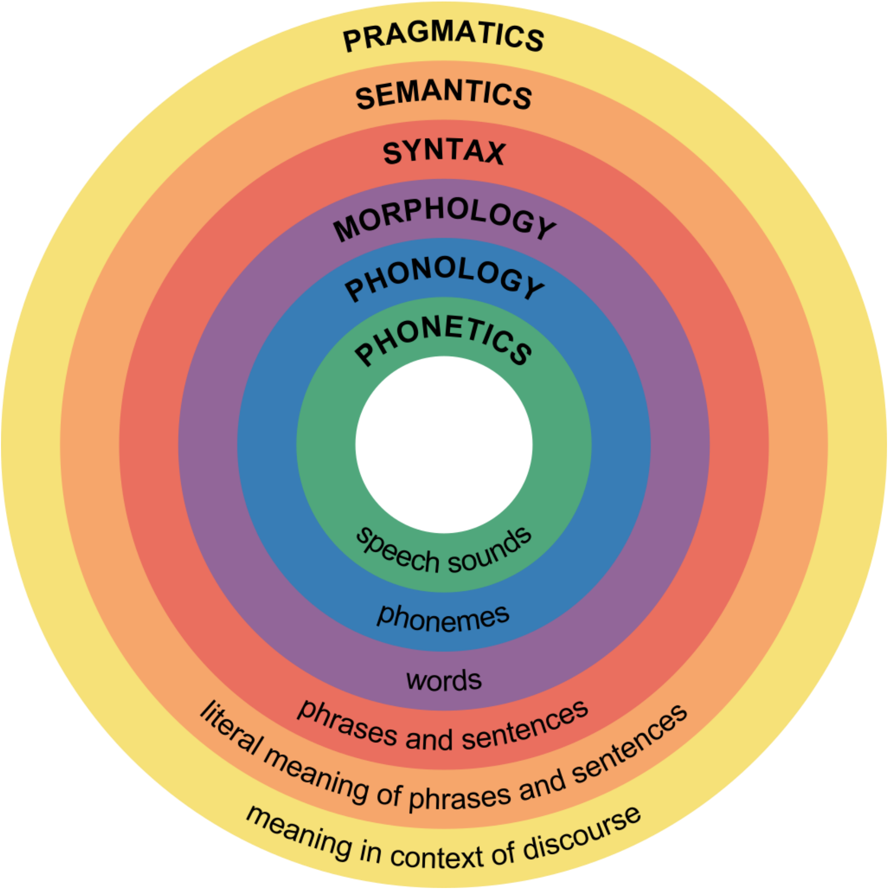
NLP Review

CMSC 473/673 - NATURAL LANGUAGE PROCESSING

Slides modified from Dr. Frank Ferraro

Linguistics

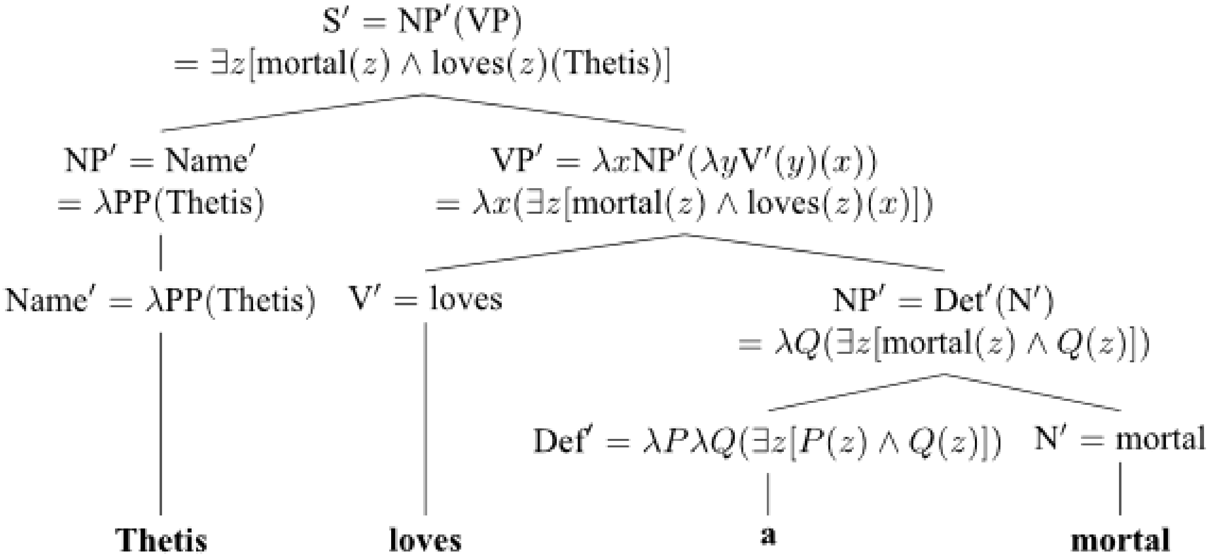
The study of language



https://en.wikipedia.org/wiki/Morphology_(linguistics)#/media/File:Major_levels_of_linguistic_structure.svg

Semantics

Meaning



$S' = NP'(VP)$
$= \exists z[\text{mortal}(z) \wedge \text{loves}(z)(\text{Thetis})]$

$NP' = \text{Name}'$
$= \lambda PP(\text{Thetis})$

$VP' = \lambda x NP'(\lambda y V'(y)(x))$
$= \lambda x(\exists z[\text{mortal}(z) \wedge \text{loves}(z)(x)])$

$\text{Name}' = \lambda PP(\text{Thetis})$

$V' = \text{loves}$

$NP' = \text{Det}'(N')$
$= \lambda Q(\exists z[\text{mortal}(z) \wedge Q(z)])$

$\text{Def}' = \lambda P \lambda Q(\exists z[P(z) \wedge Q(z)])$

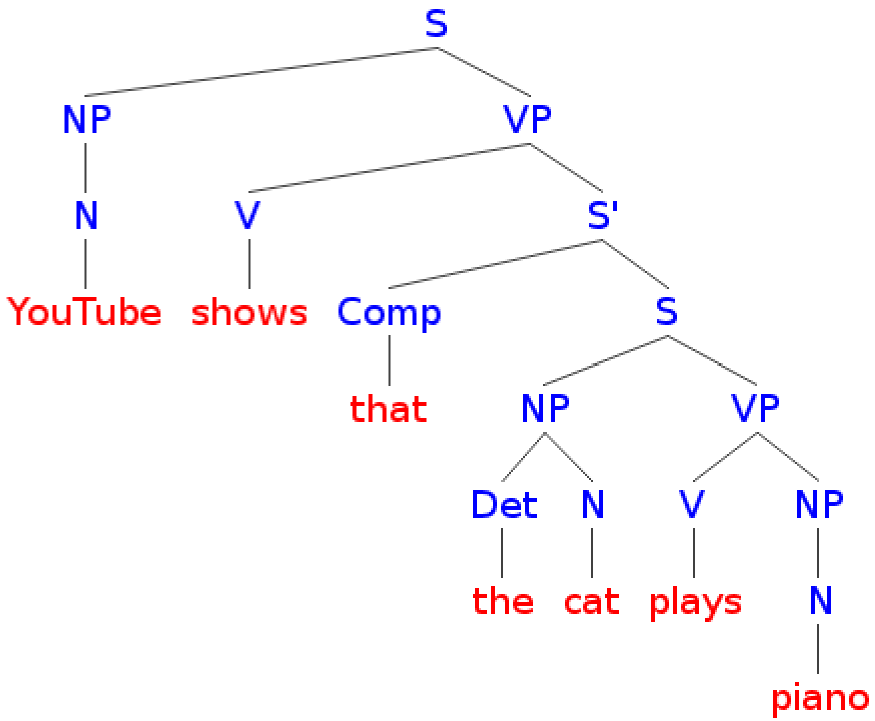
$N' = \text{mortal}$

Thetis **loves** **a** **mortal**

https://plato.stanford.edu/entries/computational-linguistics/

Syntax

Grammar



https://allthingslinguistic.com/post/100617668093/how-to-draw-syntax-trees-part-3-type-1-a

Phonology

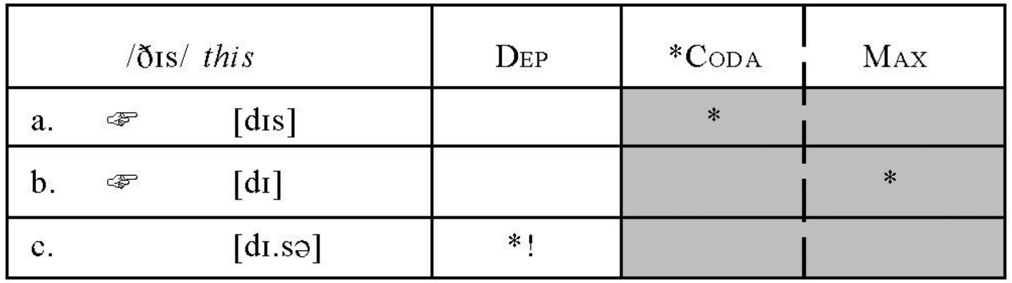
Processing of sounds

https://upload.wikimedia.org/wikipedia/commons/a/a5/Tsunami_by_hokusai_19th_century.jpg

tsunami

↓

sunami

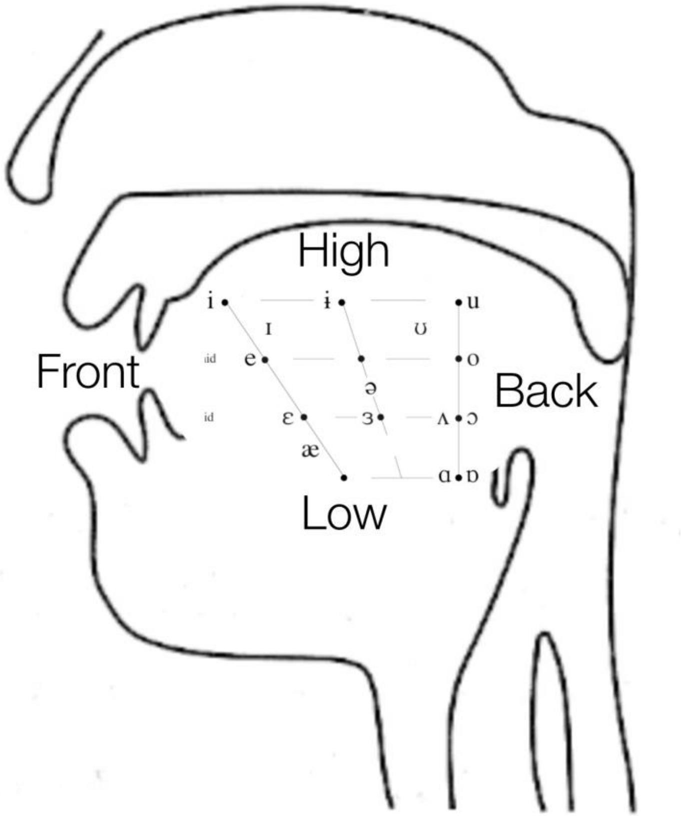


/ðɪs/ *this*			DEP	*CODA	MAX
a.	☞	[dɪs]		*	
b.	☞	[dɪ]			*
c.		[dɪ.sə]	*!		

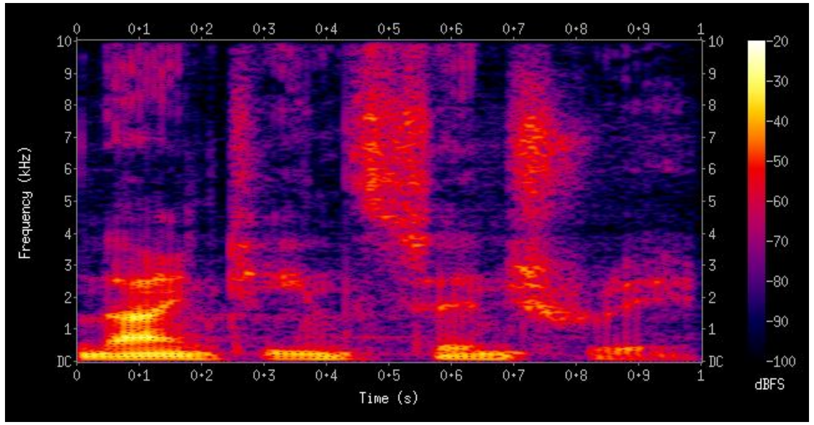
https://pubs.asha.org/doi/10.1044/0161-1461%282001/022%29

