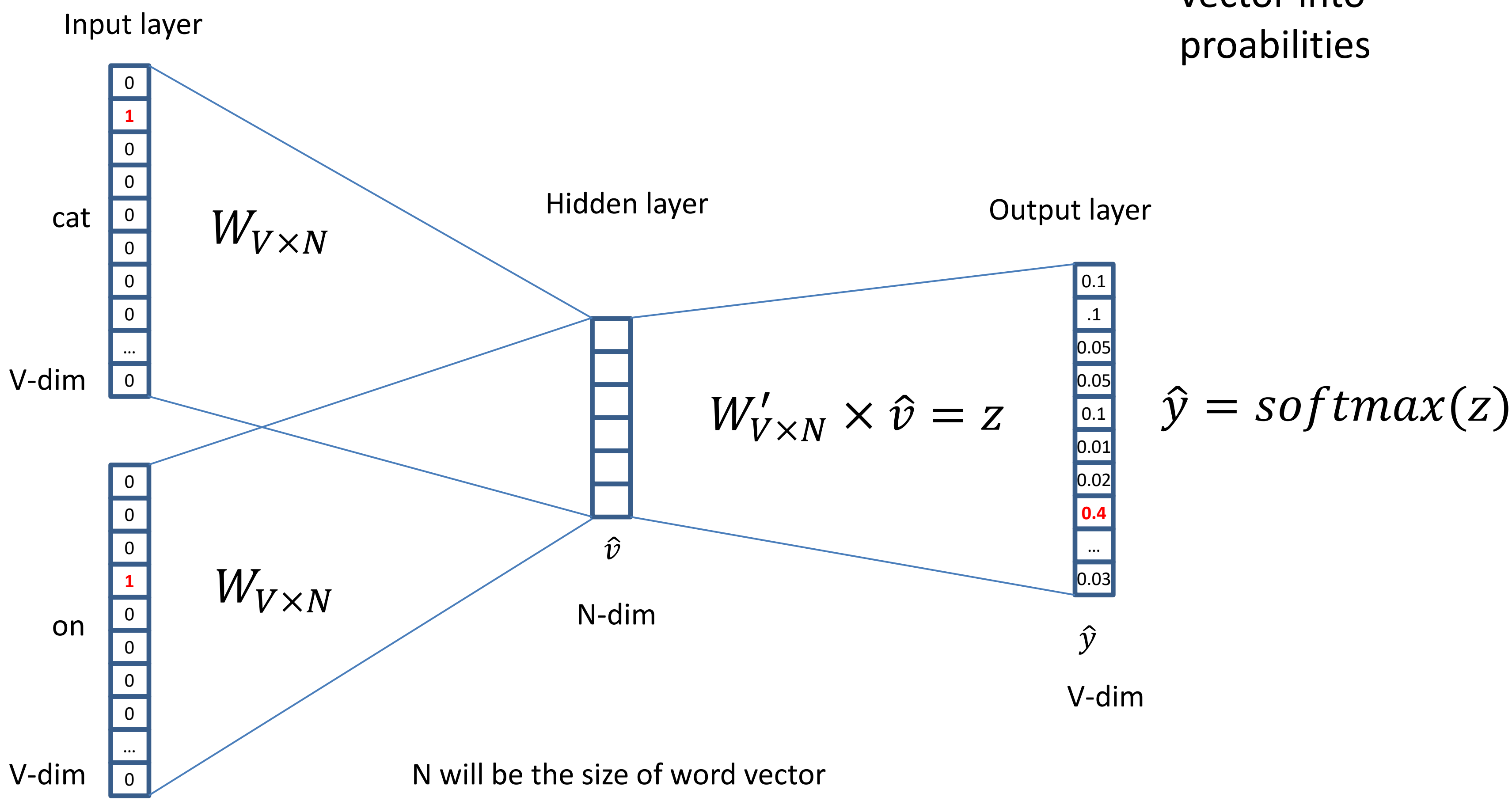
www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx



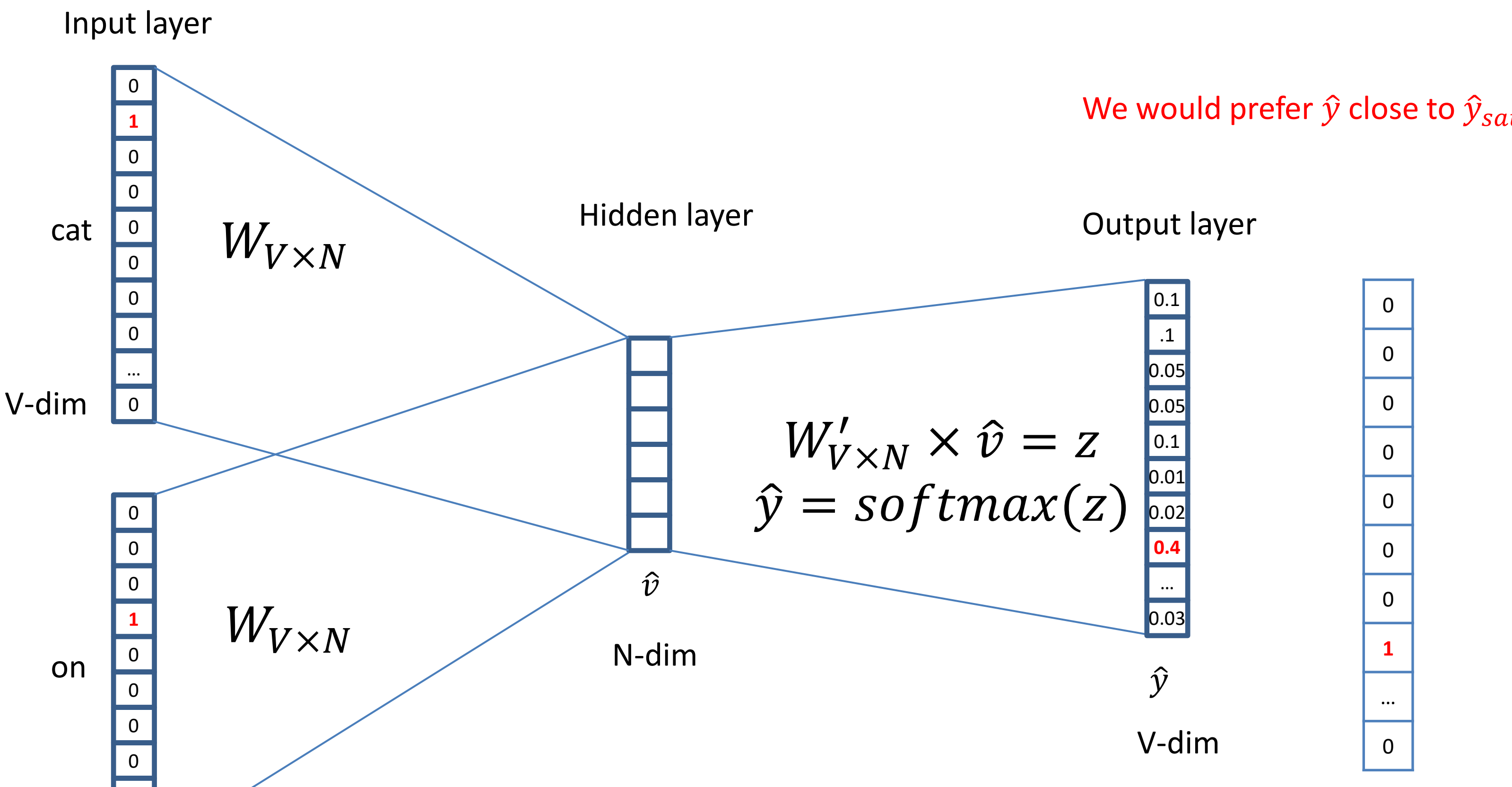
www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Softmax turns the vector into proabilities

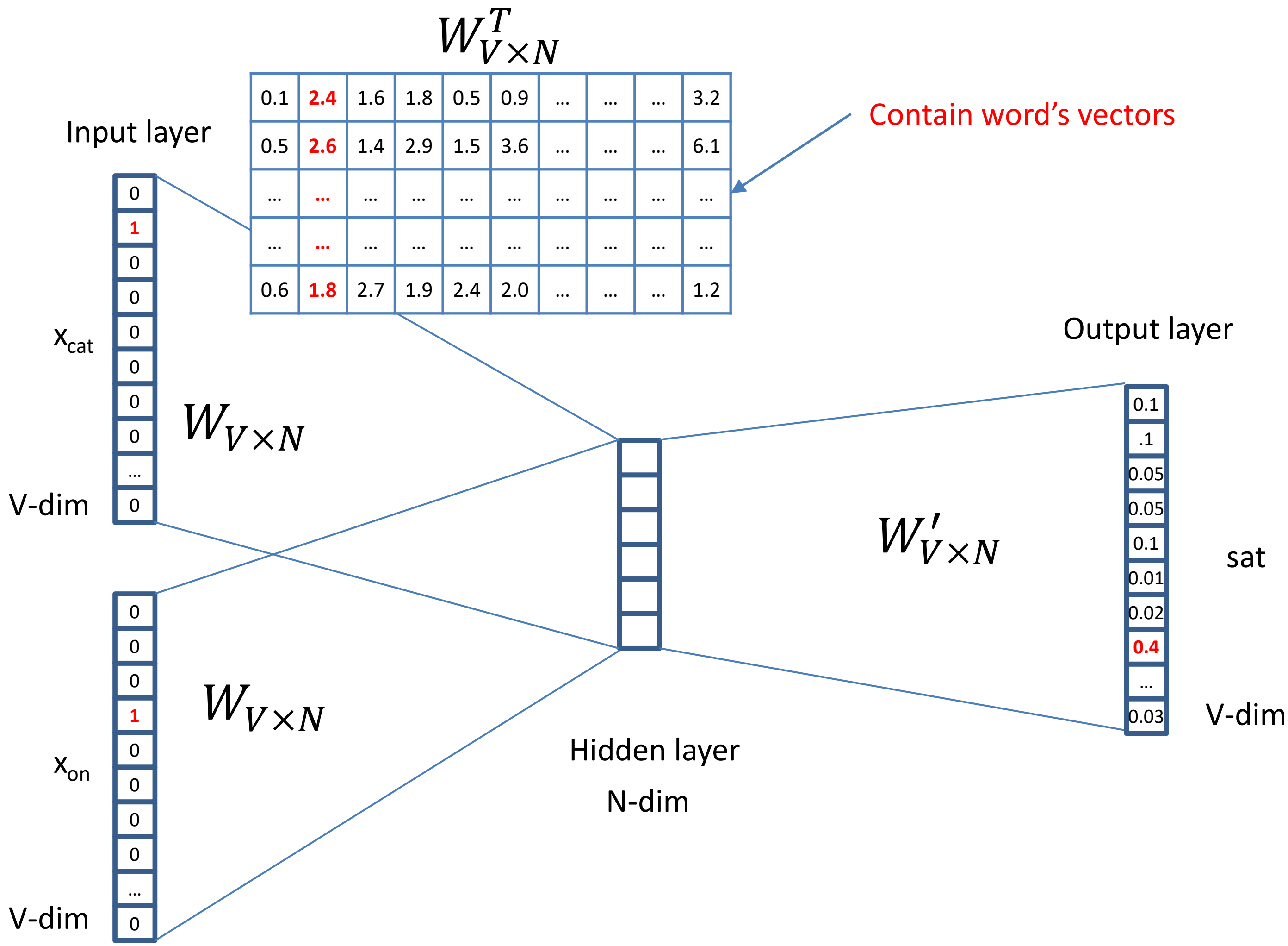


www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx



N will be the size of word vector

www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx



We can consider either W or W' as the word's representation. Or even take the average.

www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx

Some interesting results

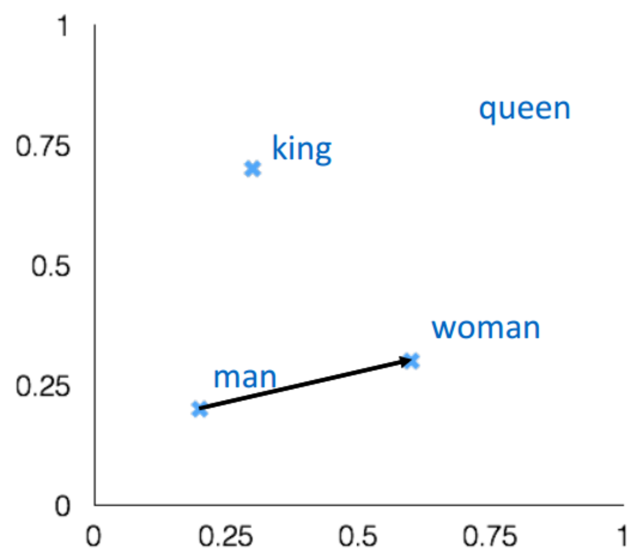
Word Analogies

Test for linear relationships, examined by Mikolov et al. (2014)

a:b :: c:?