Phonetics

Physical production/understanding of sounds



https://wstyler.ucsd.edu/talks/l111_3_phonetics_review_handout.html



https://en.wikipedia.org/wiki/Spectrogram#/media/File:Spectrogram-19thC.png

ML/NLP Framework

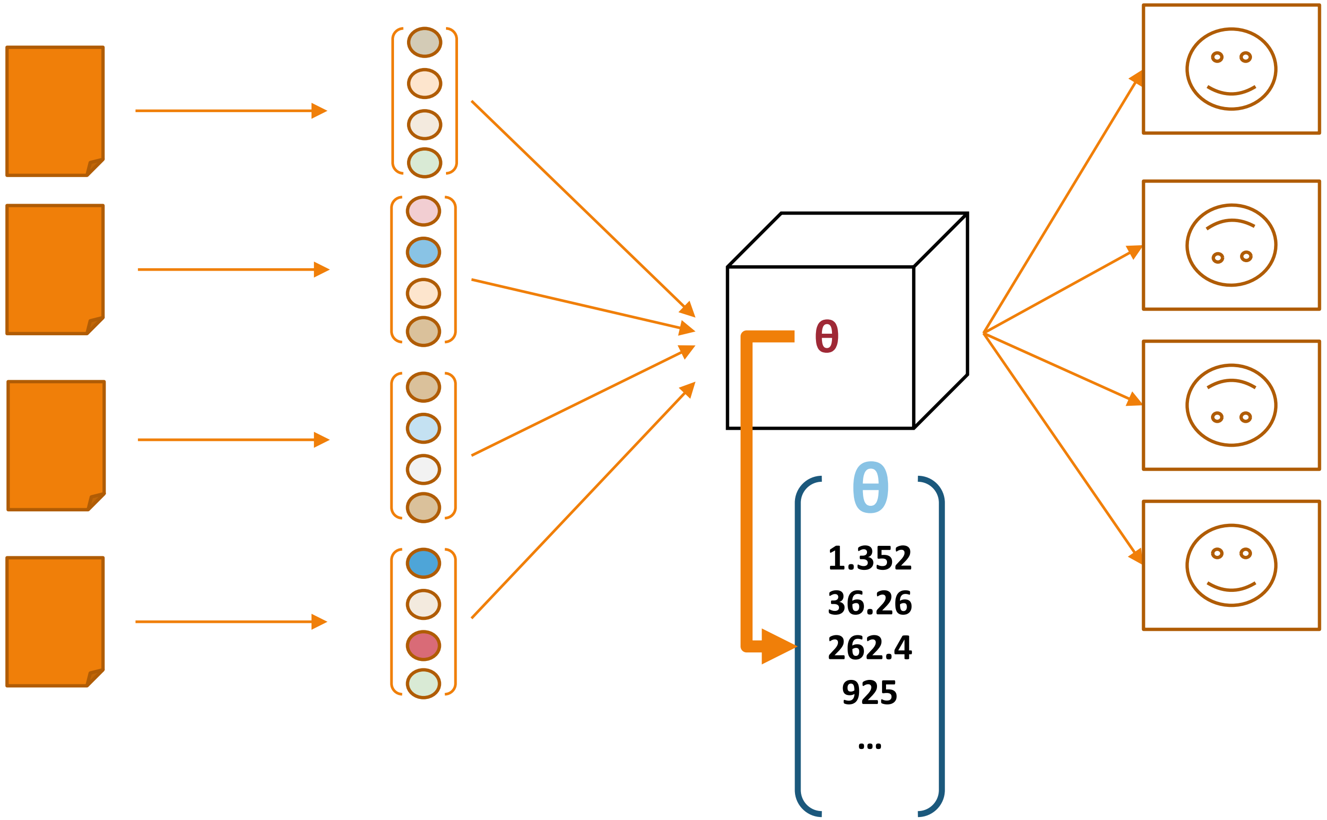
instances

features:

K-dimensional vector representations (one per instance)

ML model:

- take in featurized input
- output scores/labels
- contains weights θ



Helpful ML Terminology

Model: the (computable) way to go from **features** (input) to labels/scores (output)

Weights/parameters: vectors of numbers that control how the model produces labels/scores from inputs. These are learned through **training**.

Objective function: an algorithm/calculation, whose variables are the **weights** of the **model**, that we numerically optimize in order to learn appropriate weights based on the labels/scores. The **model's** weights are adjusted.

Evaluation function: an algorithm/calculation that scores how "correct" the **model's** predictions are. The **model's** weights are not adjusted.

Note: The evaluation and objective functions are often different!

(More) Helpful ML Terminology

Training / Learning:

- the process of adjusting the model's weights to learn to make good predictions.

Inference / Prediction / Decoding / Classification:

- the process of using a model's existing weights to make (hopefully!) good predictions

ML/NLP Framework for <u>Learning</u>

instances

features: K-dimensional vector representations (one per instance)

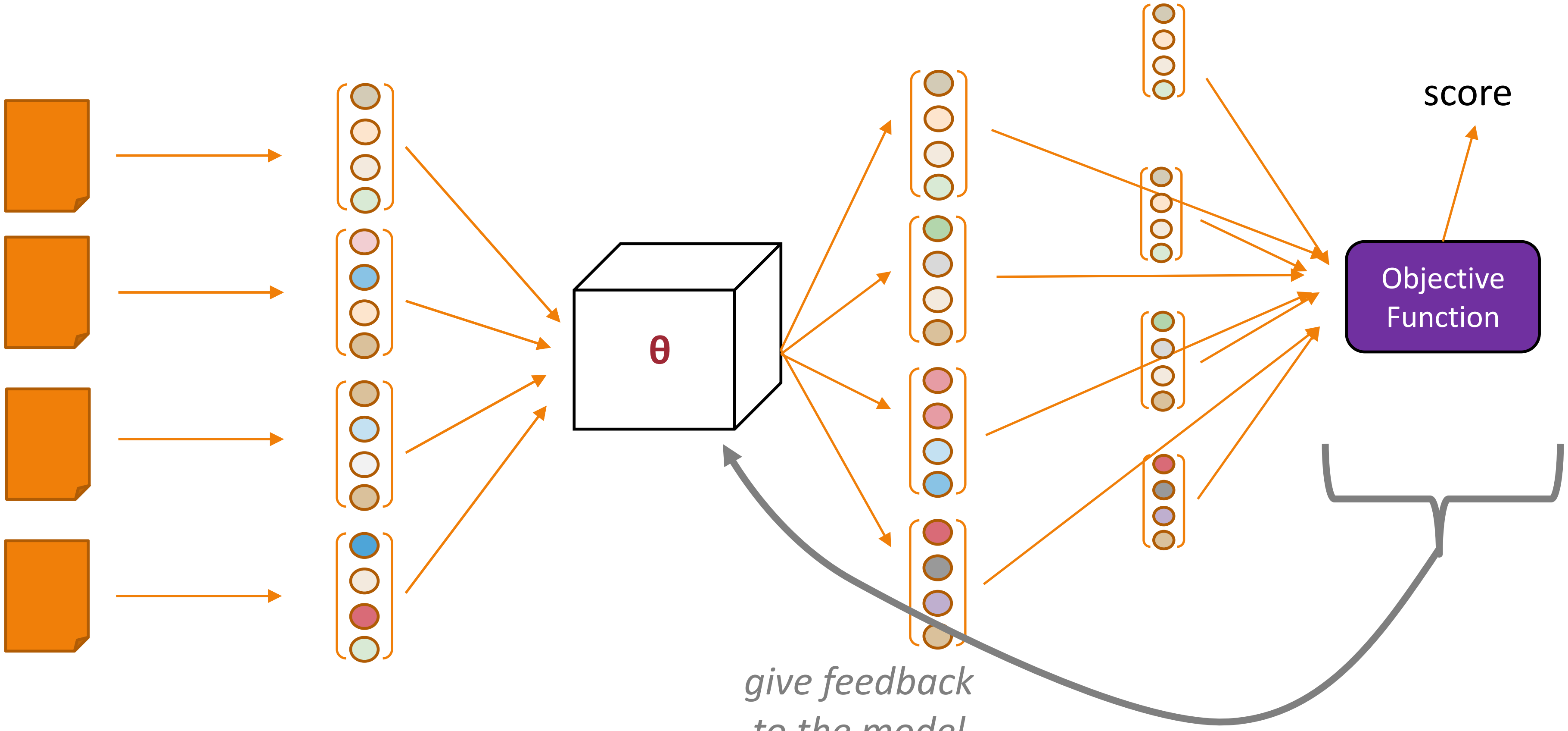
ML model:

- take in featurized input
- output scores/labels
- contains weights θ

output

“Gold” (correct) labels

Objective Function



ML/NLP Framework for Prediction

instances

features: **K-dimensional vector representations** (one per instance)

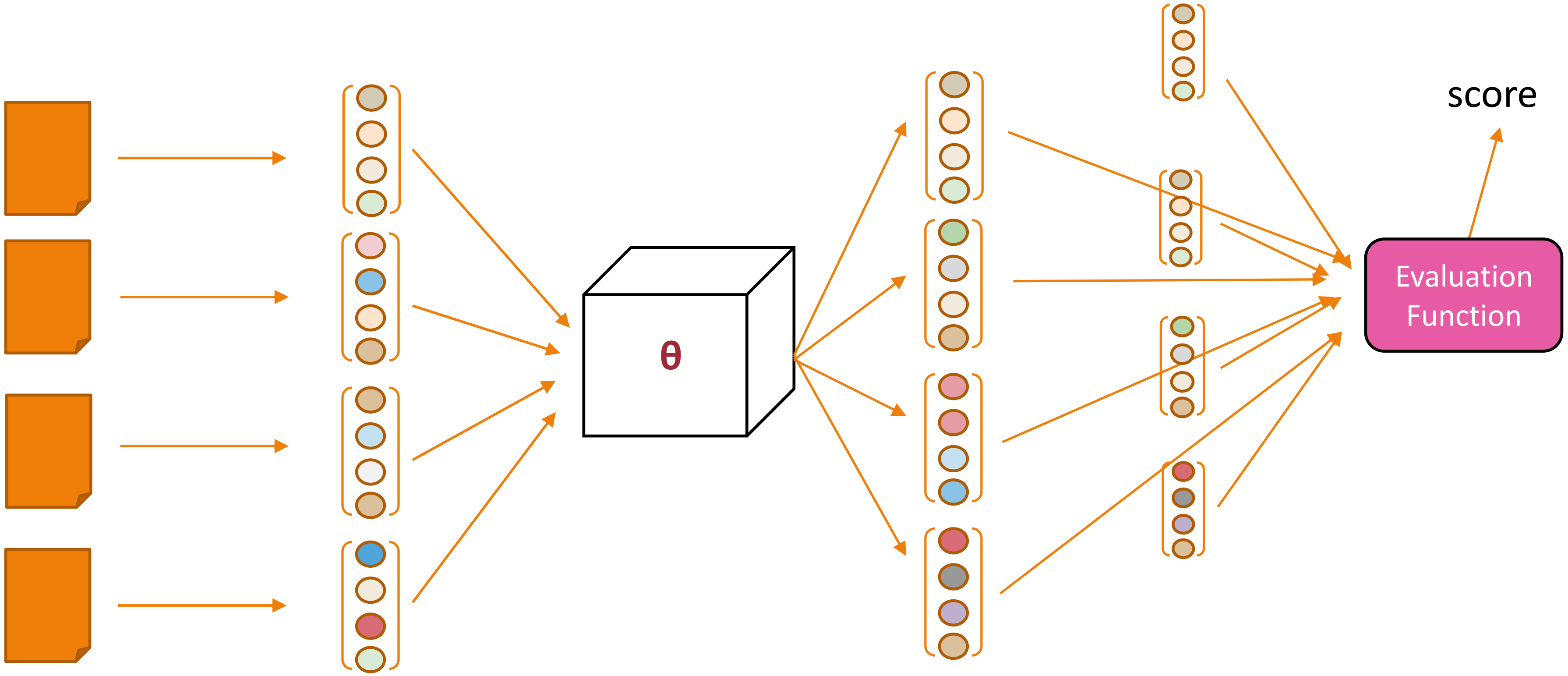
ML model:

- take in featurized input
- output scores/labels
- contains weights θ

output

“Gold” (correct) labels

Evaluation Function



instances

features: **K-dimensional vector representations** (one per instance)

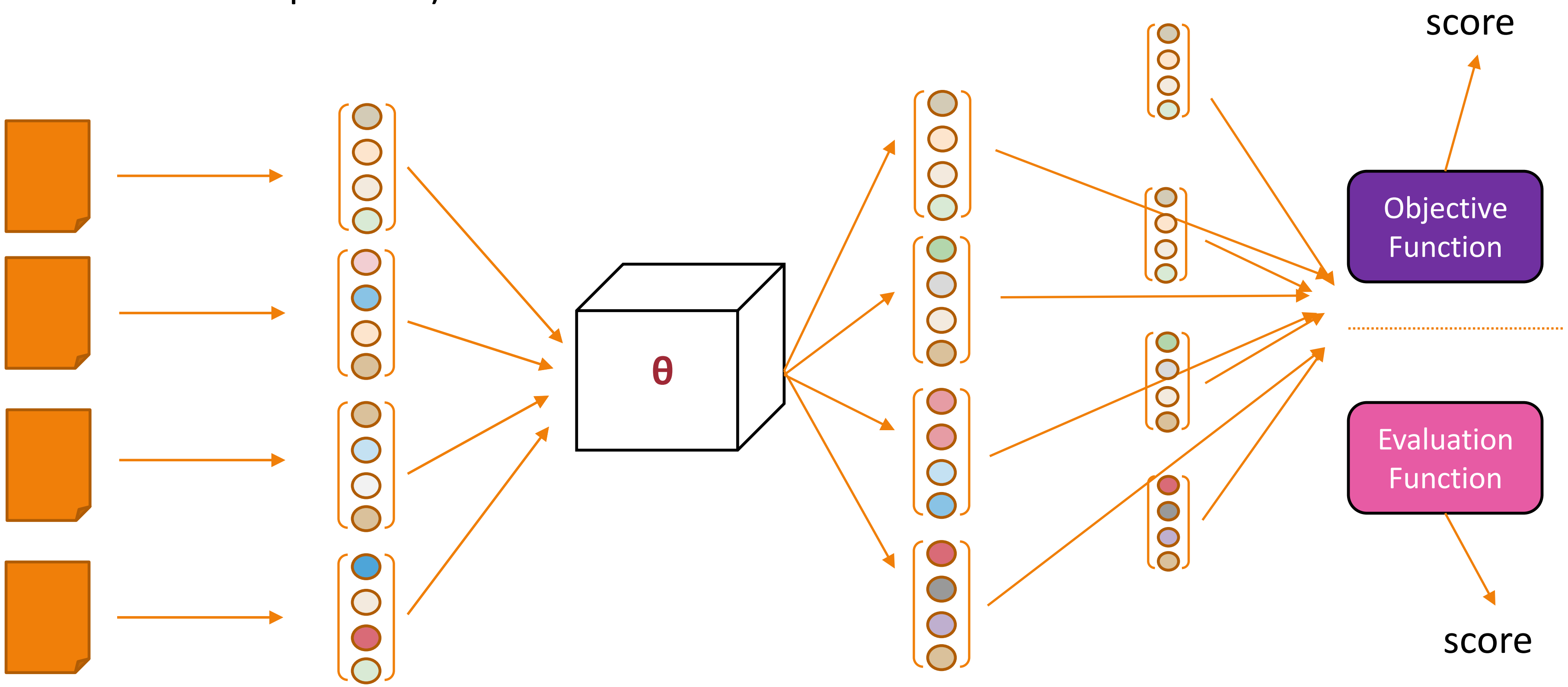
ML model:

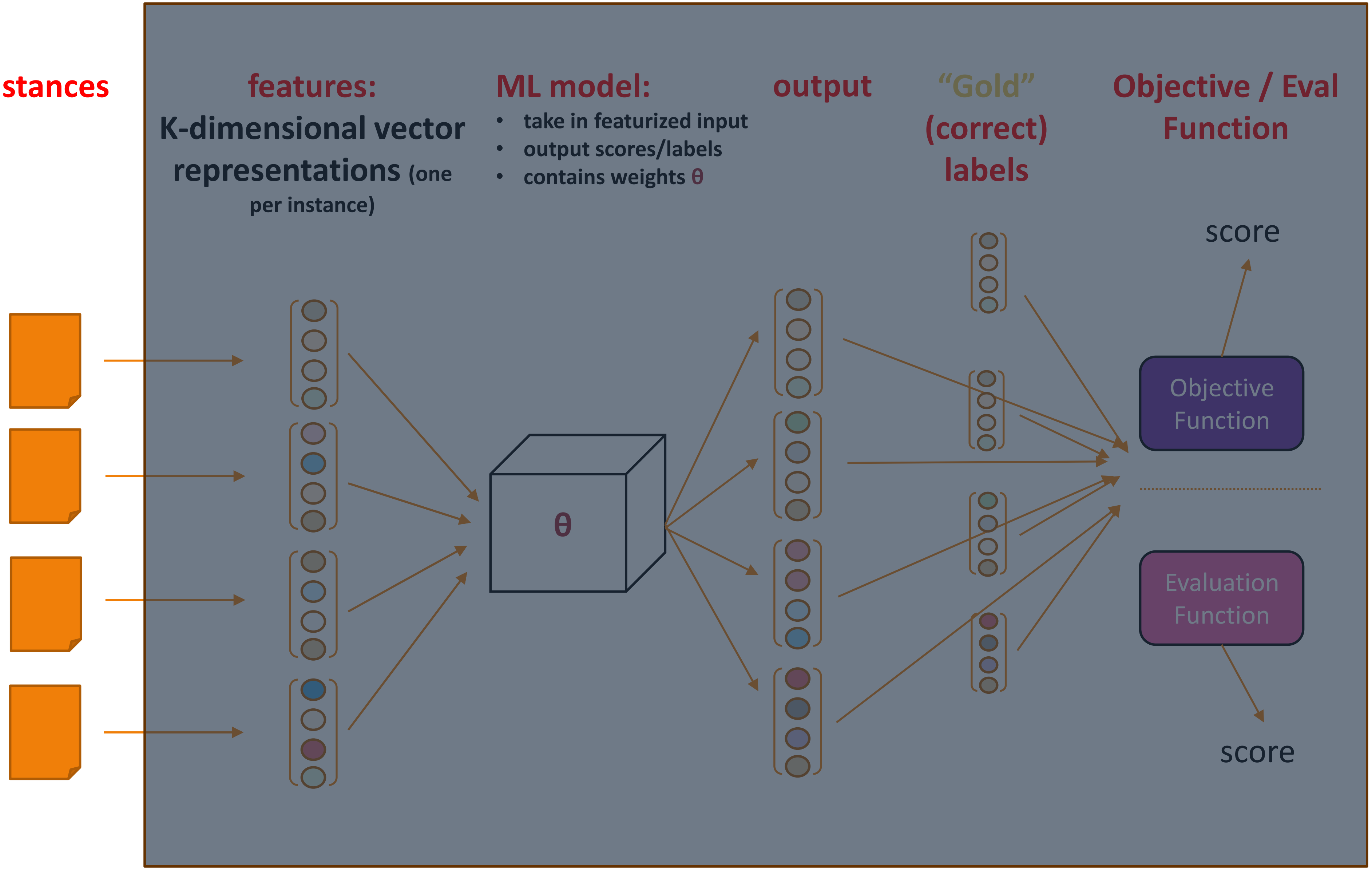
- take in featurized input
- output scores/labels
- contains weights θ

output

"Gold" (correct) labels

Objective / Eval Function





instances

features:
K-dimensional vector representations (one per instance)

ML model:

- take in featurized input
- output scores/labels
- contains weights θ

output

“Gold” (correct) labels

Objective / Eval Function

Where does the data come from?

Corpus (plural: corpora)

- Literally a “body” of text

Languages with few corpora are called “low-resource languages”

- This might not mean the language is endangered!

We can collect corpora in a few different ways:

- Curation: data tagged & organized by experts
- Internet: data “scraped” from open-access sources (Wikipedia, Reddit)
  - Or data collected with permission from closed sources (Facebook, texts) – more rare
- Elicitation: carefully getting participants to produce language (lab studies, crowdsourcing, field studies)
- Pre-existing corpora

! Facebook has gotten into trouble several times for using data or manipulating people’s feeds without their permission

Tasks!

Benchmarking

If you want people to work on your problem, make it easy for them to get started and to measure their progress. Provide:

- Test data, for evaluating the final systems
- Development data, for measuring whether a change to the system helps, and for tuning parameters
- An evaluation metric (formula for measuring how well a system does on the dev or test data)
- A program for computing the evaluation metric
- Labeled training data and other data resources
- A prize? – with clear rules on what data can be used

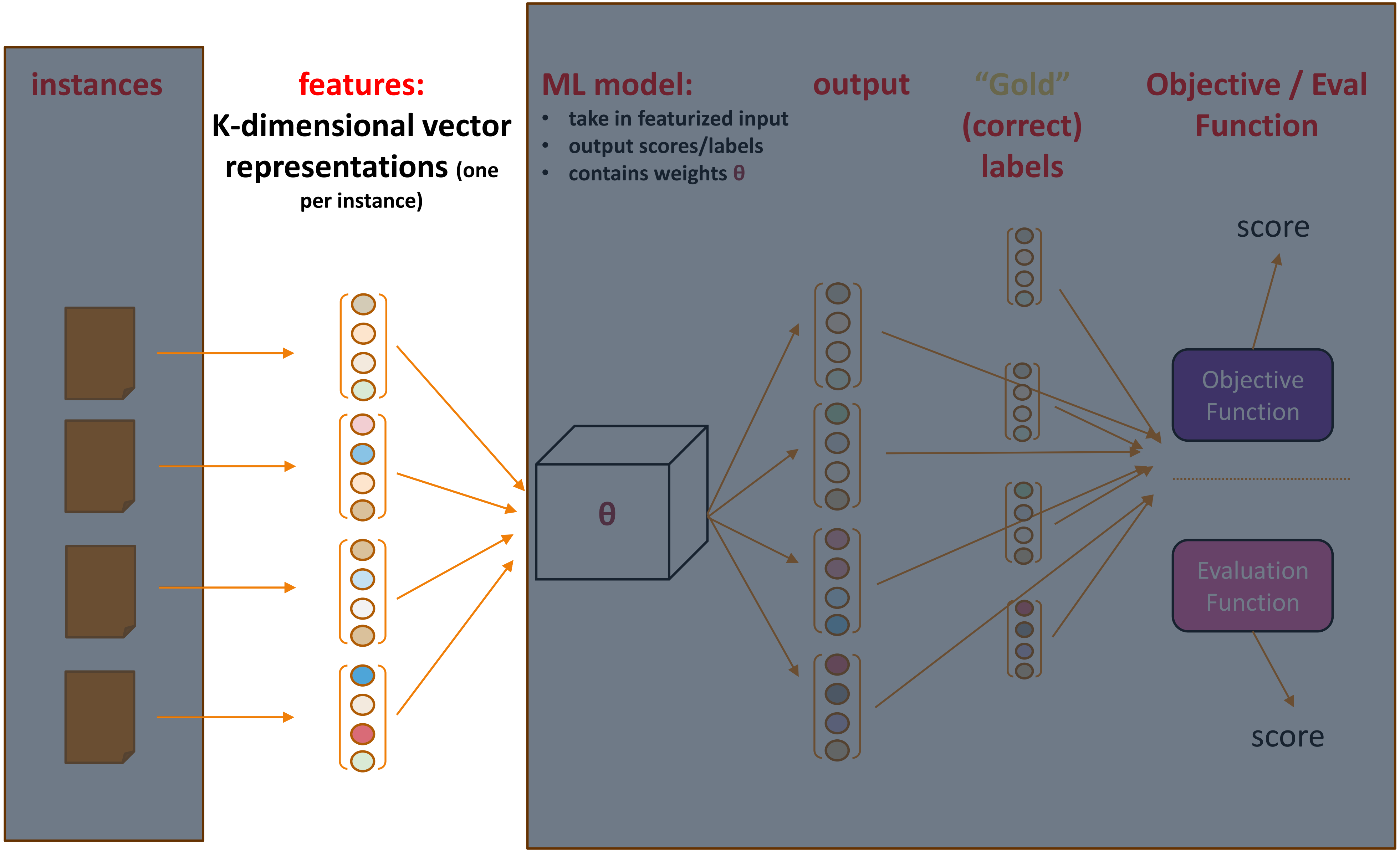
Tokens vs Types

The film got a great opening and the film went on to become a hit .

Vocabulary: the words (items) you know

Type: an element of the vocabulary.

Token: an instance of that type in running text.



instances

features: K-dimensional vector representations (one per instance)

ML model:

- take in featurized input
- output scores/labels
- contains weights θ

output

"Gold" (correct) labels

Objective / Eval Function

ML Term: “Featurization”

The procedure of extracting **features** for some input

Often viewed as a K-dimensional vector function *f* of the input language *x*

$$f(x) = (\underbrace{f_1(x)}, \ldots, \underbrace{f_K(x)})$$

Each of these is a feature
(/feature function)

Overview of Featurization

Common goal: probabilistic classifier p(y | x)

Often done by defining **features** between x and y that are meaningful

- Denoted by a **general vector of K features**

$$f(x) = (f_1(x), \dots, f_K(x))$$

Features can be thought of as “soft” rules

- E.g., POSITIVE sentiments tweets *may* be more likely to have the word “happy”

Representing Linguistic Information

User-defined

Bag of words

/ one-hot encoding

Assign each word to some index i, where $0 \leq i < V$

Represent each word w with a V-dimensional **binary** vector e_w, where $e_{w,i} = 1$ and 0 otherwise

Model-produced

Dense embedding

Let E be some *embedding size* (often 100, 200, 300, etc.)

Represent each word w with an E-dimensional **real-valued** vector e_w

Bag-of-words

Bag-of-words (or bag-of-characters, bag-of-relations)

- Identify ***unique*** sufficient atomic sub-parts (e.g., words in a document)
- Define simple features over these, e.g.,
  - Binary (0 or 1) ➔ indicating presence
  - Natural numbers ➔ indicating number of times in a context
  - Real-valued ➔ various other score (we’ll see examples throughout the semester)

Example: Document Classification via Bag-of-Words Features

Amazon acquired MGM in 2022, taking over a sprawling library that includes more than 4,000 feature films and 17,000 television shows. The tech behemoth also earned the rights to distribute all the Bond movies, but the new deal solidifies the company's oversight of Bond's big-screen future.

TECH

NOT TECH

feature extraction

With V word types, define V feature functions $f_i(x)$ as

$f_i(x)$ =# of times word type *i* appears in document x

$$f(x) = \left(f_i(x)\right)_i^V$$

Core assumption: the label can be predicted from counts of individual word types

Adapted from https://www.nbcnews.com/pop-culture/movies/amazon-taking-control-james-bond-movie-franchise-rcna192967

Example: Document Classification via Bag-of-Words Features

Amazon acquired MGM in 2022, taking over a sprawling library that includes more than 4,000 feature films and 17,000 television shows. The tech behemoth also earned the rights to distribute all the Bond movies, but the new deal solidifies the company's oversight of Bond's big-screen future.

Tech

Not Tech

feature extraction

feature $f_i(x)$	value
Amazon	1
acquired	1
behemoth	1
Bond	2
…	
sniffle	0
…	

Core assumption: the label can be predicted from counts of individual word types

Adapted from https://www.nbcnews.com/pop-culture/movies/amazon-taking-control-james-bond-movie-franchise-rcna192967

How have we represented words?

Each word is a distinct item

- Bijection between the strings and unique integer ids:
- "cat" --> 3, "kitten" --> 792 "dog" --> 17394
- Are "cat" and "kitten" similar?

Equivalently: "One-hot" encoding

- Represent each word type w with a vector the size of the vocabulary
- This vector has V-1 zero entries, and 1 non-zero (one) entry

One-Hot Encoding Example

Let our vocab be {a, cat, saw, mouse, happy}

V = # types = 5

Assign:

a	4
cat	2
saw	3
mouse	0
happy	1

How do we represent “cat?”

$$e_{\text{cat}} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

How do we represent “happy?”

$$e_{\text{happy}} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Useful Terminology: n-gram

Within a larger string (e.g., sentence),
a contiguous sequence of n items (e.g., words)

Sometimes people use a bag of n-grams!