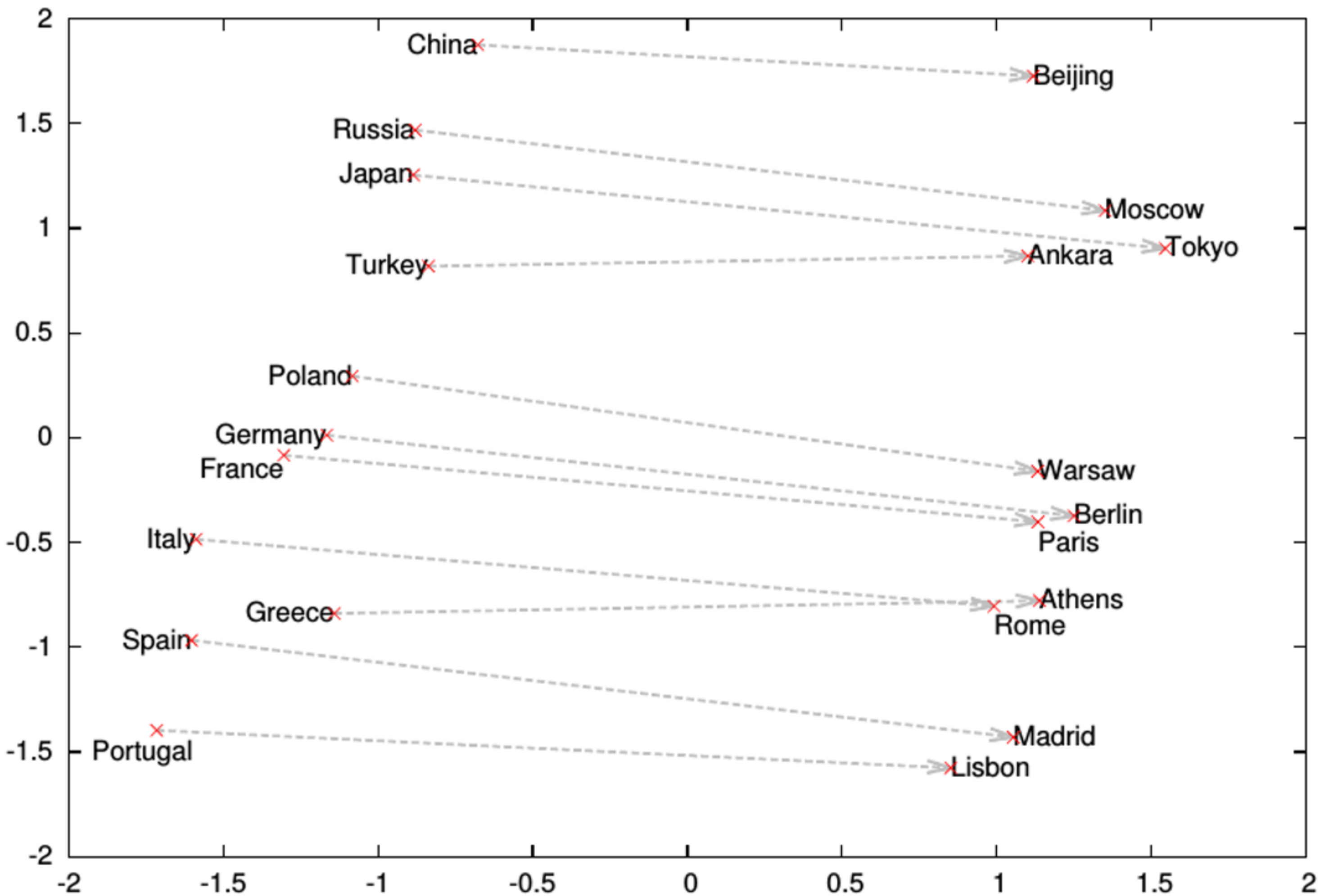
man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]



www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx

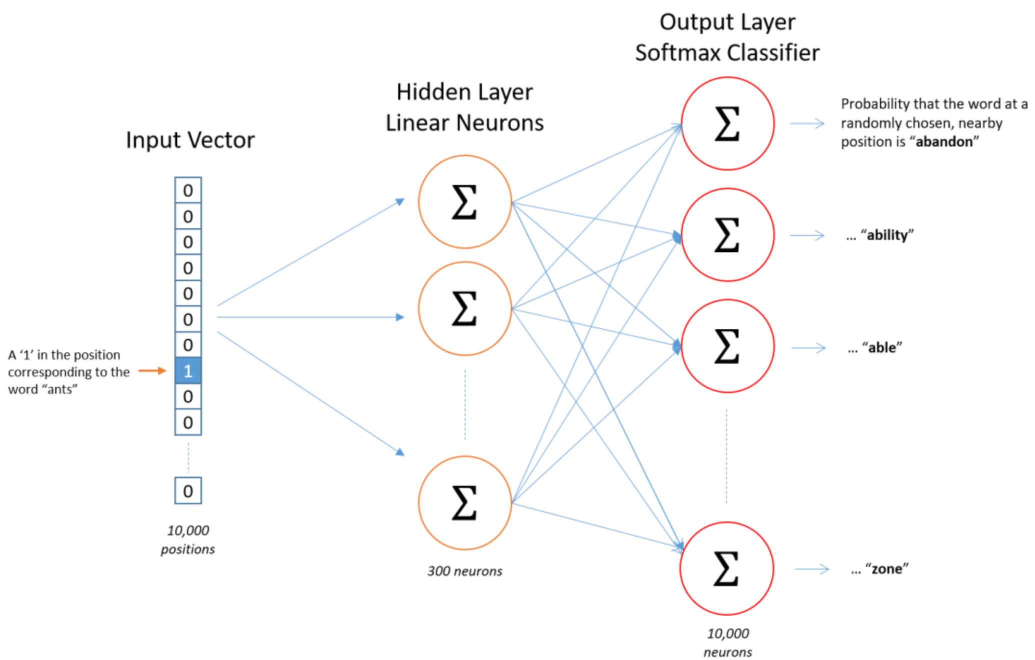
Word analogies



www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx

Skip gram

- Map from center word to probability on surrounding words. One input/output unit below.
- Start with a single word embedding and try to predict the surrounding words.
- Much less well-defined problem, but works better in practice (scales better).



http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Skip Gram (window 2)

Sam	likes	Celine	Dion	and	biking

Sam	likes	Celine	Dion	and	biking

Sam	likes
Sam	Celine

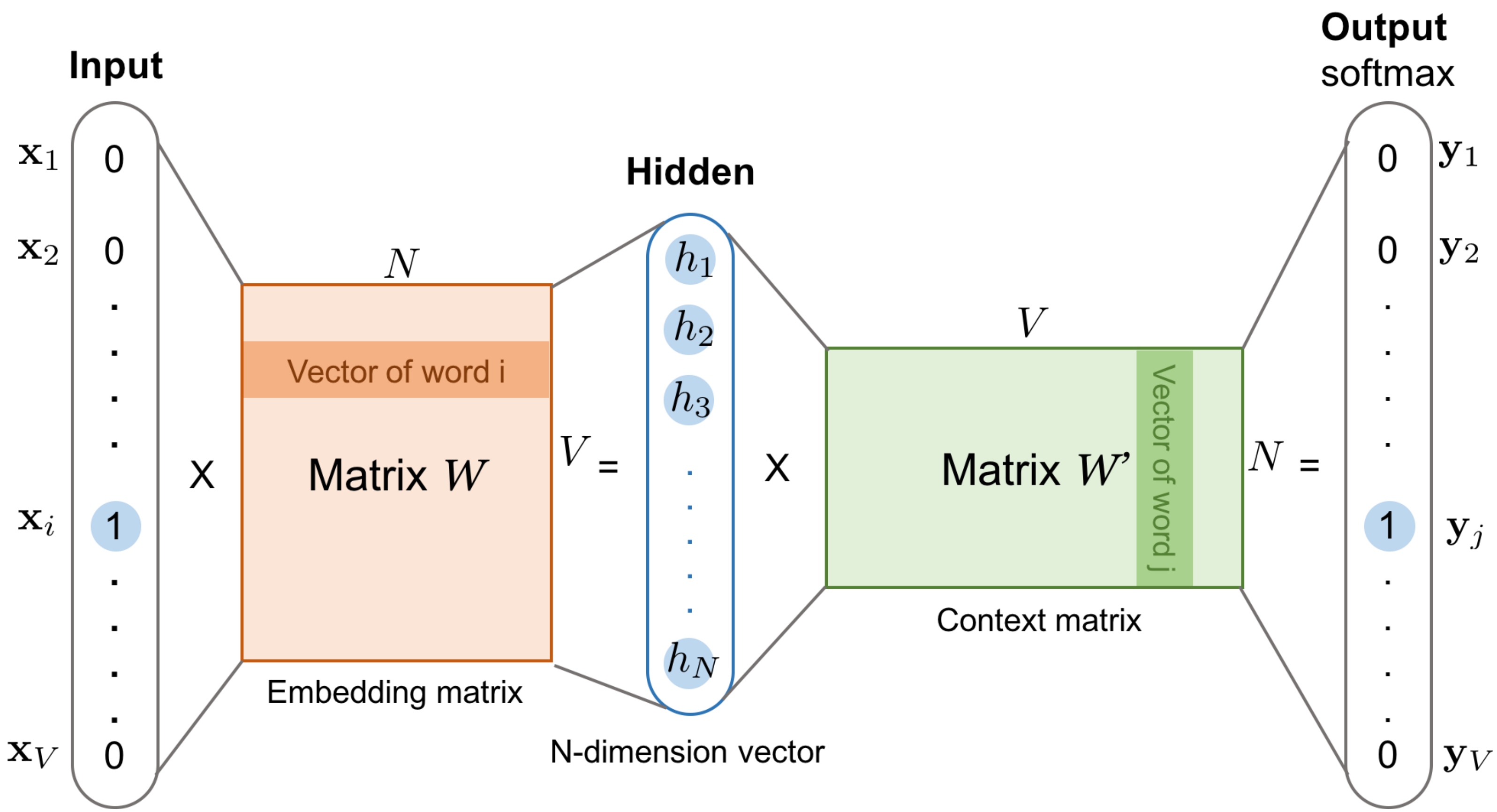
Sam	likes	Celine	Dion	and	biking

Sam	likes
Sam	Celine
likes	Sam
likes	Celine
likes	Dion

Sam	likes	Celine	Dion	and	biking

Sam	likes
Sam	Celine
likes	Sam
likes	Celine
likes	Dion
Celine	Sam
Celine	likes
Celine	Dion
Celine	and

Skip gram

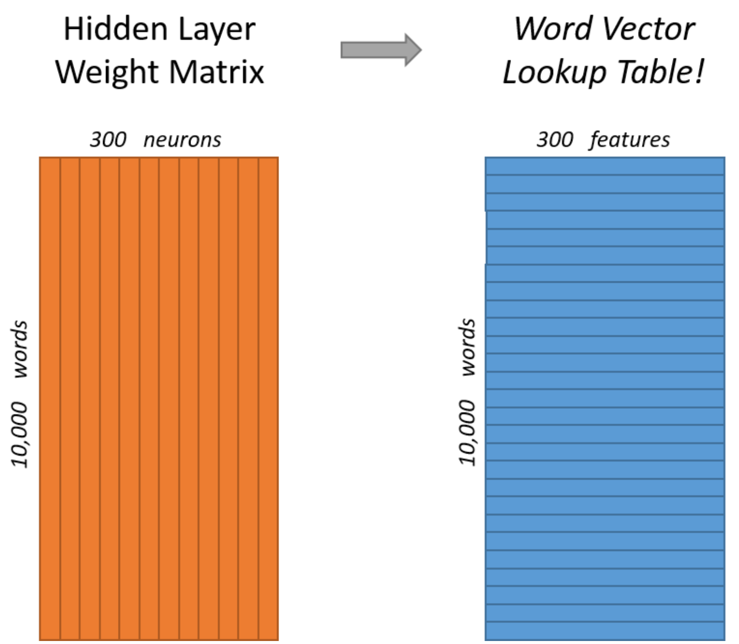


Input
Output
softmax
Hidden
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_i$
$\mathbf{x}_V$
0
0
1
0
X
N
Vector of word i
Matrix W
Embedding matrix
V =
h_1
h_2
h_3
h_N
N-dimension vector
X
V
Matrix W'
Vector of word j
Context matrix
N =
0
0
1
0
$\mathbf{y}_1$
$\mathbf{y}_2$
$\mathbf{y}_j$
$\mathbf{y}_V$

Skip gram example

- Vocabulary of 10,000 words.
- Embedding vectors with 300 features.
- So the hidden layer is going to be represented by a weight matrix of size 300 with 10,000 rows

$$[0 \quad 0 \quad 0 \quad \boxed{1} \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$



http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word2vec shortcomings

- **Problem:** 10,000 words and 300 dim embedding gives a large parameter space to learn. And 10K words is minimal for real applications.
- Slow to train, and need lots of data, particularly to learn uncommon words.

Any ideas how to make the approach more scalable?

Word2vec improvements: word pairs and phrases

- **Idea:** Treat common word pairs or phrases as single "words."
  - E.g., Boston Globe (newspaper) is different from Boston and Globe separately. Embed Boston Globe as a single word/phrase.

- **Method:** make phrases out of words which occur together often relative to the number of individual occurrences. Prefer phrases made of infrequent words in order to avoid making phrases out of common words like "and the" or "this is".

- **Pros/cons:** Increases vocabulary size but decreases training expense.

Word2vec improvements: subsample frequent words

- **Idea:** Subsample frequent words to decrease the number of training examples.
  - The probability that we cut the word is related to the word's frequency. More common words are cut more.
  - Uncommon words (anything < 0.26% of total words) are kept
  - E.g., remove some occurrences of "the."

- **Method:** For each word, cut the word with probability related to the word's frequency.

- **Benefits:** If we have a window size of 10, and we remove a specific instance of "the" from our text:
  - As we train on the remaining words, "the" will not appear in any of their context windows.

Word2vec improvements: selective updates

- **Idea:** Use “Negative Sampling”, which causes each training sample to update only a small percentage of the model’s weights.

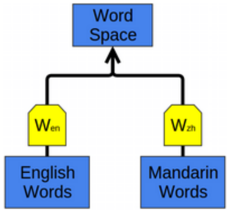
- **Observation:** A “correct output” of the network is a one-hot vector. That is, one neuron should output a 1, and *all* of the other thousands of output neurons to output a 0.

- **Method:** With negative sampling, randomly select just a small number of “negative” words (let’s say 5) to update the weights for. (In this context, a “negative” word is one for which we want the network to output a 0 for). We will also still update the weights for our “positive” word.

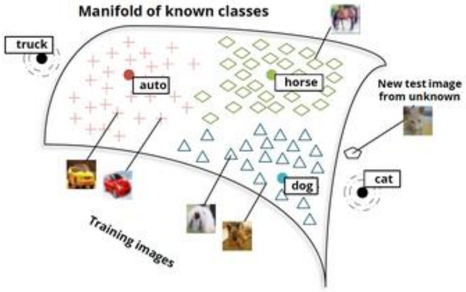
Applications

- Clustering
- Next word prediction

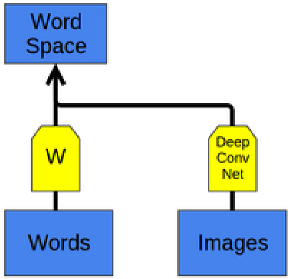
- Translation



- Other: Images ….
  - Can apply to get a joint embedding of words and images or other multi-modal data sets.
  - New classes map near similar existing classes: e.g., if 'cat' is unknown, cat images map near dog.



(Socher *et al.* (2013b))



WHAT IS THE PROBLEM WITH WORD EMBEDDINGS?

The mountain has a lot of **grass**

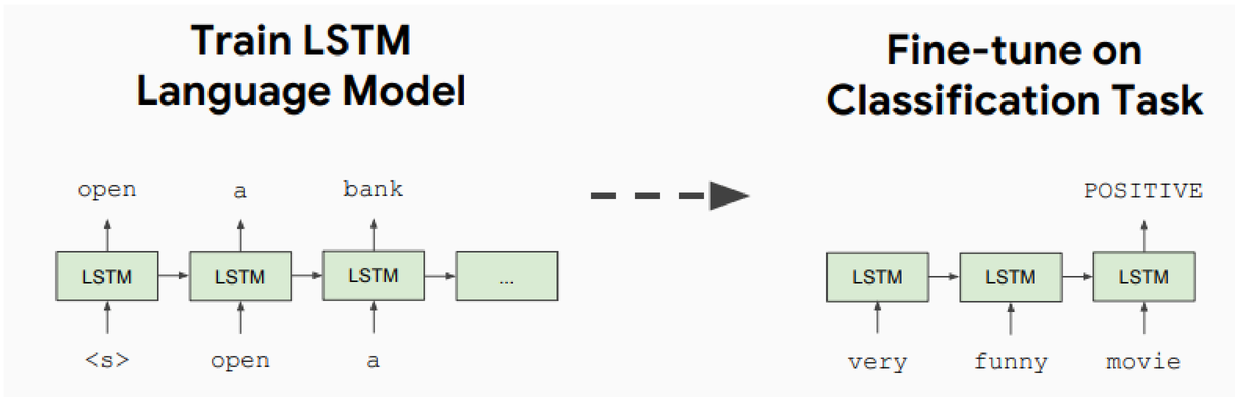
You should never smoke **grass**

same word embedding [0.99, 0.8, …]

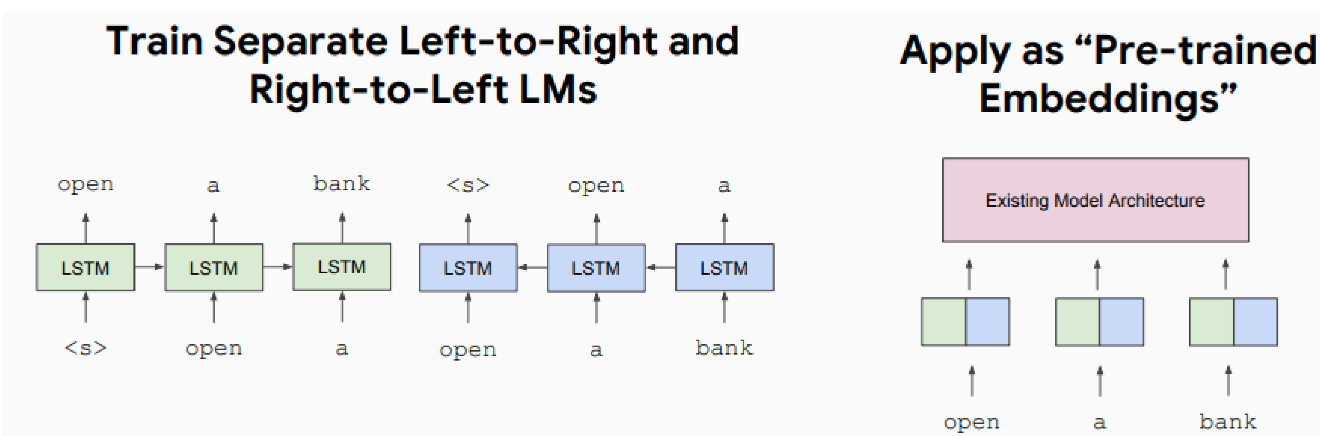
Solution: Train contextual representations on text corpus

LITTLE HISTORY

Semi-Supervised Sequence Learning, Google, 2015

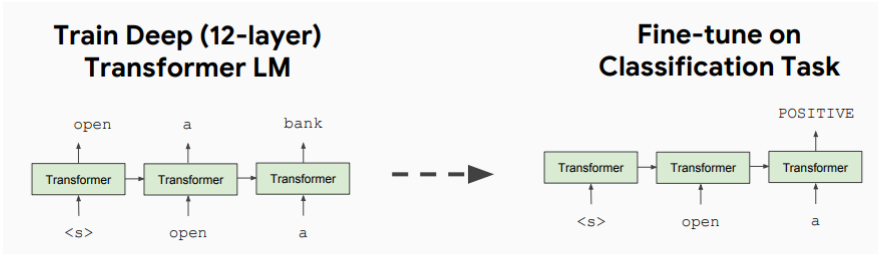


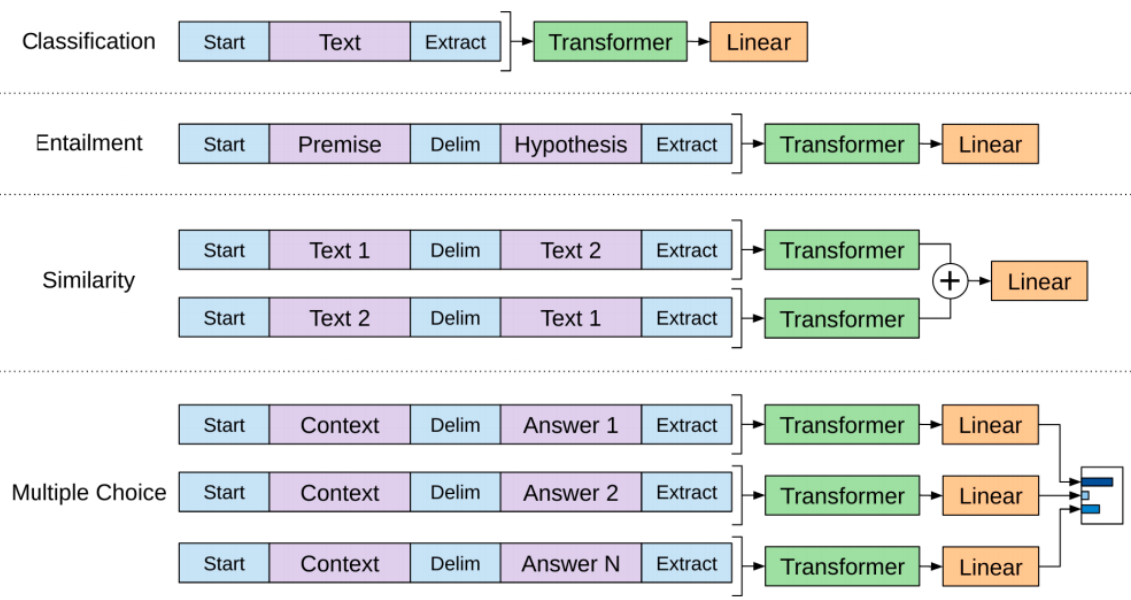
ELMo: Deep Contextual Word Embeddings, AI2 & University of Washington, 2017



GPT

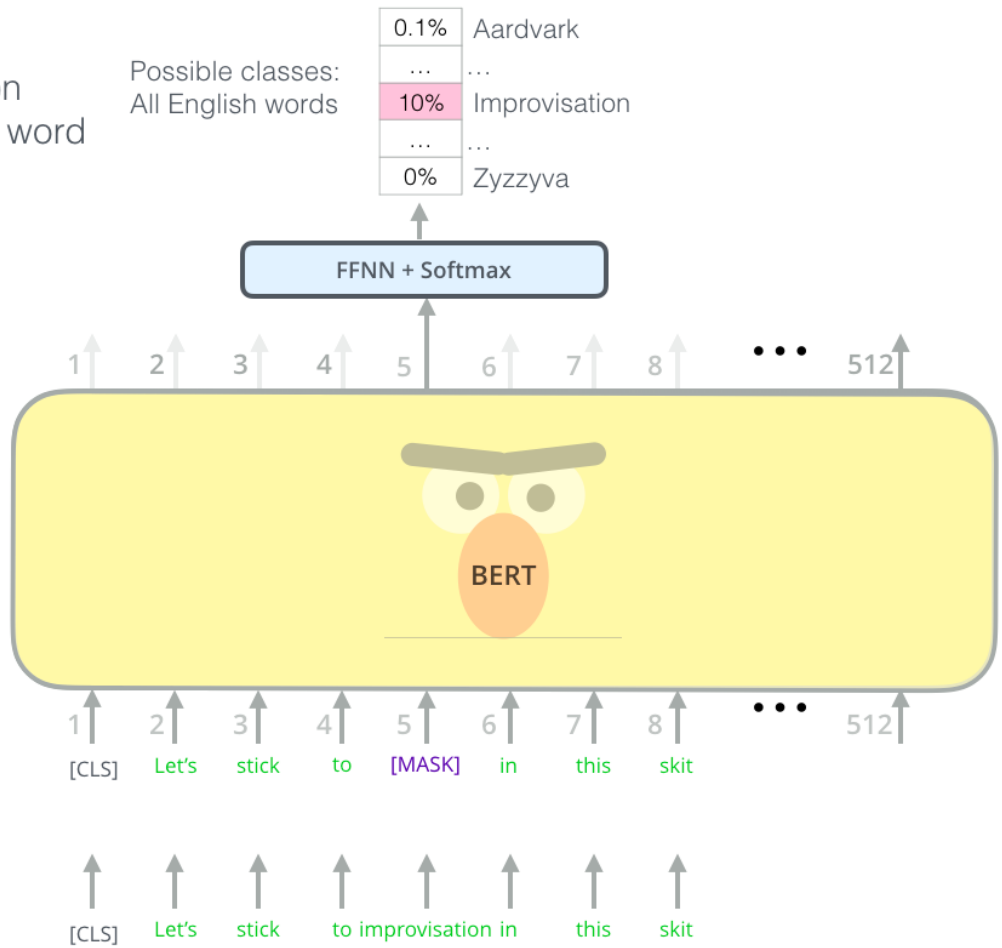
Improving Language Understanding by Generative Pre-Training, OpenAI, 2018 – Based on transformers/attention from "Attention is All You Need" Vaswani et al





BERT

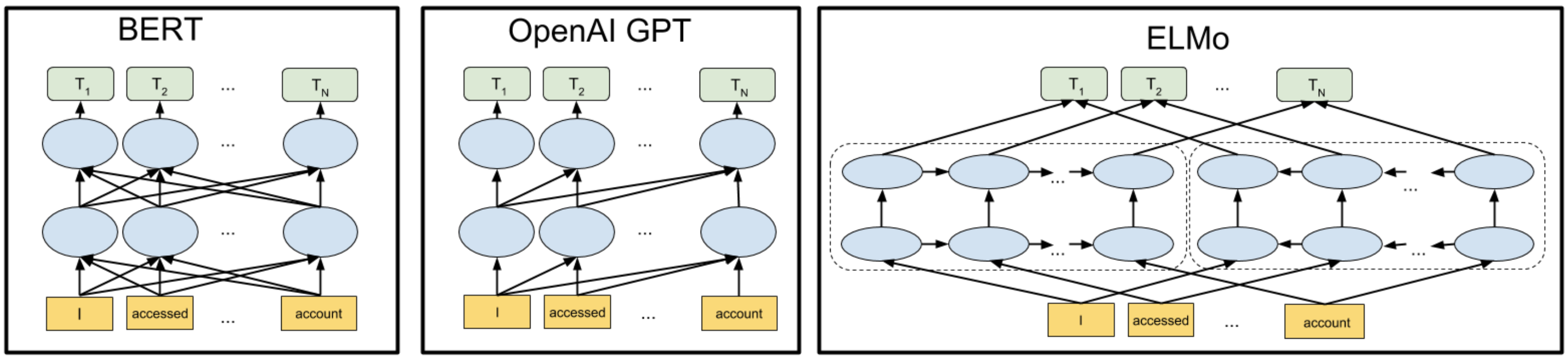
Use the output of the masked word's position to predict the masked word