Colorless green ideas sleep furiously

n	Commonly called	History Size (Markov order)	Example n-gram ending in "furiously"
1	unigram	0	furiously
2	bigram	1	sleep furiously
3	trigram (3-gram)	2	ideas sleep furiously
4	4-gram	3	green ideas sleep furiously
n	n-gram	n-1	$w_{i-n+1} \ldots w_{i-1}\ w_i$

Representing Linguistic Information

User-defined

Bag of words

/ one-hot encoding

Assign each word to some index i, where $0 \leq i < V$

Represent each word w with a V-dimensional **binary** vector e_w, where $e_{w,i} = 1$ and 0 otherwise

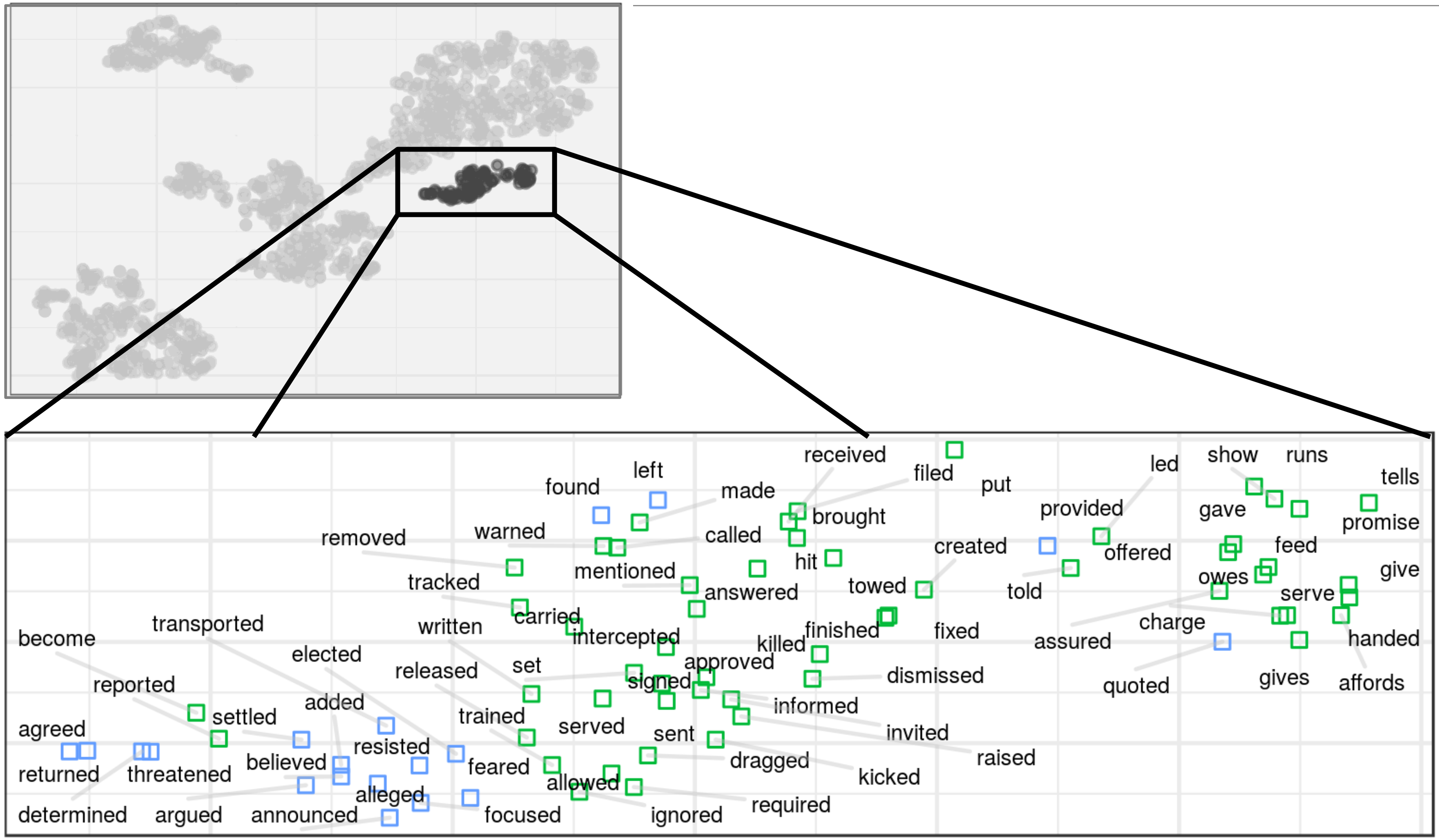
Model-produced

Dense embedding

Let E be some *embedding size* (often 100, 200, 300, etc.)

Represent each word w with an E-dimensional **real-valued** vector e_w

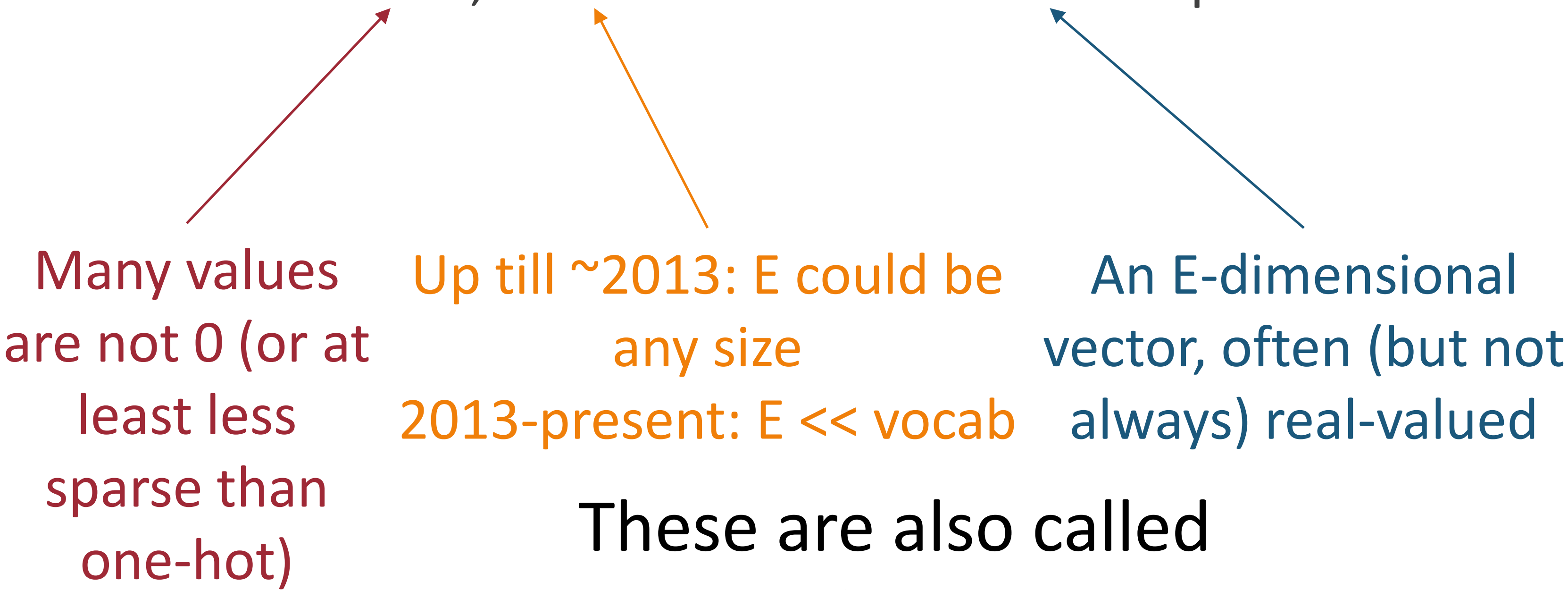
A Dense Representation (E=2)



[0.00315225, 0.00315225, 0.00547597, 0.00741556, 0.00912817, 0.01068435, 0.01212381, 0.01347162, 0.01474487, 0.0159558]

Distributional Representations

A dense, “low”-dimensional vector representation



Many values are not 0 (or at least less sparse than one-hot)

Up till ~2013: E could be any size
2013-present: E << vocab

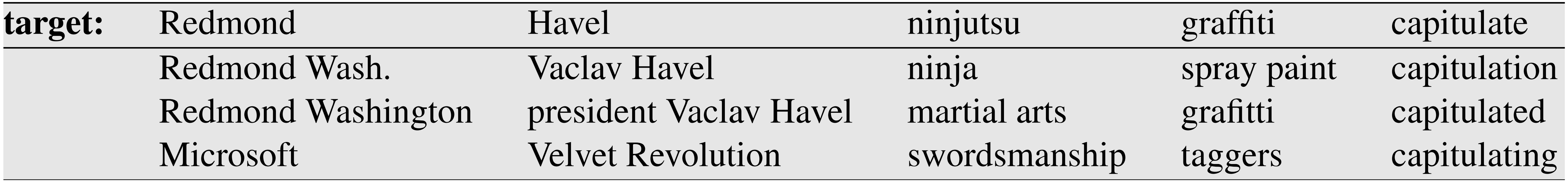
An E-dimensional vector, often (but not always) real-valued

These are also called

- **embeddings**
- **Continuous representations**
- **(word/sentence/…) vectors**
- **Vector-space models**

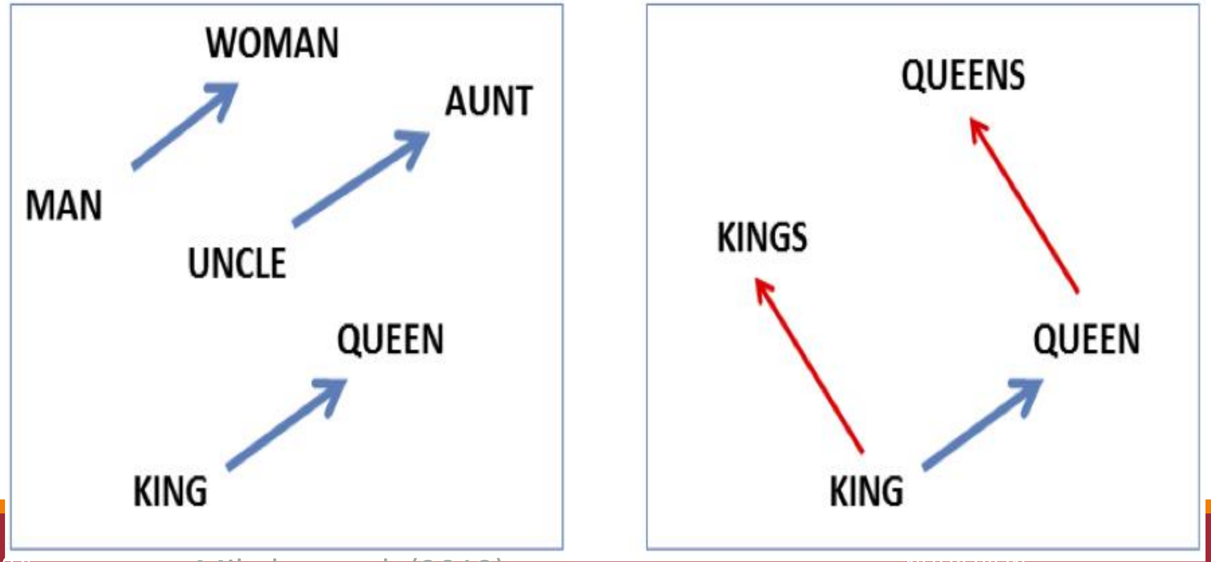
(Some) Properties of Embeddings

Capture “like” (similar) words



target:	Redmond	Havel	ninjutsu	graffiti	capitulate
	Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
	Redmond Washington	president Vaclav Havel	martial arts	grafitti	capitulated
	Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

Capture relationships



Mikolov et al. (2013)

vector(*‘king’*) –
vector(*‘man’*) +
vector(*‘woman’*) ≈
vector(‘queen’)

vector(*‘Paris’*) -
vector(*‘France’*) +
vector(*‘Italy’*) ≈
vector(‘Rome’)

Three Common Kinds of Embedding Models

1. Co-occurrence matrices
2. Matrix Factorization: Singular value decomposition/Latent Semantic Analysis, Topic Models
3. Neural-network-inspired models (skip-grams, CBOW)

Shared Intuition

Model the meaning of a word by “embedding” in a vector space

The meaning of a word is a vector of numbers

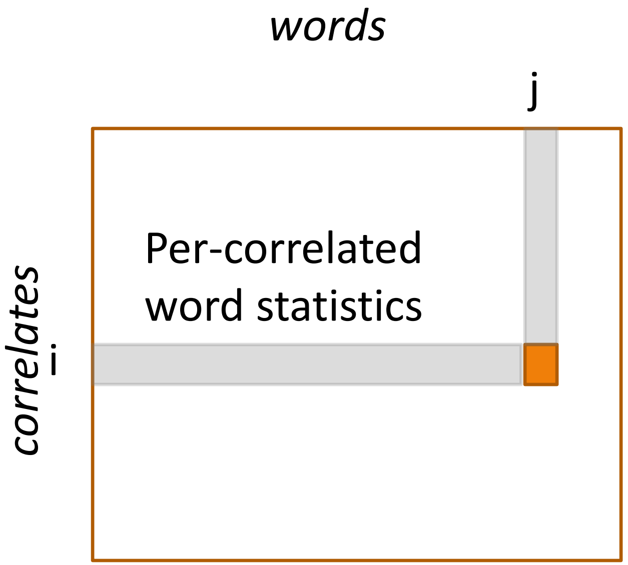
Contrast: word meaning is represented in many computational linguistic applications by a vocabulary index (“word number 545”) or the string itself

Three Common Kinds of Embedding Models

1. Co-occurrence matrices
2. Matrix Factorization: Singular value decomposition/Latent Semantic Analysis, Topic Models
3. Neural-network-inspired models (skip-grams, CBOW)

Co-occurrence Matrix

Acquire basic contextual statistics (often counts) for each word type v via *correlate*.