Randomly mask 15% of tokens

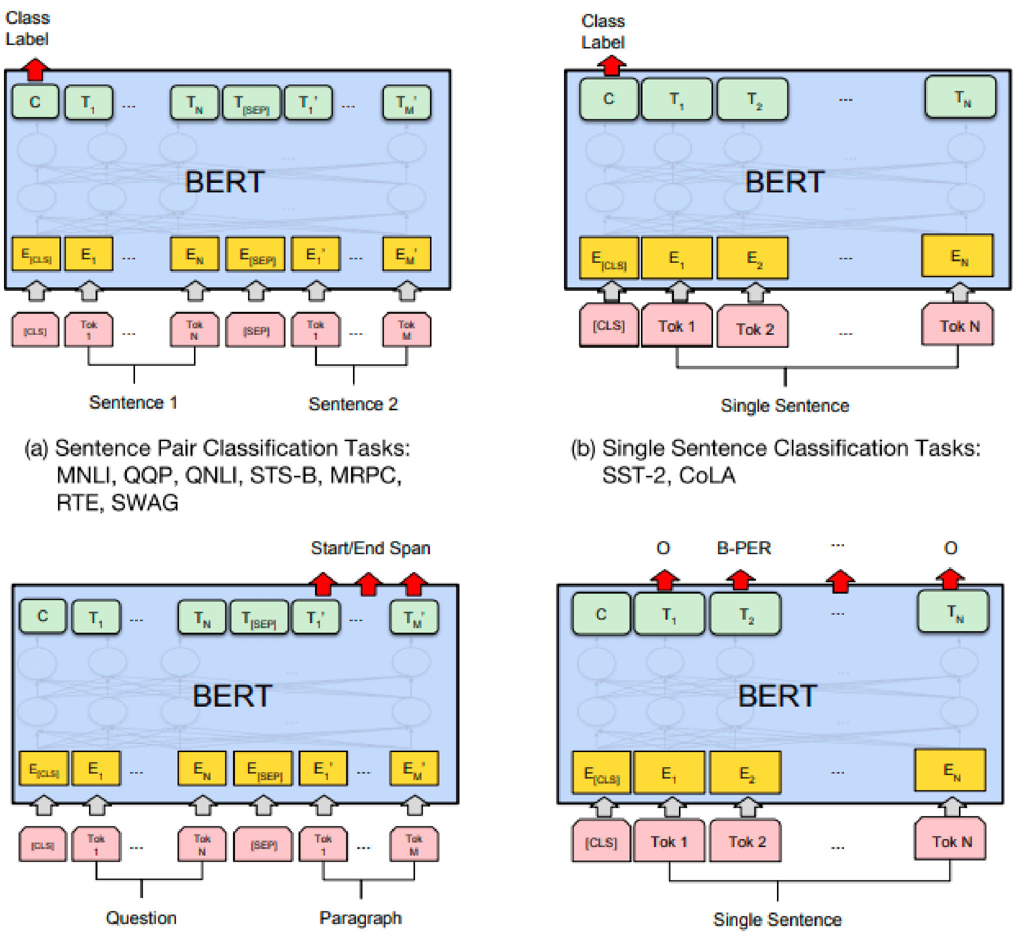
Input

BERT VS OPENAI GPT VS ELMO



[See also http://jalammar.github.io/illustrated-gpt2/](http://jalammar.github.io/illustrated-gpt2/)

TASKS



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

http://www.msmarco.org/leaders.aspx

MICROSOFT MACHINE READING COMPREHENSION DATASET (MS MARCO)

KeyPhrase Extraction(10/18/2019) ranked by F1 @3 on Eval

Rank	Model	Submission Date	Precision @1,@3,@5	Recall @1,@3,@5	F1 @1,@3,@5
1	**BERT (Base) Sequence Tagging** Si Sun (Tsinghua University), Chenyan Xiong (MSR AI), Zhiyuan Liu (Tsinghua University) [Code]	November 5th, 2019	0.484, 0.312, 0.227	0.255, 0.469, 0.563	0.321, **0.361**, 0.314
2	**Baseline finetuned on Bing Queries** MSMARCO Team	October 19th, 2019	0.397, 0.249, 0.149	0.215, 0.391, 0.391	0.267, **0.292**, 0.209
3	**Baseline** MSMARCO Team	October 19th, 2019	0.365, 0.237, 0.142	0.196, 0.367, 0.367	0.244, **0.277**, 0.198

Passage Retrieval(10/26/2018-Present) ranked by MRR on Eval

Rank	Model	Ranking Style	Submission Date	MRR@10 On Eval	MRR@10 On Dev
1	**Enriched BERT base + AOA index + CAS** Ming Yan of Alibaba Damo NLP	Full Ranking	August 20th, 2019	0.393	0.408
2	**W-Index retrieval + BERT-F re-rank** Zhuyun Dai of Carnegie Mellon University	Full Ranking	September 12th,2019	0.388	0.394
3	**Enriched BERT base + AOA index V1** Ming Yan of Alibaba Damo NLP	Full Ranking	May 13th, 2019	0.383	0.397

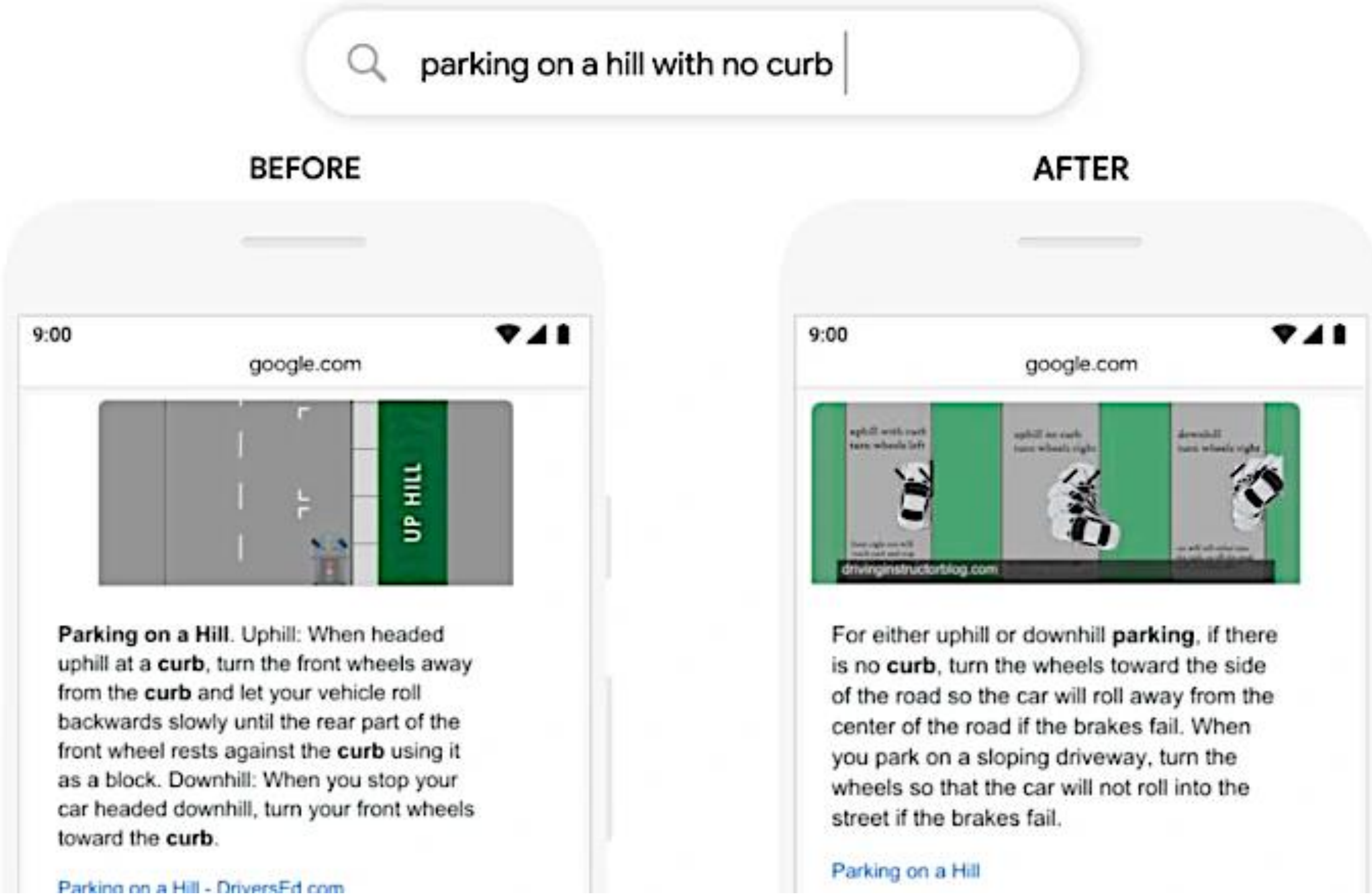
Q&A Task(03/01/2018-Present)

Rank	Model	Submission Date	Rouge-L	Bleu-1
1	**Multi-doc Enriched BERT** Ming Yan of Alibaba Damo NLP	June 20th, 2019	0.540	0.565
2	**Human Performance**	April 23th, 2018	0.539	0.485
3	**BERT Encoded T-Net** Y. Zhang, C. Wang, X.L. Chen	August 5th, 2019	0.526	0.539

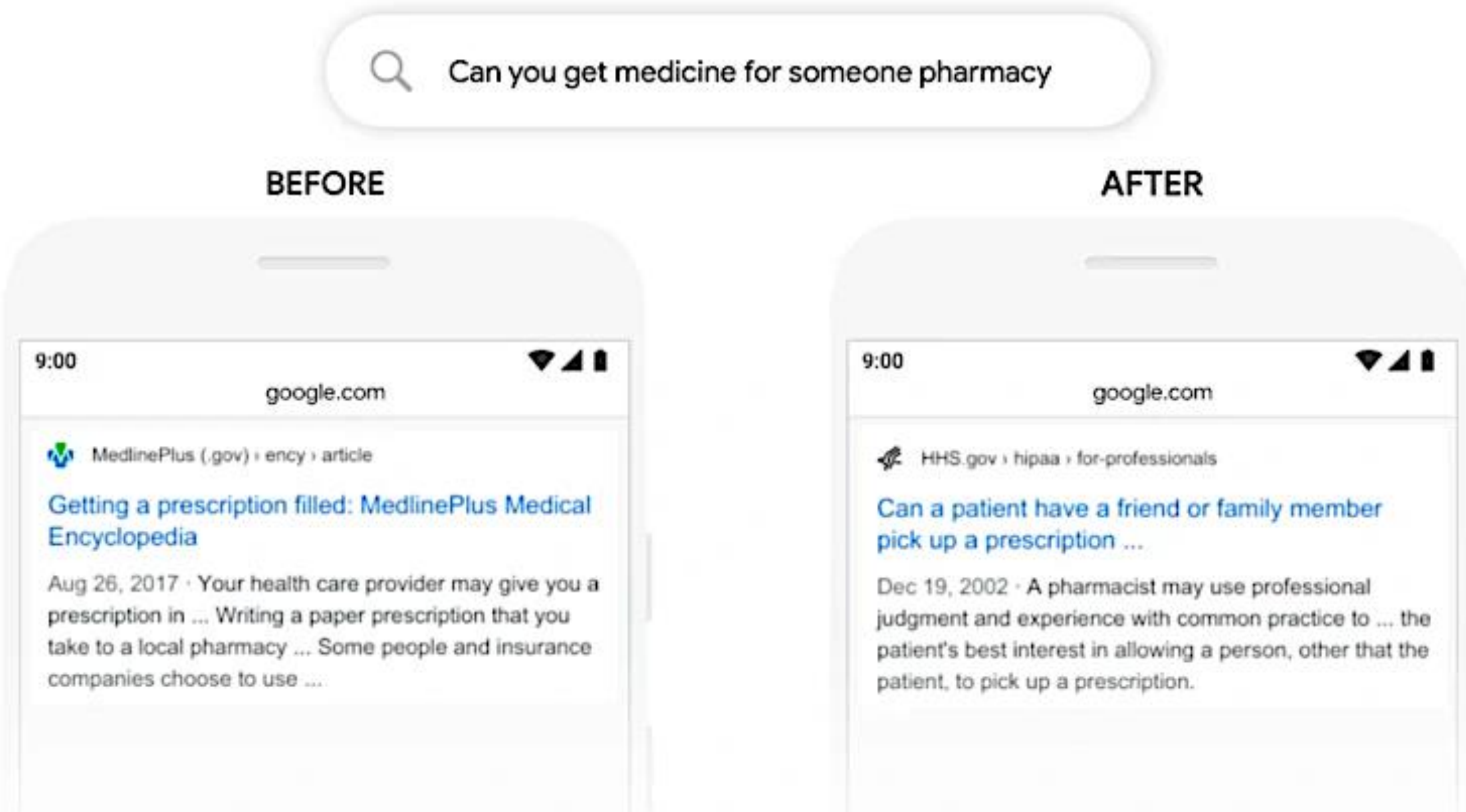
Q&A + Natural Language Generation Task(03/01/2018-Present)

Rank	Model	Submission Date	Rouge-L	Bleu-1
1	**Human Performance**	April 23th, 2018	0.632	0.530
2	**Masque NLGEN Style** NTT Media Intelligence Laboratories [Nishida et al. '19]	January 3rd, 2019	0.496	0.501
3	**BERT+ Multi-Pointer-Generator** Tongjun Li of the ColorfulClouds Tech and BUPT	June 11th,2019	0.495	0.476