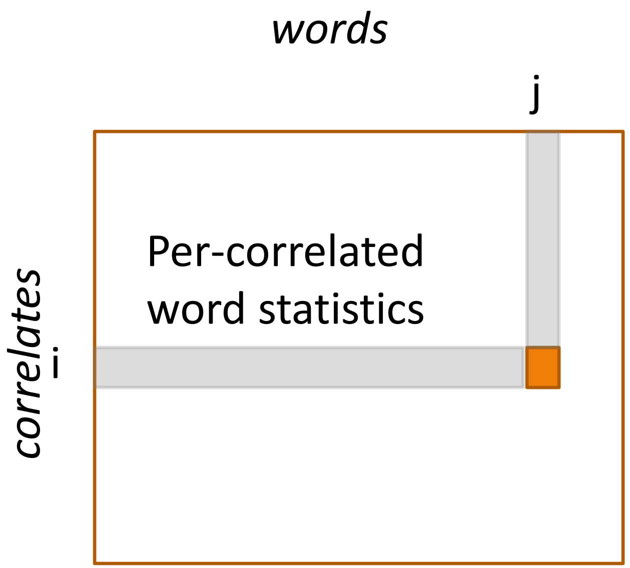
Co-occurrence Matrix

Acquire basic contextual statistics (often counts) for each word type v via *correlate*:

For example:

documents

- Record how often a word occurs in each document



# correlates = # documents

Co-occurrence Matrix

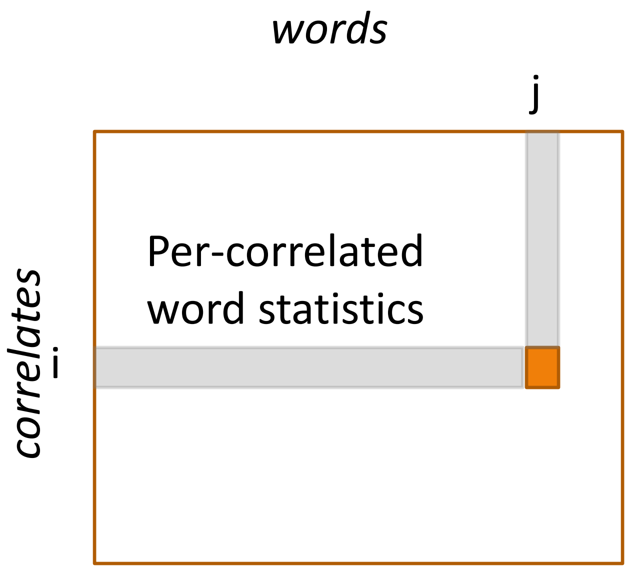
Acquire basic contextual statistics (often counts) for each word type v via *correlate*:

For example:

documents

surrounding context words

- Record how often v occurs with other word types u



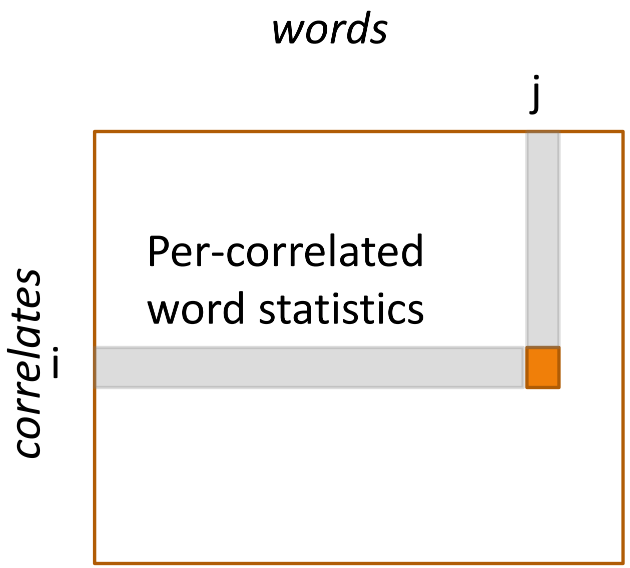
# correlates =
# word types

Co-occurrence Matrix

Acquire basic contextual statistics (often counts) for each word type v via *correlate*:

For example:

- documents
- surrounding context words
- linguistic annotations (POS tags, syntax)
- …



“You shall know a word by the company it keeps!” Firth (1957)

document (↓)-word (→) count matrix

	battle	soldier	fool	clown
As You Like It	1	2	37	6
Twelfth Night	1	2	58	117
Julius Caesar	8	12	1	0
Henry V	15	36	5	0

basic bag-of-words counting

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

fairy
always
love
to
it
it
whimsical
it
I
and
seen
are
anyone
friend
happy
dialogue
adventure
recommend
who
sweet
of
satirical
it
it
I
but
to
movie
romantic
I
several
yet
again
it
the
humor
the
seen
would
to
scenes
I
the
manages
fun
I
and
the
times
and
about
whenever
have
while
with
conventions

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

“You shall know a word by the company it keeps!” Firth (1957)

document (↓)-word (→) count matrix

	battle	soldier	fool	clown
As You Like It	1	2	37	6
Twelfth Night	1	2	58	117
Julius Caesar	8	12	1	0
Henry V	15	36	5	0

Assumption: Two documents are similar if their vectors are similar

“You shall know a word by the company it keeps!” Firth (1957)

document (↓)-word (→) count matrix

	battle	soldier	fool	clown
As You Like It	1	2	37	6
Twelfth Night	1	2	58	117
Julius Caesar	8	12	1	0
Henry V	15	36	5	0

Assumption: Two words are similar if their vectors are similar???

Issue: Count word vectors are very large, sparse, and skewed!

“You shall know a word by the company it keeps!” Firth (1957)

context (↓)-word (→) count matrix

	apricot	pineapple	digital	information
aardvark	0	0	0	0
computer	0	0	2	1
data	0	10	1	6
pinch	1	1	0	0
result	0	0	1	4
sugar	1	1	0	0

Context: those other words within a small “window” of a target word

a cloud [computer stores digital data on] a remote computer

“You shall know a word by the company it keeps!” Firth (1957)

context (↓)-word (→) count matrix

	apricot	pineapple	digital	information
aardvark	0	0	0	0
computer	0	0	2	1
data	0	10	1	6
pinch	1	1	0	0
result	0	0	1	4
sugar	1	1	0	0

Context: those other words within a small “window” of a target word

Assumption: Two words are similar if their vectors are similar

Issue: Count word vectors are very large, sparse, and skewed!

Pointwise Mutual Information (PMI): Dealing with Problems of Raw Counts

Raw word frequency is not a great measure of association between words

- It's very skewed: "the" and "of" are very frequent, but maybe not the most discriminative

We'd rather have a measure that asks whether a context word is **particularly informative** about the target word.

- (Positive) Pointwise Mutual Information ((P)PMI)

Pointwise mutual information:

Do events x and y co-occur more than if they were independent?

probability words x and y occur together (in the same context/window)

$$\mathrm{PMI}(x, y) = \log \frac{p(x, y)}{p(x)p(y)}$$

probability that word x occurs

probability that word y occurs

Three Common Kinds of Embedding Models

1. Co-occurrence matrices
2. Matrix Factorization: Singular value decomposition/Latent Semantic Analysis, Topic Models
3. Neural-network-inspired models (skip-grams, CBOW)

Word2Vec

Mikolov et al. (2013; NeurIPS): “Distributed Representations of Words and Phrases and their Compositionality”

Revisits the context-word approach

Learn a model p(*c* | *w*) to predict a context word *c* from a target word *w*

Learn two types of vector representations

- $h_c \in \mathbb{R}^E$: vector embeddings for each context word
- $v_w \in \mathbb{R}^E$: vector embeddings for each target word

$$p(c\,|w) \propto \exp(h_c^T v_w)$$

Word2Vec

context (↓)-word (→) count matrix

	apricot	pineapple	digital	information
aardvark	0	0	0	0
computer	0	0	2	1
data	0	10	1	6
pinch	1	1	0	0
result	0	0	1	4
sugar	1	1	0	0

Context: those other words within a small "window" of a target word

$$\max_{h,v} \sum_{c,w \text{ pairs}} \text{count}(c,w) \log p(c \mid w)$$

Word2Vec

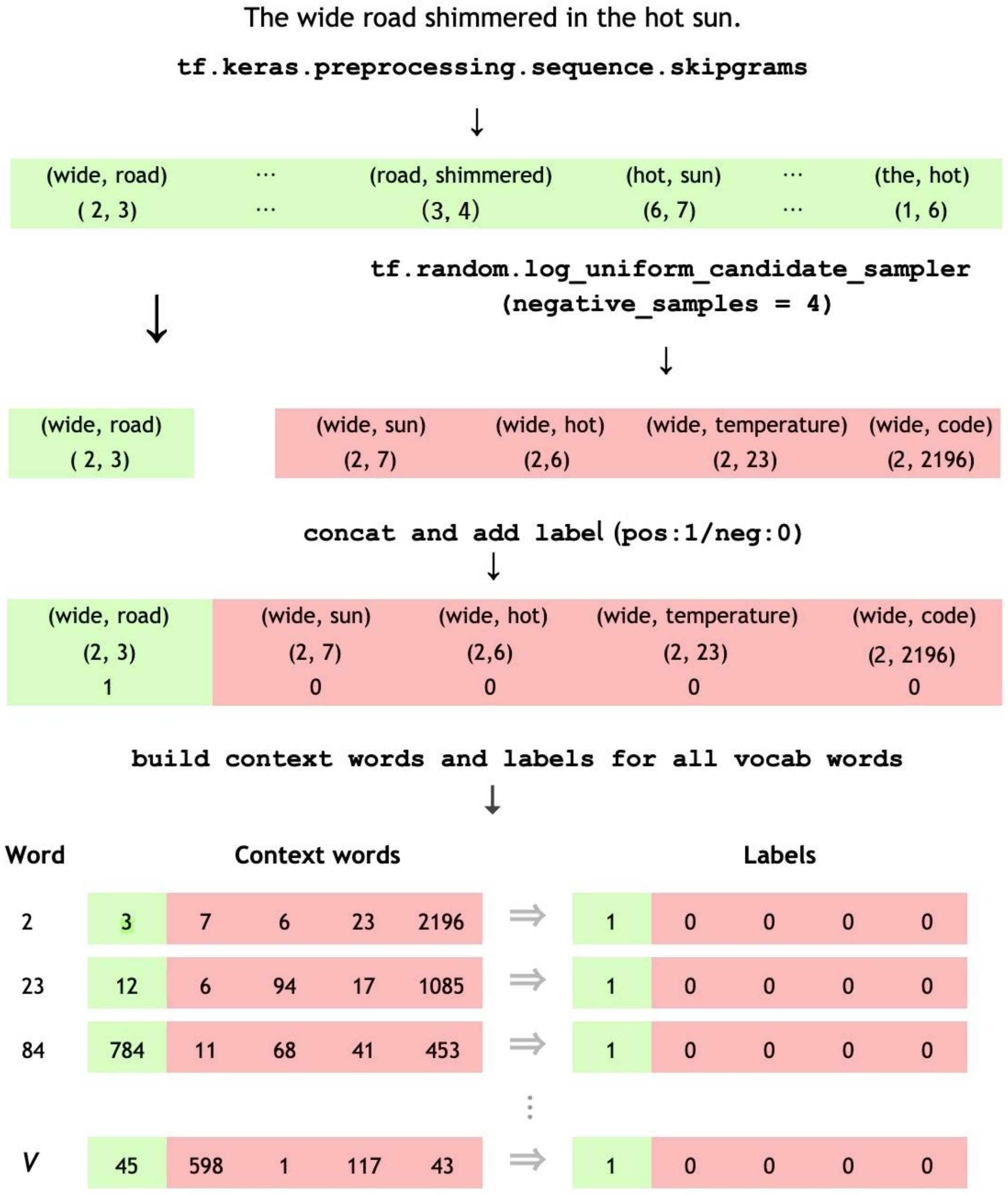
context (↓)-word (→) count matrix

	apricot	pineapple	digital	information
aardvark	0	0	0	0
computer	0	0	2	1
data	0	10	1	6
pinch	1	1	0	0
result	0	0	1	4
sugar	1	1	0	0

Context: those other words within a small "window" of a target word

$$\max_{h,v} \sum_{c,w \text{ pairs}} \text{count}(c,w) \left[h_c^T v_w - \log(\sum_u \exp(h_u^T v_w)) \right]$$

Example (Tensorflow)



The wide road shimmered in the hot sun.