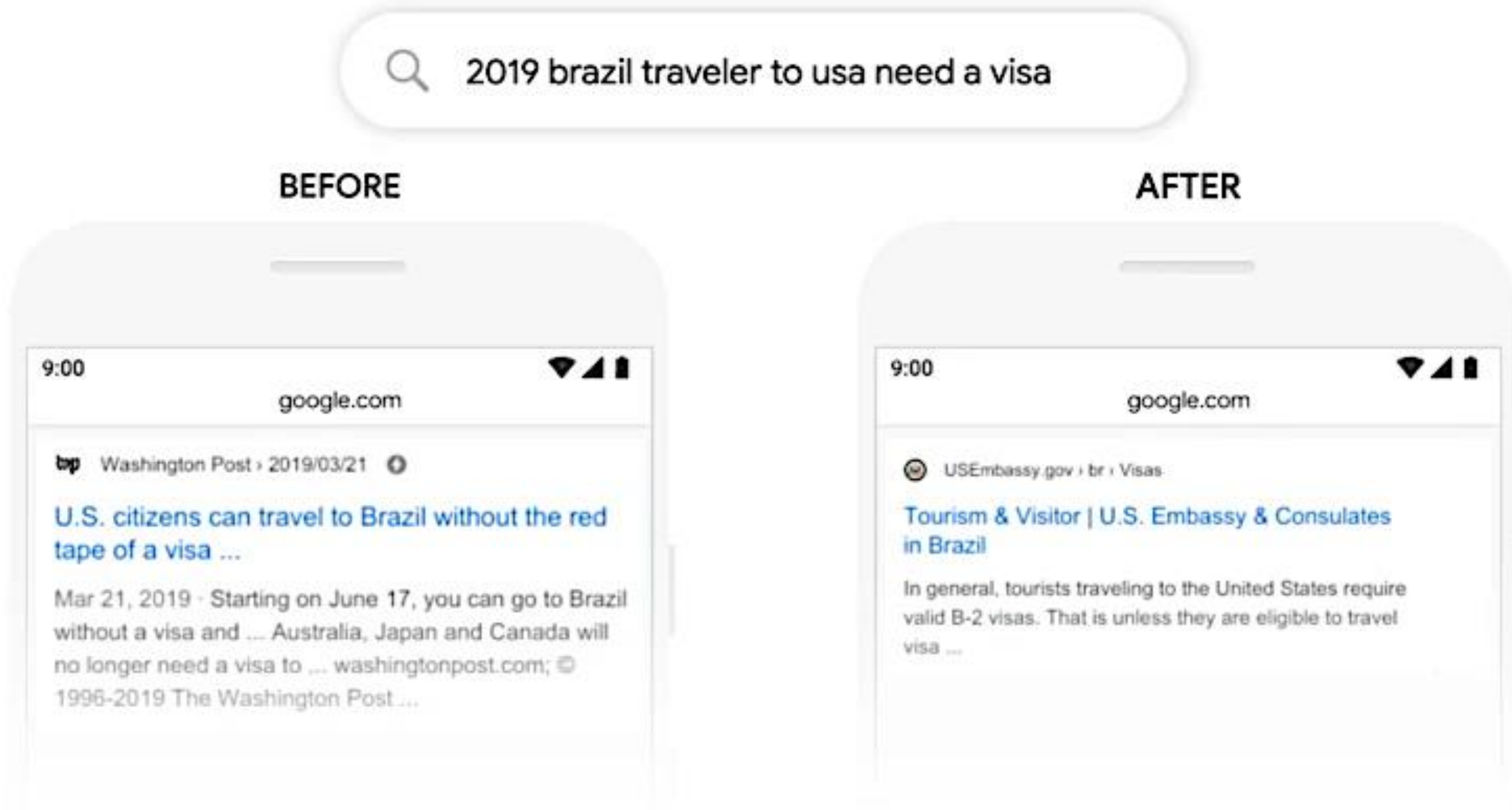
GOOGLE IS NOW USING BERT



GOOGLE IS NOW USING BERT

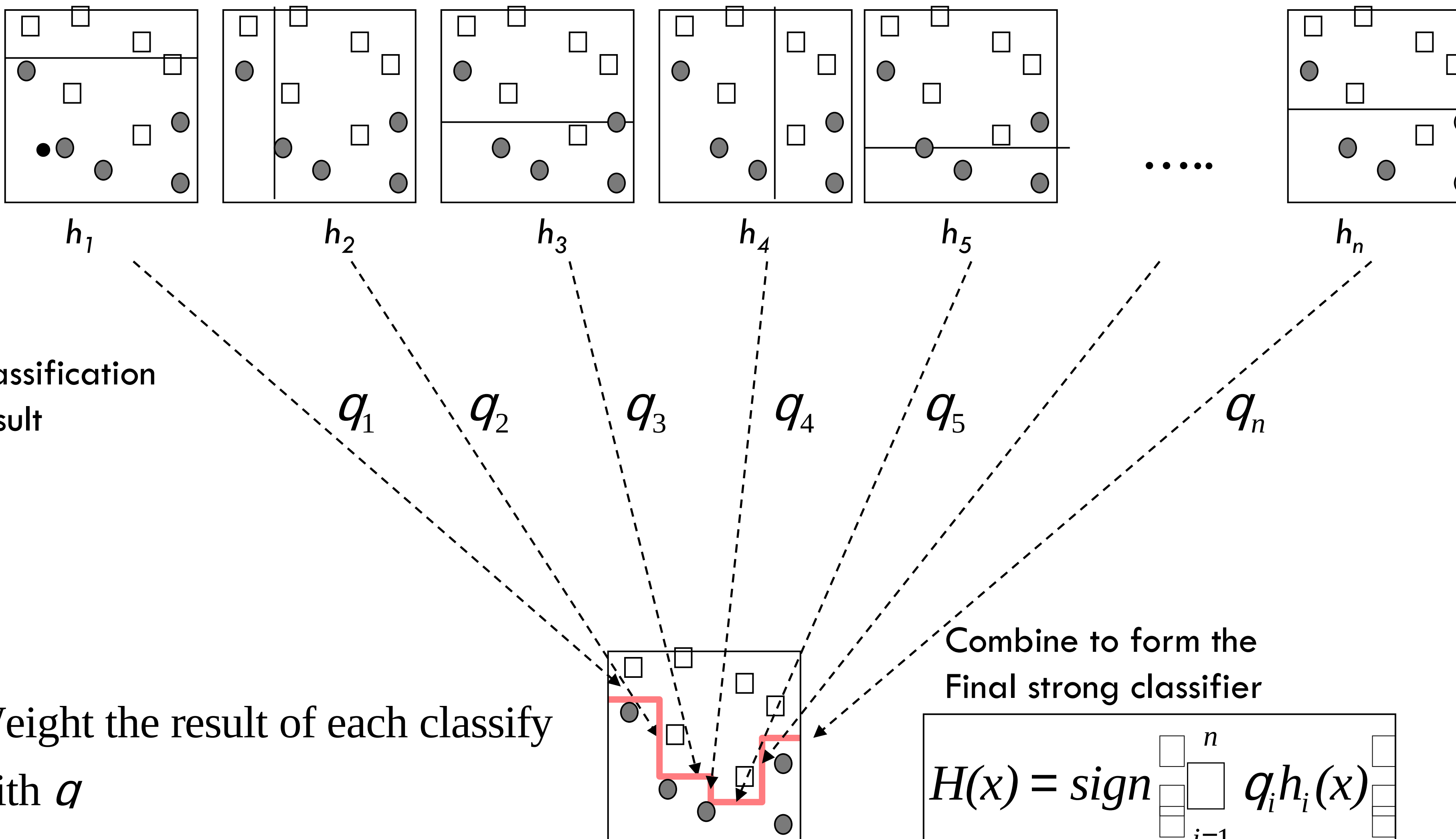


GOOGLE IS NOW USING BERT



ADABOOST - CORE IDEA

Take a set of weak classifiers (normally they should do better than guessing)



Classification Result

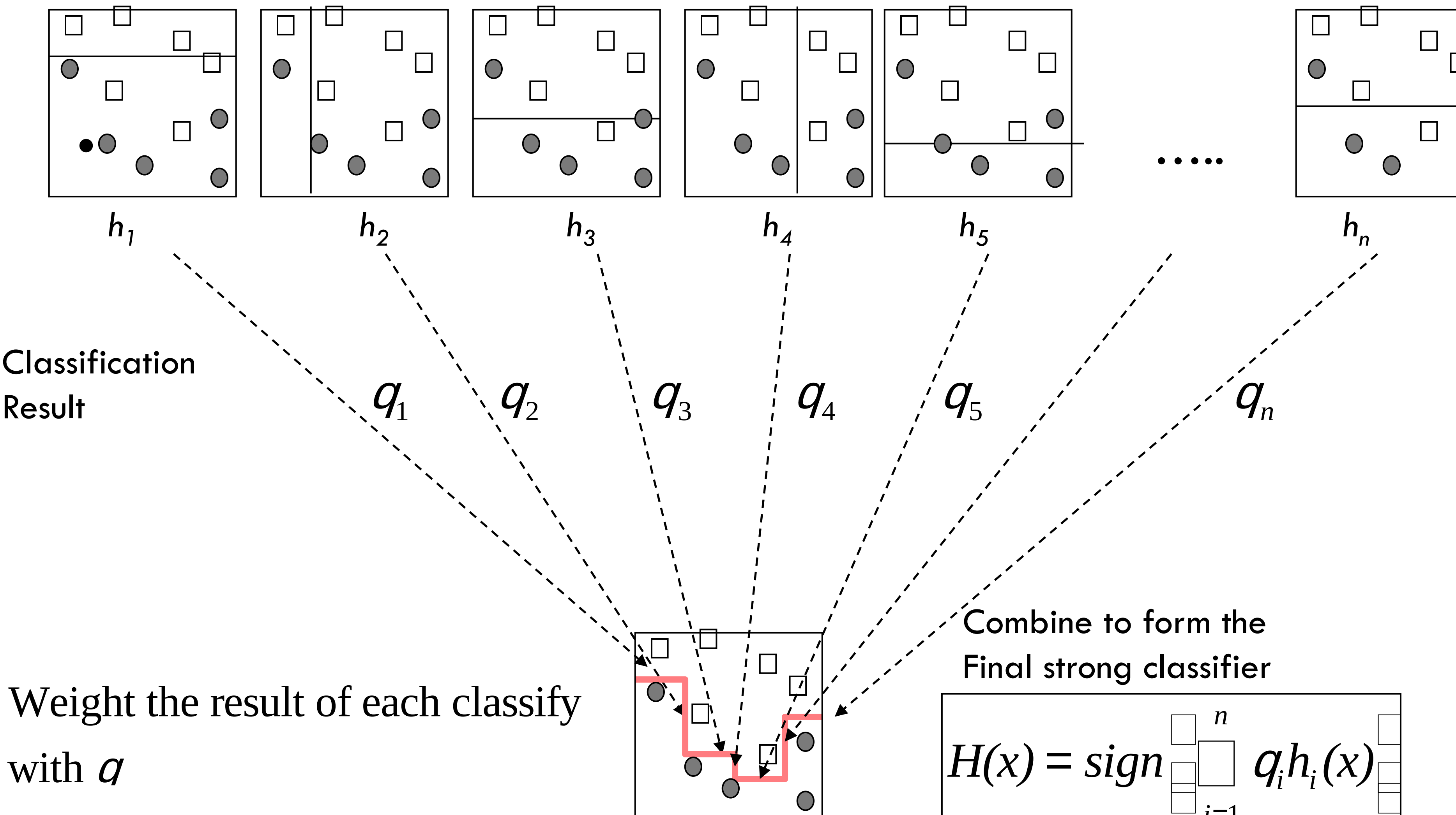
Weight the result of each classify with q

Combine to form the Final strong classifier

$$H(x) = sign\left(\sum_{i=1}^{n} q_i h_i(x)\right)$$

ADABOOST - CORE IDEA

Take a set of weak classifiers (normally they should do better than guessing)

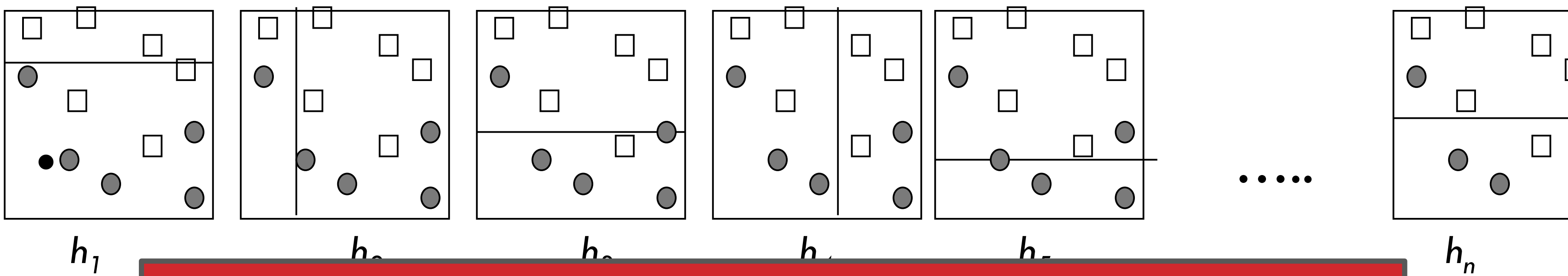


Weight the result of each classify with q

$$H(x) = sign\left(\sum_{i=1}^{n} q_i h_i(x)\right)$$

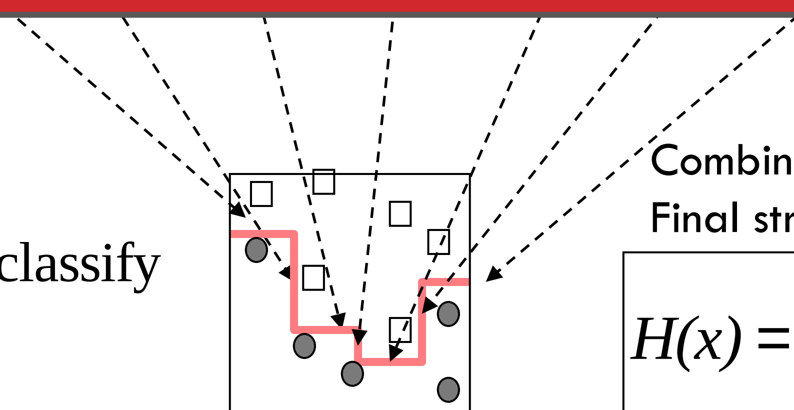
ADABOOST - CORE IDEA

Take a set of weak classifiers (normally they should do better than guessing)



Classificatio
Result

XGBoost follows the same idea



Combine to form the Final strong classifier

$$H(x) = sign\left(\sum_{i=1}^{n} q_i h_i(x)\right)$$

Weight the result of each classify with q

IDEA BEHIND GRADIENT BOOSTING

Desc	Protein	Carb	Sugar	Iron	Kcal
CHEESE,GRUYERE	30	0	0.36	0.17	413
ICE CRM SNDWCH	4	37	18.57	0.26	237
PORK,LOIN	21	0	0	0.84	143
CARROTS,RAW	1	10	4.74	0.3	41
APPLES,RAW,WITH SKIN	0	14	10.39	0.12	52
BEEF,STRIP STEAKS	23	0	0	1.85	117

Step 1: Build baseline model

https://data.nal.usda.gov/dataset/composition-foods-raw-processed-prepared-usda-national-nutrient-database-standard-reference-release-28-0

IDEA BEHIND GRADIENT BOOSTING

Desc	Protein	Carb	Sugar	Iron	Kcal
CHEESE,GRUYERE	30	0	0.36	0.17	413
ICE CRM SNDWCH	4	37	18.57	0.26	237
PORK,LOIN	21	0	0	0.84	143
CARROTS,RAW	1	10	4.74	0.3	41
APPLES,RAW,WITH SKIN	0	14	10.39	0.12	52
BEEF,STRIP STEAKS	23	0	0	1.85	117