`tf.keras.preprocessing.sequence.skipgrams`

↓

(wide, road)	⋯	(road, shimmered)	(hot, sun)	⋯	(the, hot)
(2, 3)	⋯	(3, 4)	(6, 7)	⋯	(1, 6)

↓

`tf.random.log_uniform_candidate_sampler (negative_samples = 4)`

↓

(wide, road)
(2, 3)

(wide, sun)	(wide, hot)	(wide, temperature)	(wide, code)
(2, 7)	(2,6)	(2, 23)	(2, 2196)

`concat and add label (pos:1/neg:0)`

↓

(wide, road)	(wide, sun)	(wide, hot)	(wide, temperature)	(wide, code)
(2, 3)	(2, 7)	(2,6)	(2, 23)	(2, 2196)
1	0	0	0	0

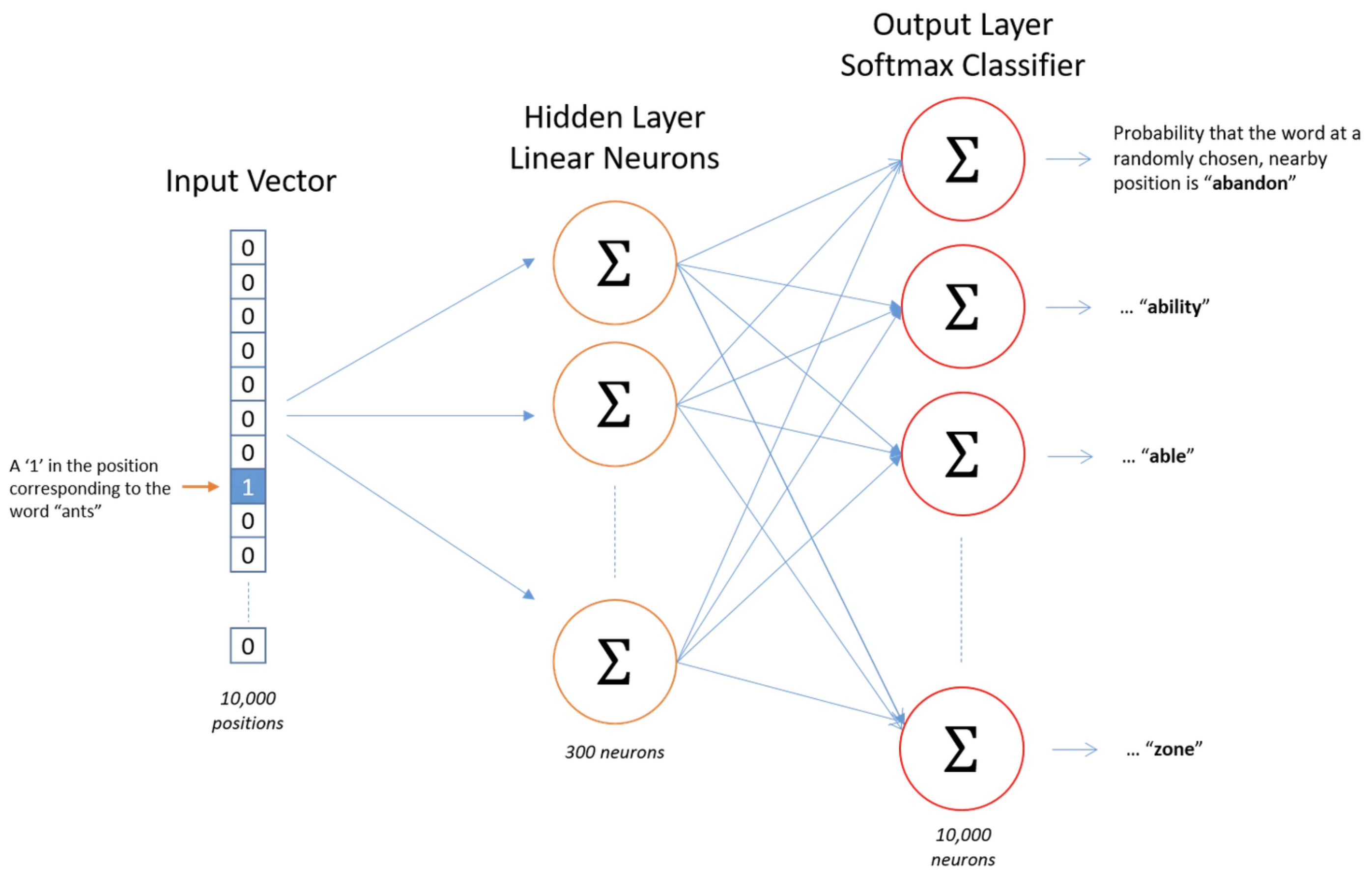
`build context words and labels for all vocab words`

↓

Word	Context words						Labels				
2	3	7	6	23	2196	⇒	1	0	0	0	0
23	12	6	94	17	1085	⇒	1	0	0	0	0
84	784	11	68	41	453	⇒	1	0	0	0	0
						⋮					
V	45	598	1	117	43	⇒	1	0	0	0	0

https://www.tensorflow.org/text/tutorials/word2vec

Word2Vec Vectors are Weights of a NN



https://medium.com/@manansuri/a-dummys-guide-to-word2vec-456444f3c673

FastText

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “**Enriching Word Vectors with Subword Information**,” *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051.

Main idea: learn **character n-gram embeddings** for the target word (not context) and modify the word2vec model to use these

Pre-trained models in 150+ languages

- https://fasttext.cc

FastText Details

Main idea: learn **character n-gram embeddings** and for the target word (not the context) modify the word2vec model to use these

Original word2vec:

$$p(c\ |w) \propto \exp(h_c^T v_w)$$

FastText:

$$p(c\ |w) \propto \exp\left(h_c^T\left(\sum\nolimits_{\text{n}-\text{gram } g \text{ in } w} z_g\right)\right)$$

FastText Details

Main idea: learn **character n-gram embeddings** and for the target word (not the context) modify the word2vec model to use these

$$p(c \,|w) \propto \exp\left(h_c^T \left(\sum\nolimits_{\mathrm{n-gram}\ g\ \mathrm{in}\ w} z_g\right)\right)$$

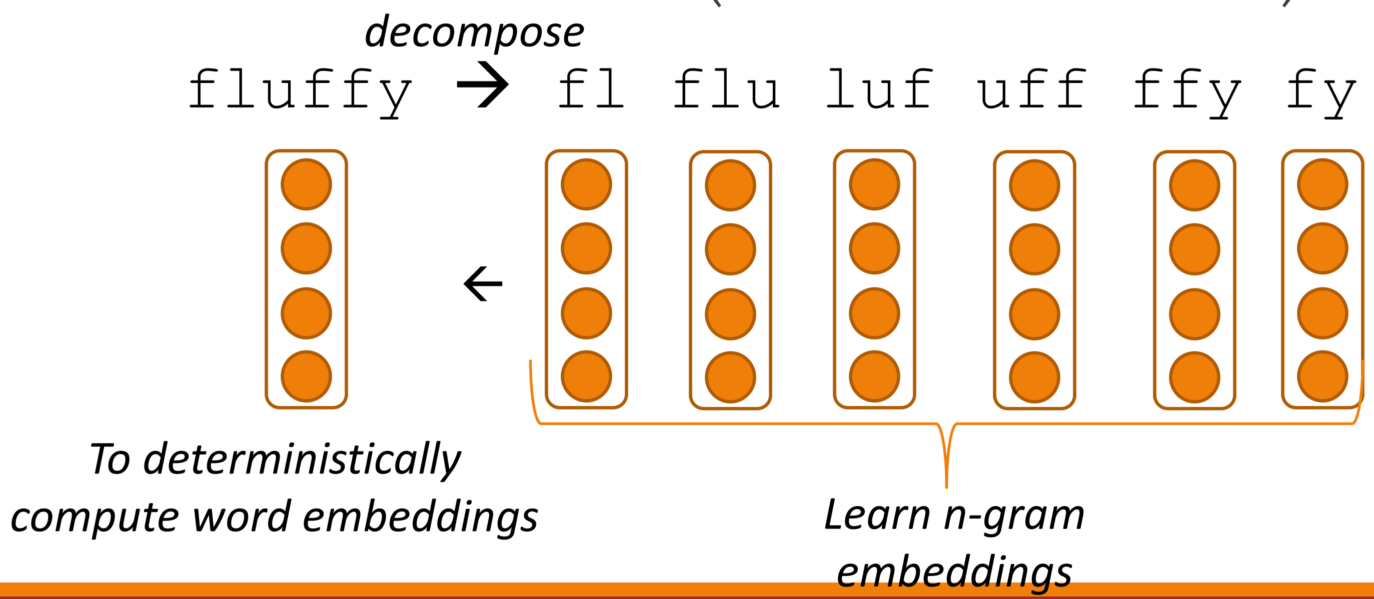
$$\texttt{fluffy} \xrightarrow{\textit{decompose}} \texttt{fl flu luf uff ffy fy}$$

Sub-word units like this have become an important part of today's NLP work!

FastText Details

Main idea: learn **character n-gram embeddings** and for the target word (not the context) modify the word2vec model to use these

$$p(c\ |w) \propto \exp\left(h_c^T\left(\sum_{\text{n−gram } g \text{ in } w} z_g\right)\right)$$



decompose
fluffy → fl flu luf uff ffy fy
←
To deterministically compute word embeddings
Learn n-gram embeddings

Contextual Word Embeddings

Word2vec-based models are not context-dependent

- Single word type → single word embedding

If a single word type can have different meanings…

- bank, bass, plant,…

… why should we only have one embedding?

Entire task devoted to classifying these meanings:
Word Sense Disambiguation

Contextual Word Embeddings

Growing interest in this

Off-the-shelf is a bit more difficult

- Download and run a model
- Can’t just download a file of embeddings

Two to know about (with code):

- ELMo: “Deep contextualized word representations” Peters et al. (2018; NAACL)
- https://allennlp.org/elmo
- BERT: “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” Devlin et al. (2019; NAACL)
  - https://github.com/google-research/bert

Evaluating Vector Embeddings

Cosine: Measuring Similarity

Given 2 target words v and w how similar are their vectors?

Dot product or inner product from linear algebra

$$\text{dot-product}(\vec{v},\vec{w}) = \vec{v}\cdot\vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + ... + v_N w_N$$

- High when two vectors have large values in same dimensions, low for orthogonal vectors with zeros in complementary distribution

Correct for high magnitude vectors $\dfrac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|}$

Cosine Similarity

Divide the dot product by the length of the two vectors

$$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$$

This is the cosine of the angle between them

$$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos\theta$$

$$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} = \cos\theta$$

Cosine Similarity

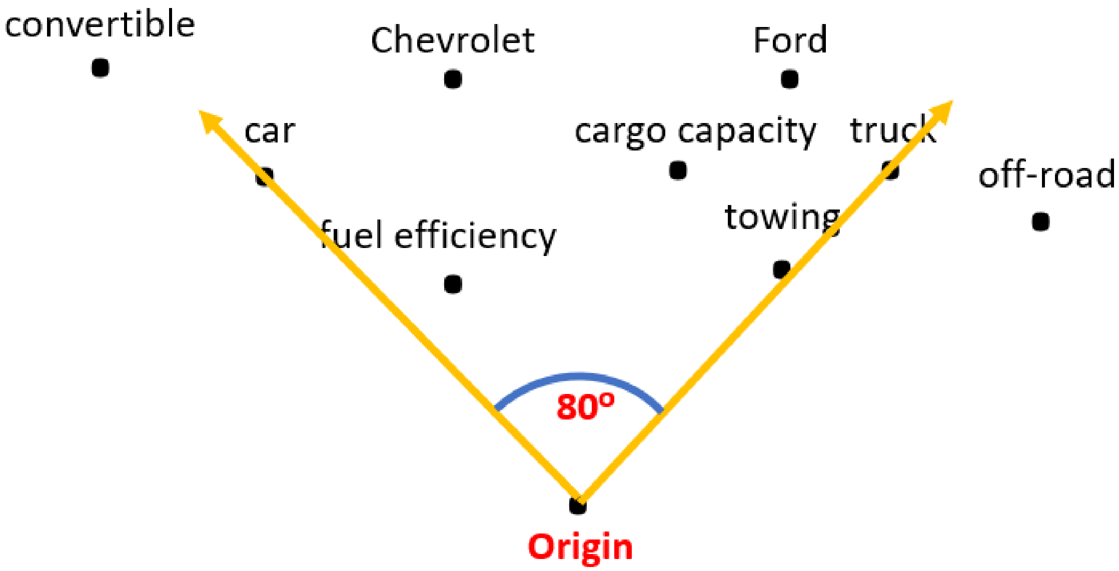
Divide the dot product by the length of the two vectors