Step 1: Build baseline model

1a) Naïve baseline: average (413+237+…+117)/6 = 167

https://data.nal.usda.gov/dataset/composition-foods-raw-processed-prepared-usda-national-nutrient-database-standard-reference-release-28-0

IDEA BEHIND GRADIENT BOOSTING

Desc	Protein	Carb	Sugar	Iron	Kcal	Residual
CHEESE,GRUYERE	30	0	0.36	0.17	413	246
ICE CRM SNDWCH	4	37	18.57	0.26	237	70
PORK,LOIN	21	0	0	0.84	143	-24
CARROTS,RAW	1	10	4.74	0.3	41	-126
APPLES,RAW,WITH SKIN	0	14	10.39	0.12	52	-115
BEEF,STRIP STEAKS	23	0	0	1.85	117	-50

Step 1: **Build baseline model**

1a) Naïve baseline: average (413+237+…+117)/6 = 167

1b) Calculate **residuals**: actual value – predicted value

For example, for Brie: 413 - 167 = 246

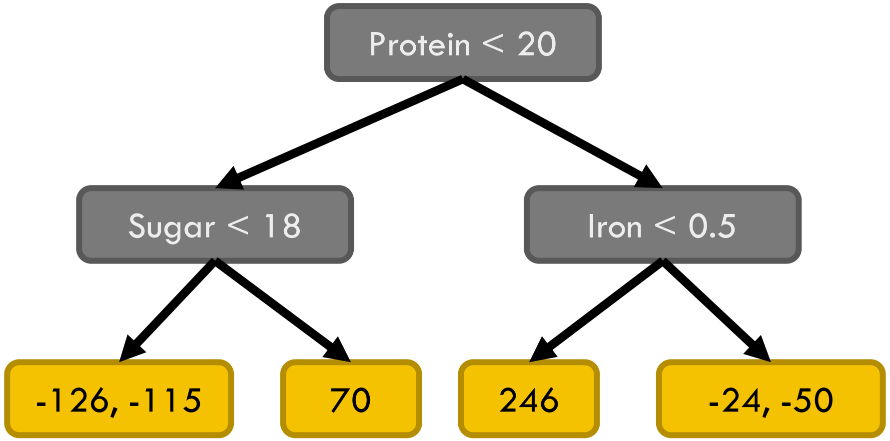
https://data.nal.usda.gov/dataset/composition-foods-raw-processed-prepared-usda-national-nutrient-database-standard-reference-release-28-0

IDEA BEHIND GRADIENT BOOSTING

Desc	Protein	Carb	Sugar	Iron	Kcal	Residual
CHEESE,GRUYERE	30	0	0.36	0.17	413	246
ICE CRM SNDWCH	4	37	18.57	0.26	237	70
PORK,LOIN	21	0	0	0.84	143	-24
CARROTS,RAW	1	10	4.74	0.3	41	-126
APPLES,RAW,WITH SKIN	0	14	10.39	0.12	52	-115
BEEF,STRIP STEAKS	23	0	0	1.85	117	-50

Step 1: Build baseline model

Step 2: Build tree over residuals (not labels)



https://data.nal.usda.gov/dataset/composition-foods-raw-processed-prepared-usda-national-nutrient-database-standard-reference-release-28-0

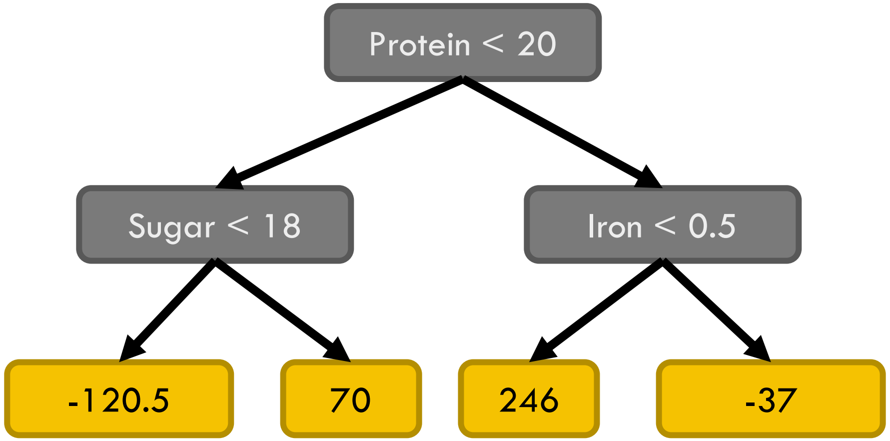
IDEA BEHIND GRADIENT BOOSTING

Desc	Protein	Carb	Sugar	Iron	Kcal	Residual
CHEESE,GRUYERE	30	0	0.36	0.17	413	246
ICE CRM SNDWCH	4	37	18.57	0.26	237	70
PORK,LOIN	21	0	0	0.84	143	-24
CARROTS,RAW	1	10	4.74	0.3	41	-126
APPLES,RAW,WITH SKIN	0	14	10.39	0.12	52	-115
BEEF,STRIP STEAKS	23	0	0	1.85	117	-50

Step 1: **Build baseline model**

Step 2: **Build tree over residuals (not labels)**

Compute average for residuals in the same leaf



https://data.nal.usda.gov/dataset/composition-foods-raw-processed-prepared-usda-national-nutrient-database-standard-reference-release-28-0

IDEA BEHIND GRADIENT BOOSTING

Desc	Protein	Carb	Sugar	Iron	Kcal	Residual	Predictions
CHEESE,GRUYERE	30	0	0.36	0.17	413	246	413
ICE CRM SNDWCH	4	37	18.57	0.26	237	70	237
PORK,LOIN	21	0	0	0.84	143	-24	130
CARROTS,RAW	1	10	4.74	0.3	41	-126	47
APPLES,RAW,WITH SKIN	0	14	10.39	0.12	52	-115	47
BEEF,STRIP STEAKS	23	0	0	1.85	117	-50	130

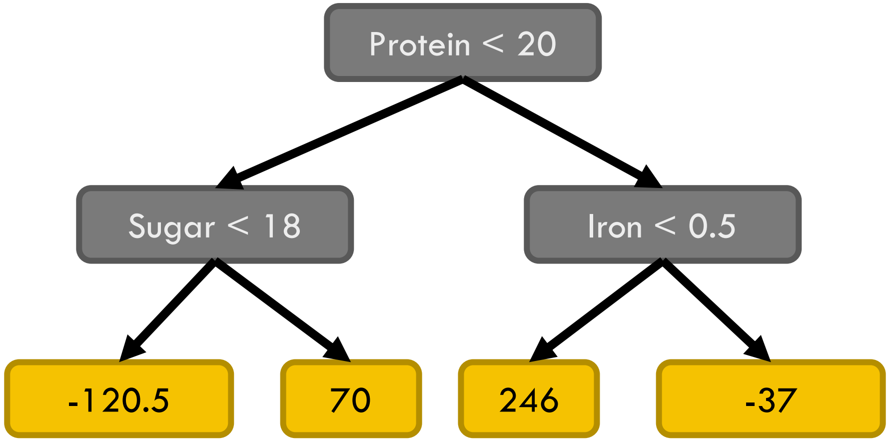
Step 1: Build baseline model

Step 2: Build tree over residuals (not labels)

Step 3': Predict the target labels using all trees (currently just one)

Prediction: avg + residual predicted by decision tree

For example for Gruyere: 167 + 246 = 413

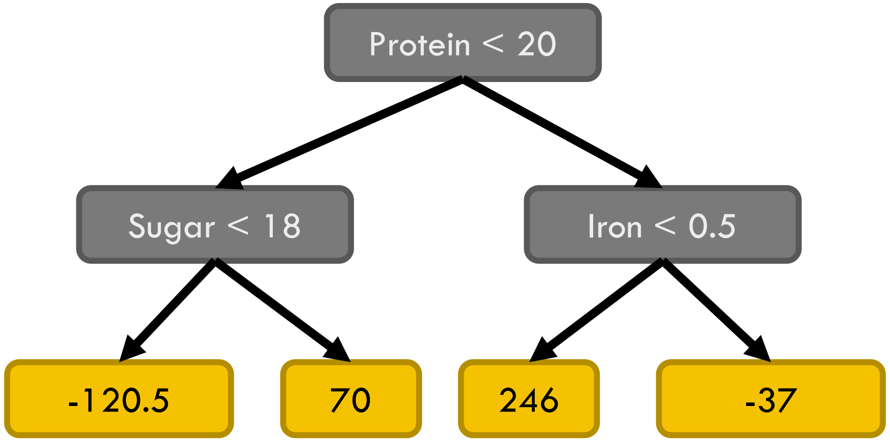


https://data.nal.usda.gov/dataset/composition-foods-raw-processed-prepared-usda-national-nutrient-database-standard-reference-release-28-0

IDEA BEHIND GRADIENT BOOSTING

Desc	Protein	Carb	Sugar	Iron	Kcal	Residual	Predictions	New Residuals
CHEESE,GRUYERE	30	0	0.36	0.17	413	246	413	0
ICE CRM SNDWCH	4	37	18.57	0.26	237	70	237	0
PORK,LOIN	21	0	0	0.84	143	-24	130	13
CARROTS,RAW	1	10	4.74	0.3	41	-126	47	-6
APPLES,RAW,WITH SKIN	0	14	10.39	0.12	52	-115	47	6
BEEF,STRIP STEAKS	23	0	0	1.85	117	-50	130	-13

Step 1: Build baseline model
Step 2: Build tree over residuals (not labels)
Step 3': Predict the target labels using all trees (currently just one)
Step 4': Compute the new residuals



https://data.nal.usda.gov/dataset/composition-foods-raw-processed-prepared-usda-national-nutrient-database-standard-reference-release-28-0

IDEA BEHIND GRADIENT BOOSTING

Desc	Protein	Carb	Sugar	Iron	Kcal	Residual	Predictions
CHEESE,GRUYERE	30	0	0.36	0.17	413	246	192
ICE CRM SNDWCH	4	37	18.57	0.26	237	70	174
PORK,LOIN	21	0	0	0.84	143	-24	163
CARROTS,RAW	1	10	4.74	0.3	41	-126	155
APPLES,RAW,WITH SKIN	0	14	10.39	0.12	52	-115	155
BEEF,STRIP STEAKS	23	0	0	1.85	117	-50	163

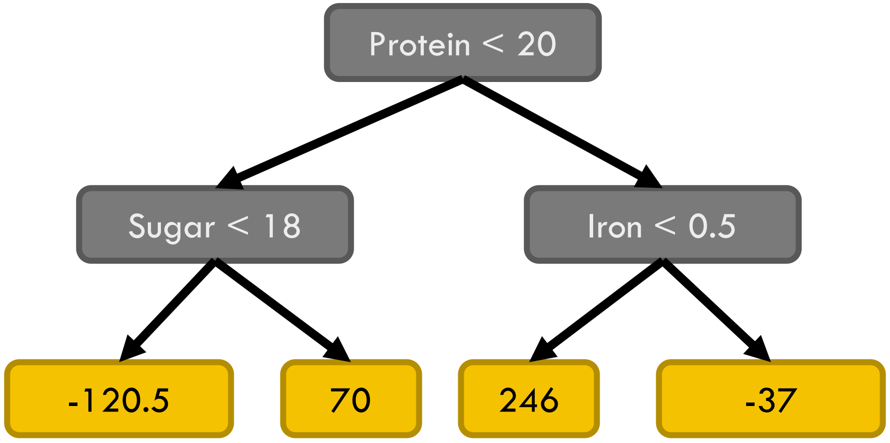
Step 1: Build baseline model

Step 2: Build tree over residuals (not labels)

Step 3: Predict the target labels using all trees with Learning Rate

Prediction: avg + **LR** * residual predicted by decision tree

For example for Gruyere: 167 + 0.1 * 246 = 191.6



https://data.nal.usda.gov/dataset/composition-foods-raw-processed-prepared-usda-national-nutrient-database-standard-reference-release-28-0

IDEA BEHIND GRADIENT BOOSTING

Desc	Protein	Carb	Sugar	Iron	Kcal	Residual1	Predictions1	Residual2
CHEESE,GRUYERE	30	0	0.36	0.17	413	246	192	221
ICE CRM SNDWCH	4	37	18.57	0.26	237	70	174	63
PORK,LOIN	21	0	0	0.84	143	-24	163	-20
CARROTS,RAW	1	10	4.74	0.3	41	-126	155	-114
APPLES,RAW,WITH SKIN	0	14	10.39	0.12	52	-115	155	-103
BEEF,STRIP STEAKS	23	0	0	1.85	117	-50	163	-46

Step 1: Build baseline model

Step 2: Build tree over residuals (not labels)

Step 3: Predict the target labels using all trees

Step 4: Compute the new residuals

Step 5: Repeat steps 3 to 5 until breaking criteria (test/validation error, iterations, etc.)