$$\frac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|}$$

This is the cosine of the angle between them

$$\vec{a}\cdot\vec{b} = |\vec{a}||\vec{b}|\cos\theta$$

$$\frac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|} = \cos\theta$$



https://upload.wikimedia.org/wikipedia/commons/2/23/CosineSimilarity.png

Example: Word Similarity

$$\cos(x, y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

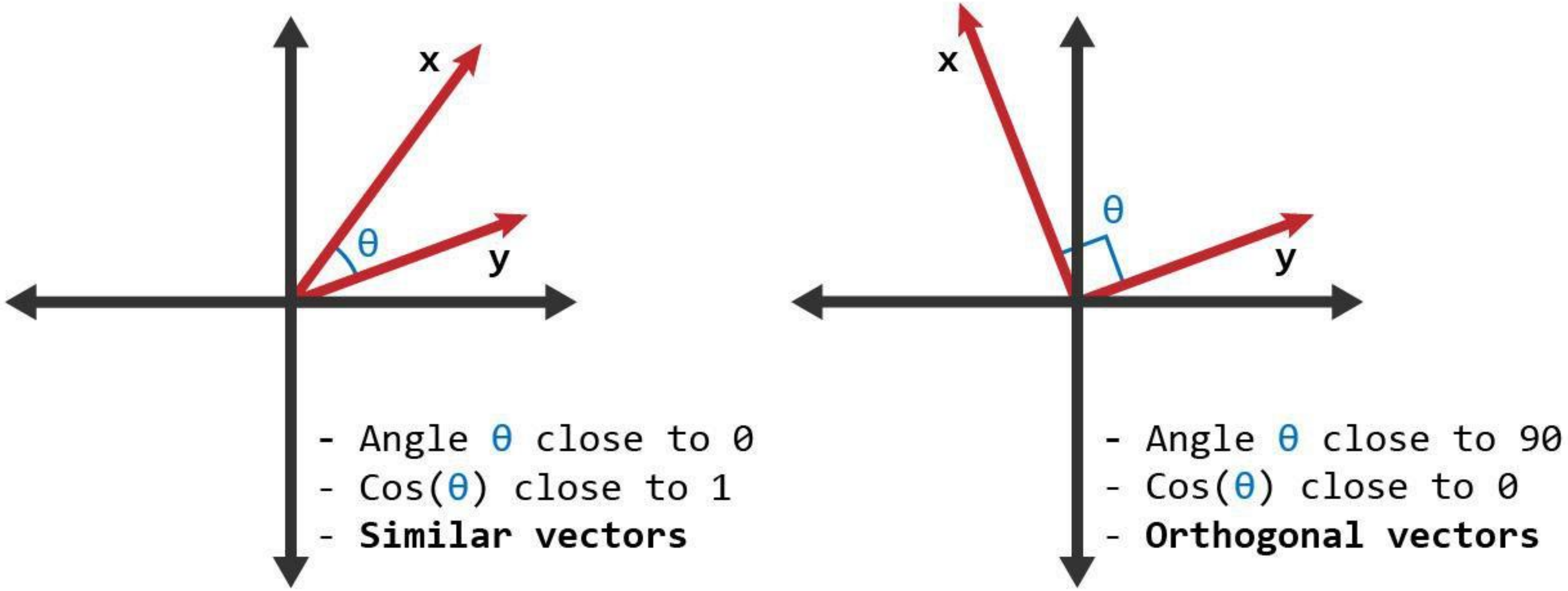
	Dim. 1	Dim. 2	Dim. 3
apricot	2	0	0
digital	0	1	2
information	1	6	1

cosine(apricot,information) = $\frac{2+0+0}{\sqrt{4+0+0}\sqrt{1+36+1}} = 0.1622$

cosine(digital,information) = $\frac{0+6+2}{\sqrt{0+1+4}\sqrt{1+36+1}} = 0.5804$

cosine(apricot,digital) = $\frac{0+0+0}{\sqrt{4+0+0}\sqrt{0+1+4}} = 0.0$

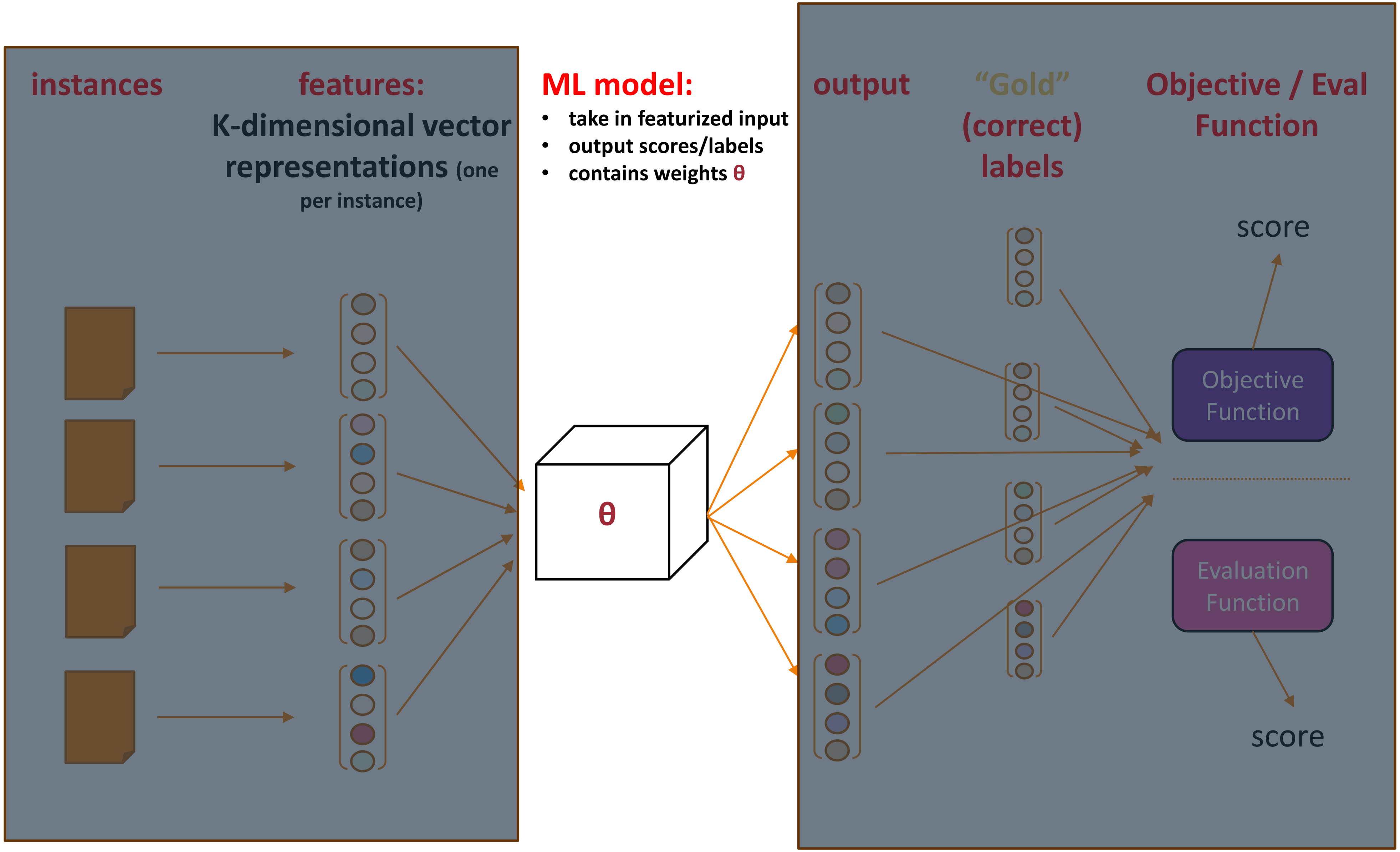
Cosine Similarity Range



- Angle θ close to 0
- Cos(θ) close to 1
- **Similar vectors**

- Angle θ close to 90
- Cos(θ) close to 0
- **Orthogonal vectors**

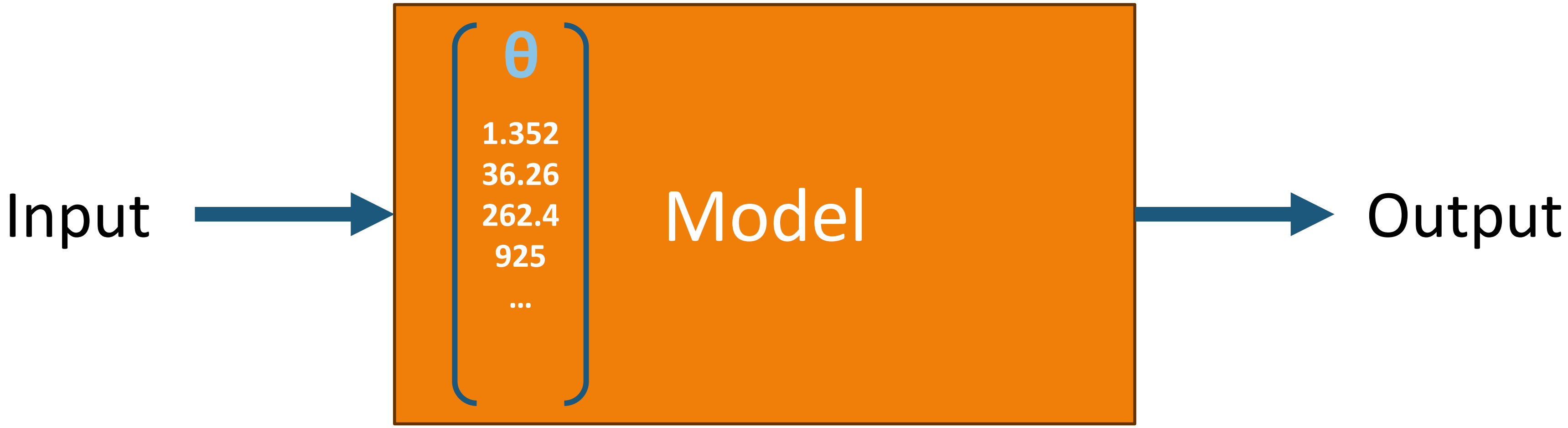
https://www.learndatasci.com/glossary/cosine-similarity/


instances
features:
K-dimensional vector representations (one per instance)
ML model:
take in featurized input
output scores/labels
contains weights θ
θ
output
“Gold” (correct) labels
Objective / Eval Function
score
Objective Function
Evaluation Function
score

Helpful ML Terminology

Model: the (computable) way to go from **features** (input) to labels/scores (output)

Weights/parameters (θ): vectors of numbers that control how the model produces labels/scores from inputs. These are learned through **training**.



Types of models

CLASSIFICATION

Model outputs comes from a finite set of values

Discrete result

Examples:

- What type of animal is this a picture of?
- Predicting the weather (sunny, cloudy, or rainy?)
- Ranking: Is this result *better* than this result?

REGRESSION

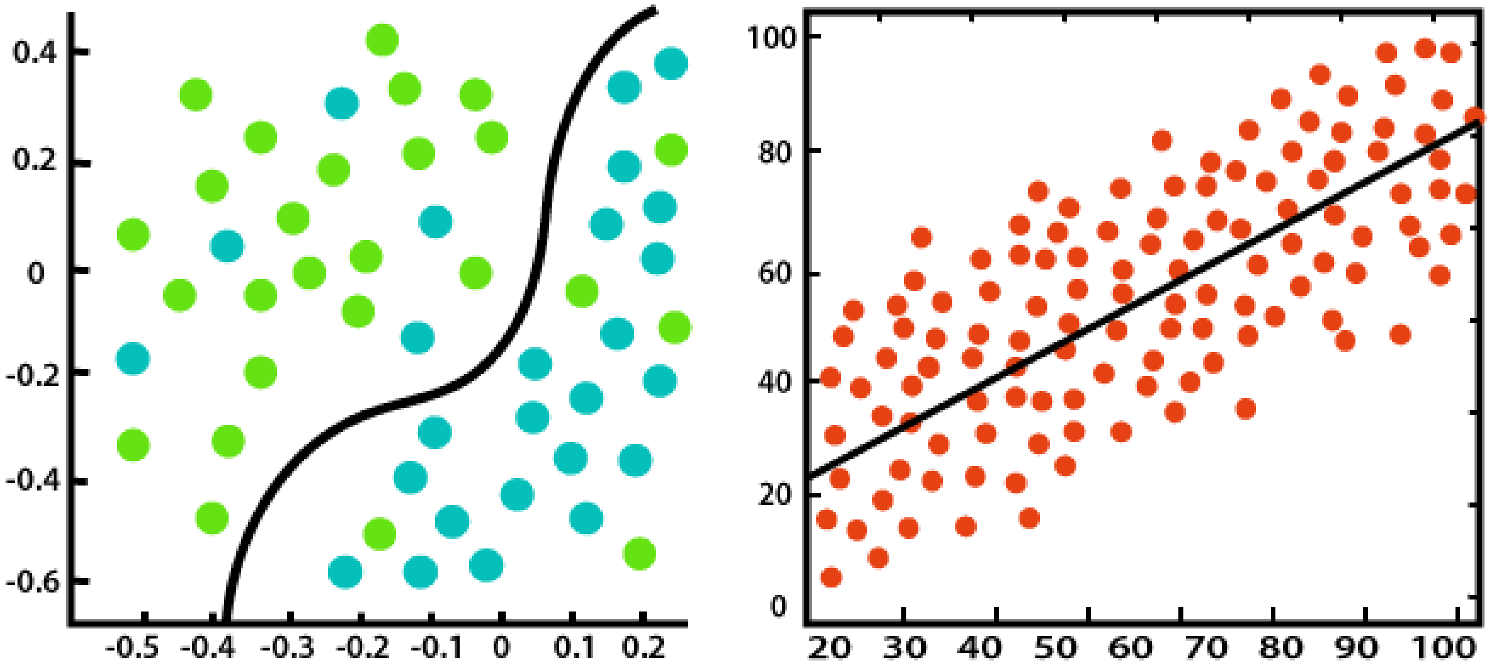
Model outputs are continuous values

Continuous result

Examples:

- How far will I move if I drive my motors at this speed for 1 second?
- Predicting the weather (temperature)
- Ranking: *how good* is this result?

Types of models



Classification

Regression

https://medium.com/unpackai/classification-regression-in-machine-learning-7cf3b13b0b09

What are some other examples of these?

CLASSIFICATION

Tone tagging

Sentiment classification

Named entity recognition

REGRESSION

Quantity/scale of how much it sounds like a specific author

Numerical sentiment value

Political “score” from document

Likelihoods

Predicted Goodreads score

Classification

Modeling

Classification/ Text Processing

$$P(y \mid x)$$

Language Model (LM) / Generation

$$P(w_t | w_{t-1}, w_{t-2} \cdots)$$

A language model is used to **generate** the next word(s) given a history of words.