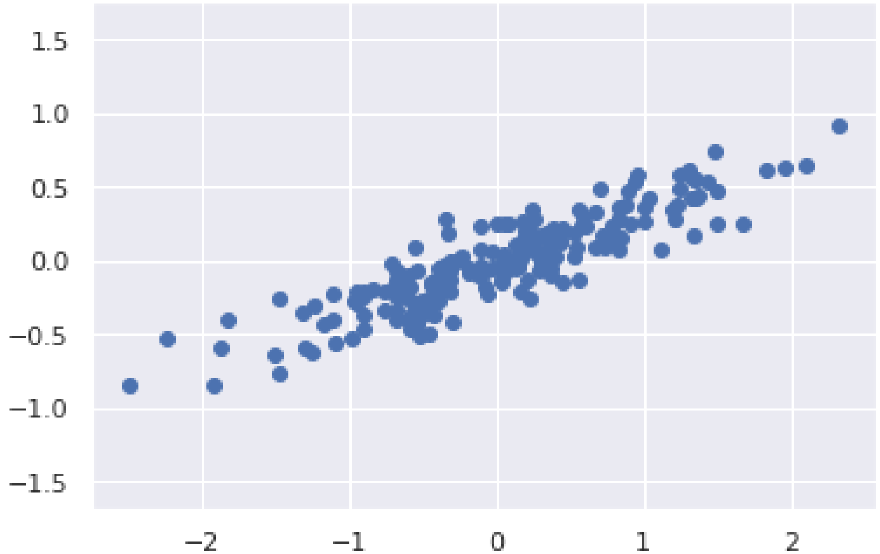
Once trained, use all of the trees in the ensemble to make a final prediction as to the value of the target variable

AVG + LR * Predicted Residual1 + LR * Predicted Residual2 + …

https://data.nal.usda.gov/dataset/composition-foods-raw-processed-prepared-usda-national-nutrient-database-standard-reference-release-28-0

PRINCIPAL COMPONENT ANALYSIS (PCA)

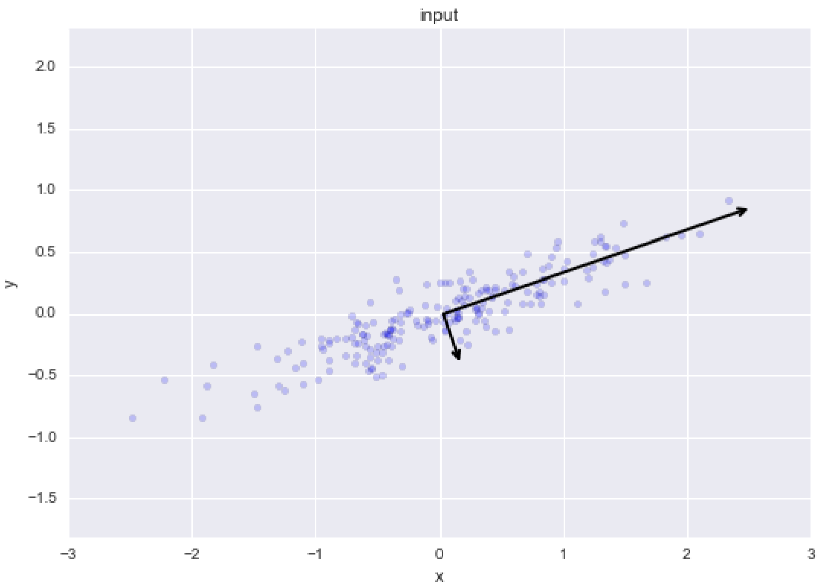
Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components



https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.09-Principal-Component-Analysis.ipynb#scrollTo=V7ssdJKEi9O1

PRINCIPAL COMPONENT ANALYSIS (PCA)

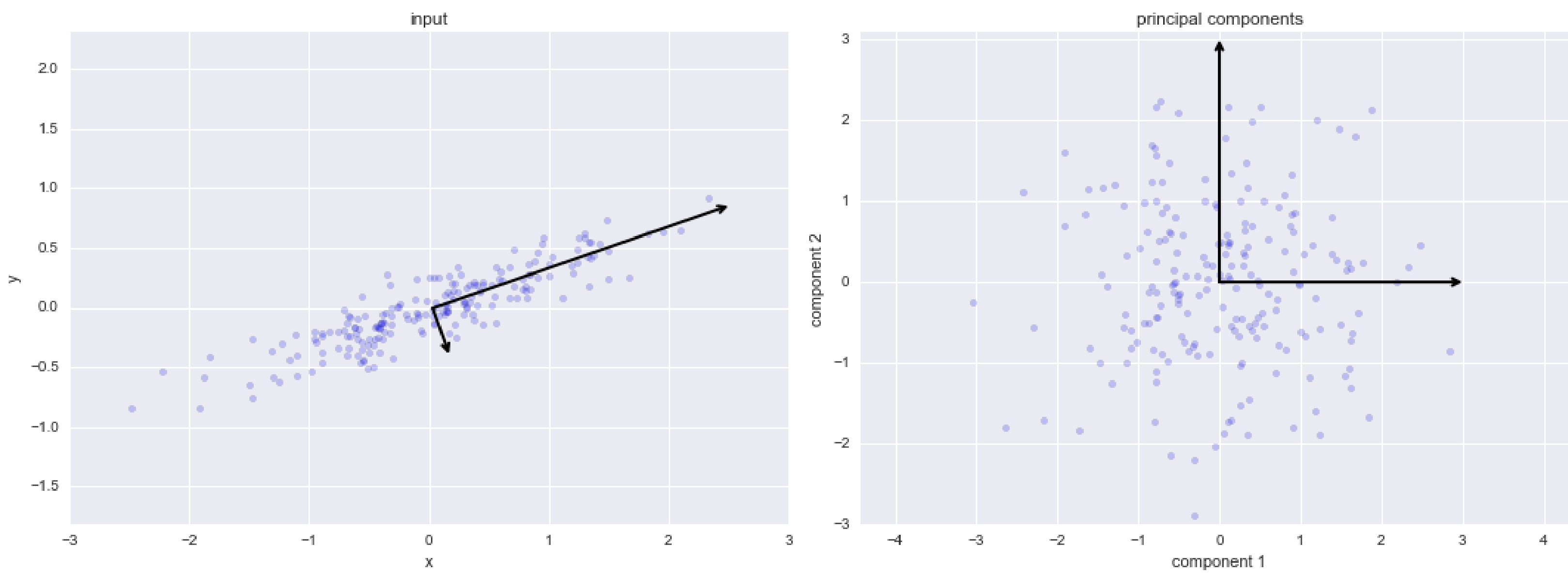
Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components



https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.09-Principal-Component-Analysis.ipynb#scrollTo=V7ssdJKEi9O1

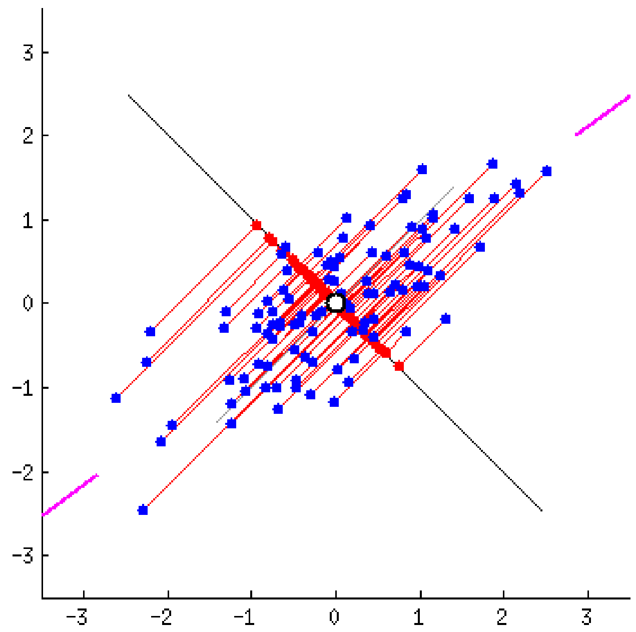
PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components

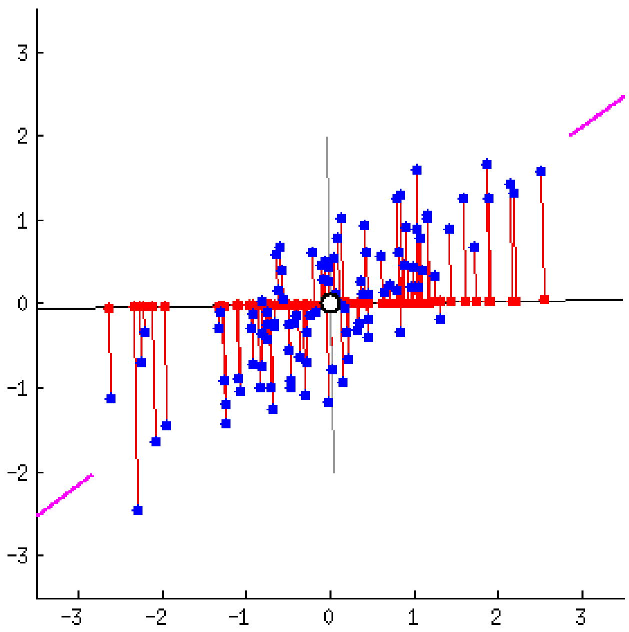


https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.09-Principal-Component-Analysis.ipynb#scrollTo=V7ssdJKEi9O1

FIRST PRINCIPLE COMPONENT



CLICKER: CAN WE USE LR TO FIND THE FIRST PRINCIPLE COMPONENT?



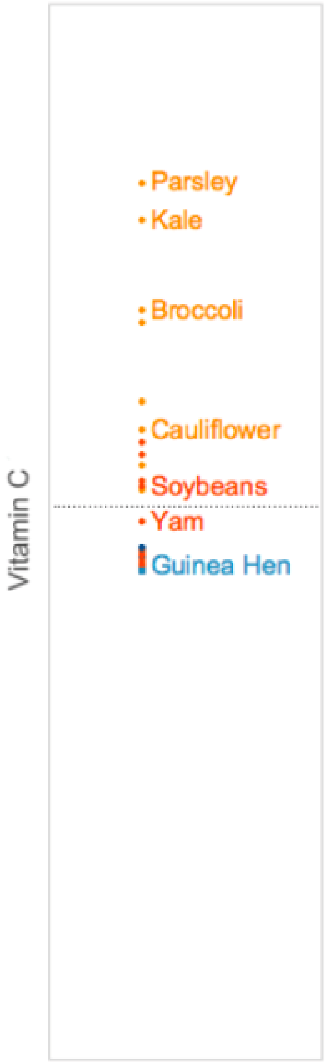
a) Yes
b) No

(if you say yes, argue why. If not, create a toy example to explain why not)

PCA FOR NUTRITION DATA

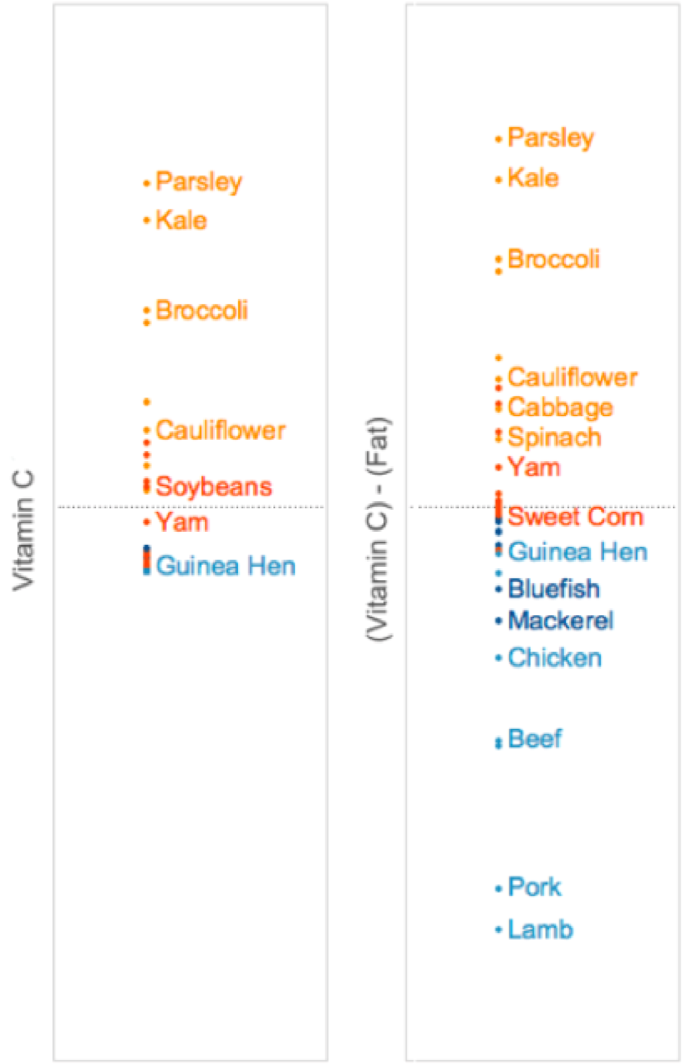
Shrt_Desc	Protein_(g)	Fiber_TD_(g)	Vit_C_(mg)	FA_Sat_(g)
CHICKEN,GIZZARD,ALL CLASSES,RAW	17.66	0	3.7	0.529
PORK,FRSH,LEG (HAM),RUMP HALF,LN&FAT,CKD,RSTD	27.03	0	0	3.369
BABYFOOD,DINNER,VEG&TURKEY,STR	2.32	1.5	0.7	0.236
CEREALS RTE,FRSTD OAT CRL W/MARSHMALLOWS	7.1	4.3	20	0.62
CANDIES,FRUIT SNACKS,W/ HI VIT C	0.08	0	136.4	0
PIKE,NORTHERN,CKD,DRY HEAT	24.69	0	3.8	0.151
APPLEBEE'S,CHICK TENDERS,FROM KIDS' MENU	19.25	1.2		2.852
FRUIT JUC SMOOTHIE,BOLTHOUSE FARMS,BERRY BOOST	0.63	0	108.6	0.003
NOODLES,EGG,DRY,UNENR	14.16	3.3	0	1.18
BABYFOOD,FRUIT,BANANAS W/TAPIOCA,STR	0.4	1.6	16.7	0.035
EGG,WHOLE,COOKED,OMELET	10.57	0	0	3.319
FAST FOODS,BISCUIT,W/HAM	11.85	0.7	0.1	10.096
CEREALS RTE,GENERAL MILLS,COOKIE CRISP	5.2	5.1	23.1	0.8
CANDIES,MARS SNACKFOOD US,MILKY WAY MIDNIGHT BAR	3.2	2.9	0.2	11.474
BEEF,RND,BTTM RND RST,LN,1/8" FAT,SEL,CKD,RSTD	28.45	0	0	1.608
BEEF,NZ,IMP,SUBCUTANEOUS FAT,CKD	6.5	0	0	30.256
WHEAT GERM,CRUDE	23.15	13.2	0	1.665
APPLEBEE'S,FRENCH FR	3.31	3.9	0.7	2.333
TSTR PSTRS,KELLOGG,KELLOGG'S LF POP TARTS,FRSTD STRAWBERRY	4.2	5.6		1.7
CHICKEN,BROILERS OR FRYERS,WING,MEAT ONLY,CKD,RSTD	30.46	0	0	2.26
DILL WEED,DRIED	19.96	13.6	50	0.234
FAT,BEEF TALLOW	0	0	0	49.8
TUNA,LT,CND IN H2O,WO/SALT,DRND SOL	25.51	0	0	0.234
PORK SAUSAGE,LINK/PATTY,UNPREP	15.39	0	0	7.57
INF FORMULA,NESTLE,GOOD START SUPREME,W/ IRON,PDR	11.3	0	46.1	11.66
BEEF,RIB,BACK RIBS,BONE-IN,LN,0" FAT,CHOIC,RAW	18.72	0	0	7.812
BREAD,PUMPERNICKEL	8.7	6.5	0	0.437
PIE,BANANA CRM,PREP FROM RECIPE	4.4	0.7	1.6	3.758
DIGIORNO PIZZA,CHS TOPPING,CHS STUFFED CRUST,FRZ,BKD	13.48	1.9	0.5	5.63
PORK,FRSH,LOIN,BLADE (ROASTS),BNLESS,LN,CKD,RSTD	27.58	0	0	2.56

TRYING TO MAP FOOD INTO 1DIM



https://annalyzin.wordpress.com/2016/06/15/principal-component-analysis-tutorial/

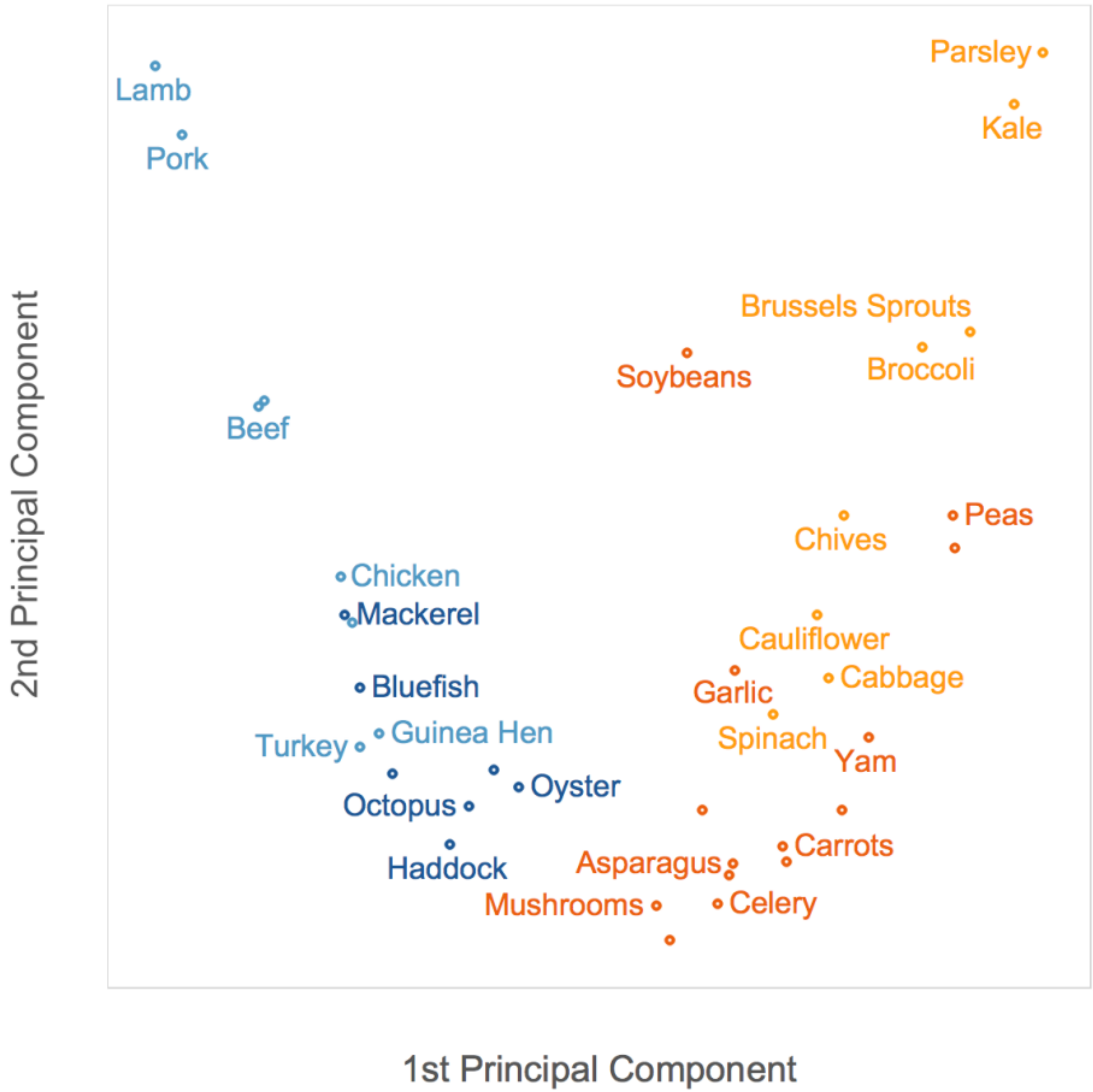
TRYING TO MAP FOOD INTO 1DIM



TRYING TO MAP FOOD INTO 1DIM

	PC1
Fat	-0.45
Protein	-0.55
Fiber	0.55
Vitamin C	0.44

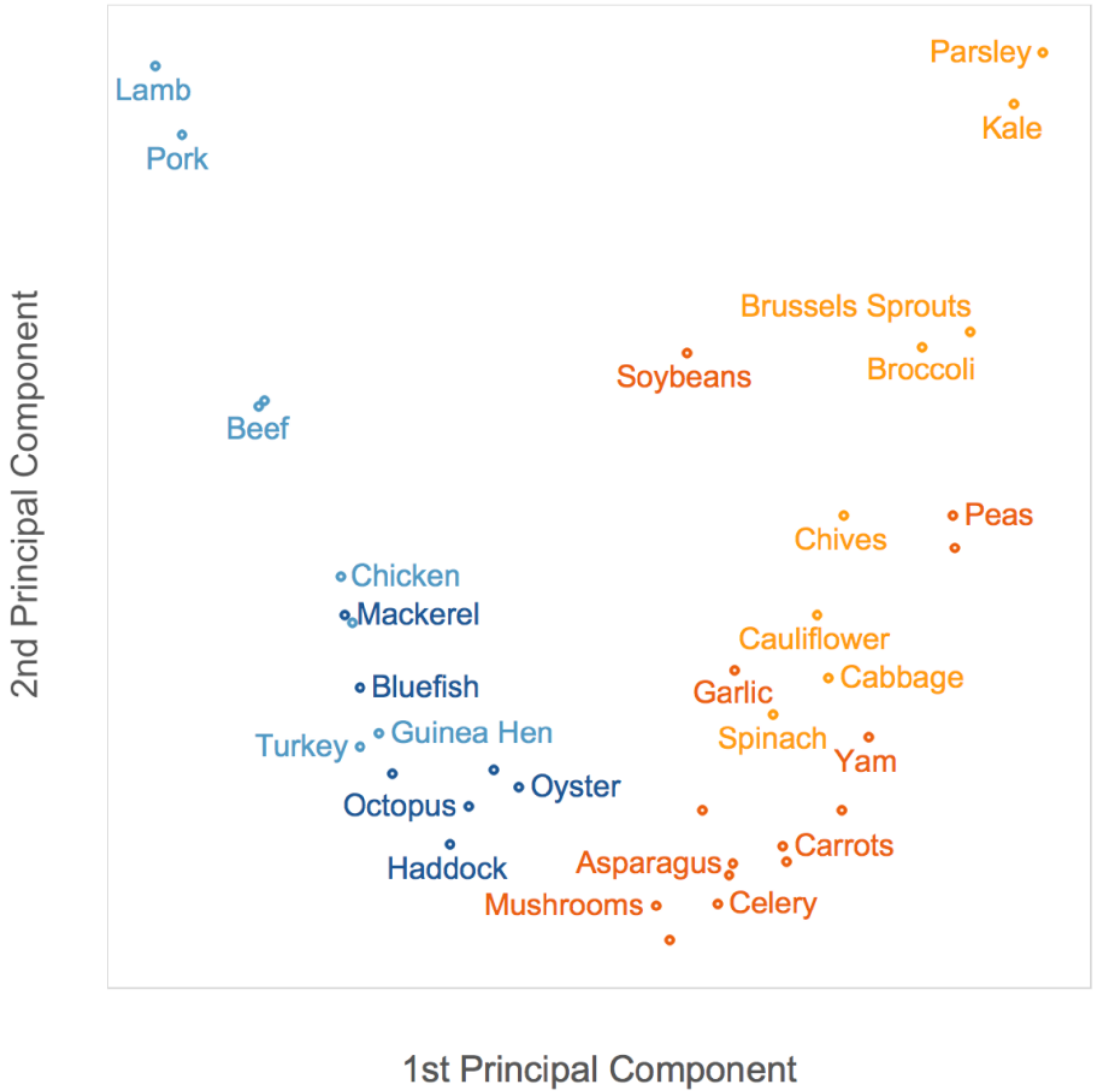
	PC1	PC2	PC3	PC4
Fat	-0.45	0.66	0.58	0.18
Protein	-0.55	0.21	-0.46	-0.67
Fiber	0.55	0.19	0.43	-0.69
Vitamin C	0.44	0.70	-0.52	0.22



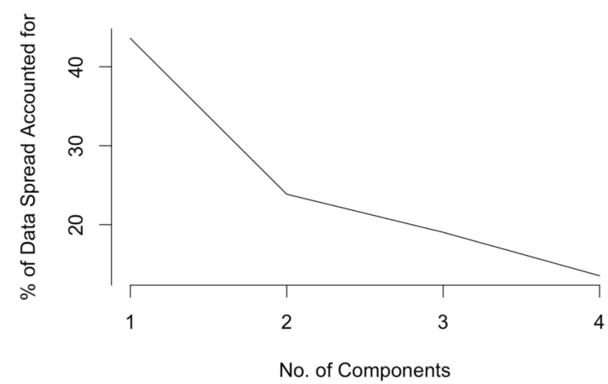
	PC1	PC2
Fat	-0.45	0.66
Protein	-0.55	0.21
Fiber	0.55	0.19
Vitamin C	0.44	0.70

https://annalyzin.wordpress.com/2016/06/15/principal-component-analysis-tutorial/

HOW MANY COMPONENTS TO USE



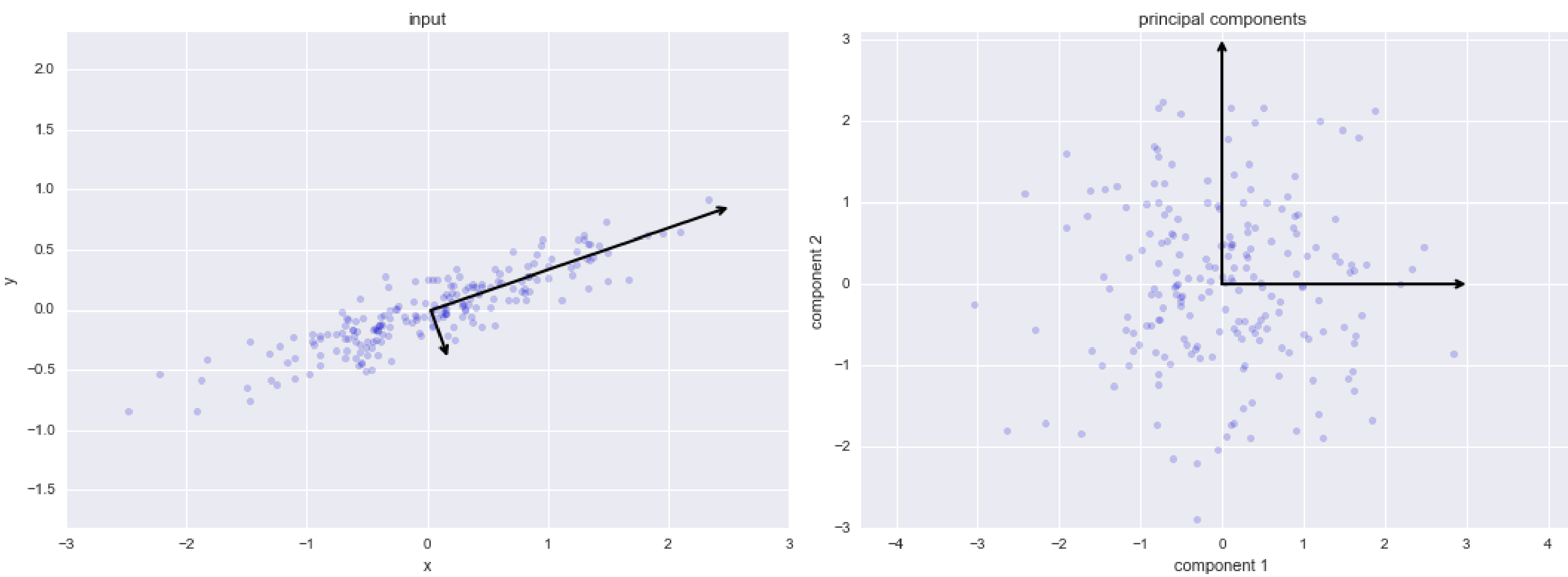
	PC1	PC2	PC3	PC4
Fat	-0.45	0.66	0.58	0.18
Protein	-0.55	0.21	-0.46	-0.67
Fiber	0.55	0.19	0.43	-0.69
Vitamin C	0.44	0.70	-0.52	0.22



https://annalyzin.wordpress.com/2016/06/15/principal-component-analysis-tutorial/

PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components



https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.09-Principal-Component-Analysis.ipynb#scrollTo=V7ssdJKEi9O1