Classification Types (Terminology)

Name	Number of Tasks (Domains) Labels are Associated with	# Label Types	Example
(Binary) Classification	1	2	Sentiment: Choose one of {positive or negative}
Multi-class Classification	1	> 2	Part-of-speech: Choose one of {Noun, Verb, Det, Prep, …}
Multi-label Classification	1	> 2	Sentiment: Choose multiple of {positive, angry, sad, excited, …}
Multi-task Classification	> 1	Per task: 2 or > 2 (can apply to binary or multi-class)	Task 1: part-of-speech Task 2: named entity tagging … ---------------------- Task 1: document labeling Task 2: sentiment

Maxent Models for Classification: Discriminatively or …

Directly model the posterior

$$p(Y \mid X) = \mathbf{maxent}(X; Y)$$

Discriminatively trained classifier

"Discriminative classifiers like logistic regression instead learn what features from the input are most useful to discriminate between the different possible classes."

SLP, ch. 4

Bayes' Rule

$$\underbrace{P(Y|X)}_{\text{Posterior}} = \frac{\overbrace{P(X|Y)}^{\text{Likelihood}}\cdot\overbrace{P(Y)}^{\text{Prior}}}{P(X)}$$

Posterior:
probability of event Y with knowledge that X has occurred
NLP pg. 478

Likelihood:
probability of event X given that Y has occurred
NLP pg. 478

Prior:
probability of event X occurring (regardless of what other events happen)
NLP pg. 478

Bayes’ Rule

$$P(c|d) = \frac{P(d|c)\cdot P(c)}{P(d)}$$

$$P\left(\textsc{Entailed}\ \middle|\ \boxed{\begin{array}{l}\text{s: Michael Jordan, coach Phil Jackson and the star}\\ \text{cast, including Scottie Pippen, took the Chicago}\\ \text{Bulls to six National Basketball Association}\\ \text{championships.}\\ \text{h: The Bulls basketball team is based in Chicago.}\end{array}}\right)$$

$$= \frac{P\left(\boxed{\begin{array}{l}\text{s: Michael Jordan, coach Phil Jackson and the star}\\ \text{cast, including Scottie Pippen, took the Chicago}\\ \text{Bulls to six National Basketball Association}\\ \text{championships.}\\ \text{h: The Bulls basketball team is based in Chicago.}\end{array}}\ \middle|\ \textsc{Entailed}\right)\cdot P\left(\textsc{Entailed}\right)}{P\left(\boxed{\begin{array}{l}\text{s: Michael Jordan, coach Phil Jackson and the star}\\ \text{cast, including Scottie Pippen, took the Chicago}\\ \text{Bulls to six National Basketball Association}\\ \text{championships.}\\ \text{h: The Bulls basketball team is based in Chicago.}\end{array}}\right)}$$

Maxent Models for Classification: Discriminatively or Generatively Trained

Directly model the posterior

$$p(Y \mid X) = \mathbf{maxent}(X; Y)$$

Discriminatively trained classifier

Model the posterior with Bayes rule

$$p(Y \mid X) \propto p(X \mid Y) p(Y)$$

Generatively trained classifier with maxent-based language model

Maximum Entropy (Log-linear) Models For Discriminatively Trained Classifiers

$$p(y \mid x) = \text{maxent}(\underbrace{x, y})$$

Modeled jointly!

Core Aspects to Maxent Classifier p(y|x)

We need to define:

- **features** $f(x)$ from x that are meaningful;
- **weights** θ (at least one per feature, often one per feature/label combination) to say how important each feature is; and
- a way to **form probabilities** from f and θ

Example: Document Classification via Bag-of-Words Features

Amazon acquired MGM in 2022, taking over a sprawling library that includes more than 4,000 feature films and 17,000 television shows. The tech behemoth also earned the rights to distribute all the Bond movies, but the new deal solidifies the company's oversight of Bond's big-screen future.

Tech

Not Tech

w: weights

feature	weight
Amazon	.43
acquired	0.025
behemoth	0.008
Bond	-0.0001
…	

f(**x**): “bag of words”

feature $f_i(x)$	value
Amazon	1
acquired	1
behemoth	1
Bond	2
…	
sniffle	0
…	

Adapted from https://www.nbcnews.com/pop-culture/movies/amazon-taking-control-james-bond-movie-franchise-rcna192967

Maxent Modeling

p(Tech |

Amazon acquired MGM in 2022, taking over a sprawling library that includes more than 4,000 feature films and 17,000 television shows. The tech behemoth also earned the rights to distribute all the Bond movies, but the new deal solidifies the company's oversight of Bond's big-screen future.

) ∝

exp(weight$_{1, \text{Tech}}$ * applies$_1$(📄) +

weight$_{2, \text{Tech}}$ * applies$_2$(📄) +

weight$_{3, \text{Tech}}$ * applies$_3$(📄) +

…

))

K different weights…

for K different features

multiplied and then summed

Maxent Modeling

$p(\text{TECH} \mid$

Amazon acquired MGM in 2022, taking over a sprawling library that includes more than 4,000 feature films and 17,000 television shows. The tech behemoth also earned the rights to distribute all the Bond movies, but the new deal solidifies the company's oversight of Bond's big-screen future.

$) \propto$

$$\left[.31\ \ -.5\ \ .1\ \ .002\ \ .522\ \ldots\right] \times \begin{bmatrix}1\\1\\1\\2\\0\\\ldots\end{bmatrix}$$

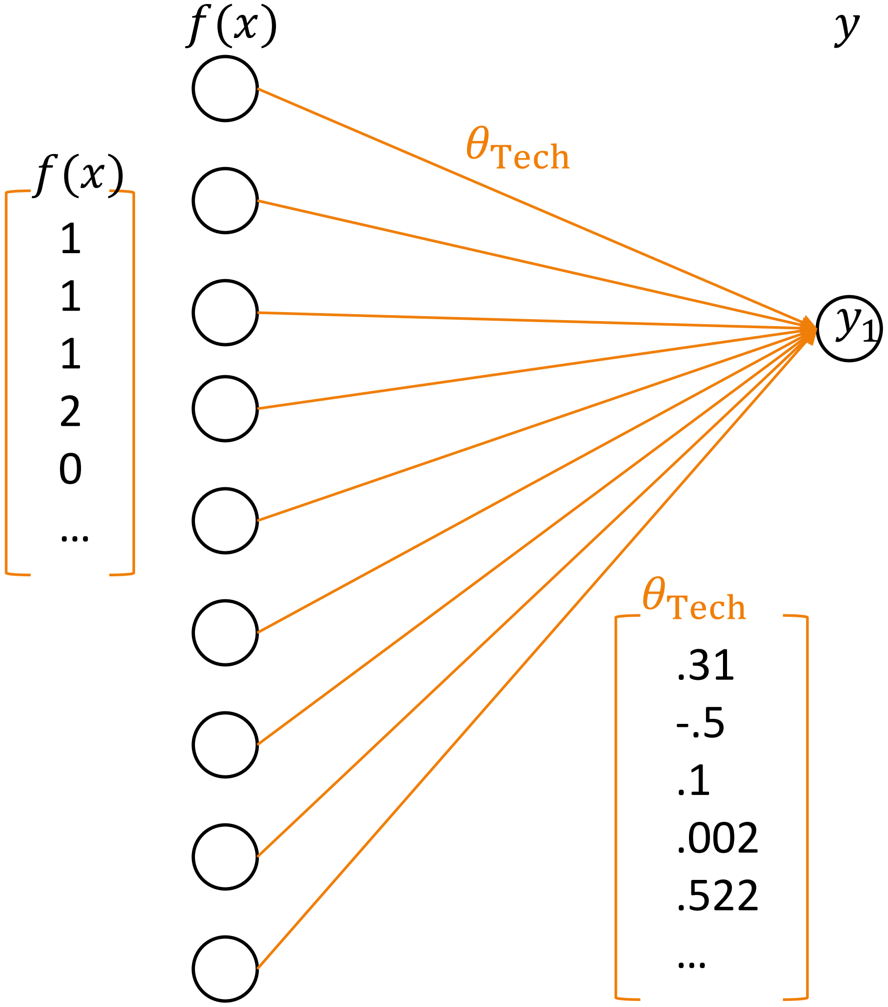
$\exp(\theta^{T}_{\text{TECH}} f($📄$))$

K different weights…

for K different features

multiplied and then summed

Maxent Classifier, schematically



$$f(x) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 0 \\ \dots \end{bmatrix}$$

$$\theta_{\text{Tech}} = \begin{bmatrix} .31 \\ -.5 \\ .1 \\ .002 \\ .522 \\ \dots \end{bmatrix}$$

$$p(y = \text{Tech}|x) \propto \exp(\theta_{\text{Tech}} f(x))$$

Maxent Modeling

$p($ TECH $\mid$

> Amazon acquired MGM in 2022, taking over a sprawling library that includes more than 4,000 feature films and 17,000 television shows. The tech behemoth also earned the rights to distribute all the Bond movies, but the new deal solidifies the company's oversight of Bond's big-screen future.

$) \propto$

$\frac{1}{Z}\exp($ $\theta_{\text{TECH}}^{T} f($📄$)$ $)$

Normalization for Classification

$$Z = \sum_{\text{label } j} \exp(\theta_j^T f(\text{🗎}))$$

$p(y \mid x) \propto \exp(\theta_y^T f(x))$

classify doc x with label y in one go

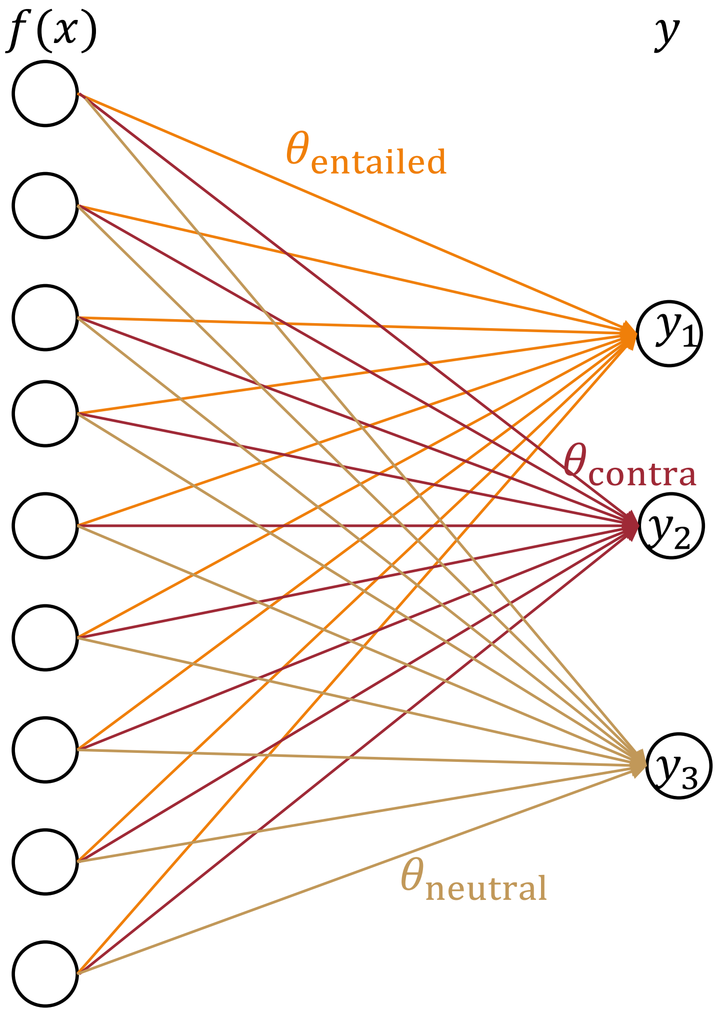
Normalization for Classification (long form)

$$Z = \sum_{\text{label } j} \exp\left(\begin{array}{l} \text{weight}_{1,\,j} * \text{applies}_1(\text{🗎}) \; + \\ \text{weight}_{2,\,j} * \text{applies}_2(\text{🗎}) \; + \\ \text{weight}_{3,\,j} * \text{applies}_3(\text{🗎}) \; + \\ \qquad \ldots \end{array} \right)$$

$$p(y \mid x) \propto \exp(\theta_y^T f(x))$$

classify doc x with label y in one go

Multi-label Maxent Classifier, schematically



$p(y = \text{entailed}|x) \propto$
$\exp(\theta_{\text{entailed}} f(x))$

$p(y = \text{contra}|x) \propto$
$\exp(\theta_{\text{contra}} f(x))$

$p(y = \text{neutral}|x) \propto$
$\exp(\theta_{\text{neutral}} f(x))$

output:
i = argmax score_i
class *i*

Final Equation for Logistic Regression

$$p(y \mid x) = \frac{\exp(\theta_y^T f(x))}{\sum_{y'} \exp(\theta_{y'}^T f(x))}$$

$$p(Y \mid x) = \mathrm{softmax}(\theta f(x))$$

Maxent Models for Classification: Discriminatively or Generatively Trained

Directly model the posterior

$$p(Y \mid X) = \mathbf{maxent}(X; Y)$$

Discriminatively trained classifier

Model the posterior with Bayes rule

$$p(Y \mid X) \propto p(X \mid Y)p(Y)$$

Generatively trained classifier with maxent-based language model

Bayes’ Rule

$$\underbrace{P(Y|X)}_{\text{Posterior}} = \frac{\overbrace{P(X|Y)}^{\text{Likelihood}} \cdot \overbrace{P(Y)}^{\text{Prior}}}{P(X)}$$

It’s harder to model P(Y|X) directly since it might be that we only see that set of features once!

Bayes’ Rule → Naïve Bayes Assumption

Bayes

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c) \cdot P(c)}{P(d)}$$

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c) \cdot P(c)}{\cancel{P(d)}}$$

We can make this assumption because P(d) stays the same regardless of the class!

Naïve Bayes

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, P(c|d) \approx \underset{c \in C}{\operatorname{argmax}}\, P(d|c) \cdot P(c)$$

Bayes' Rule → Naïve Bayes Assumption

Bayes

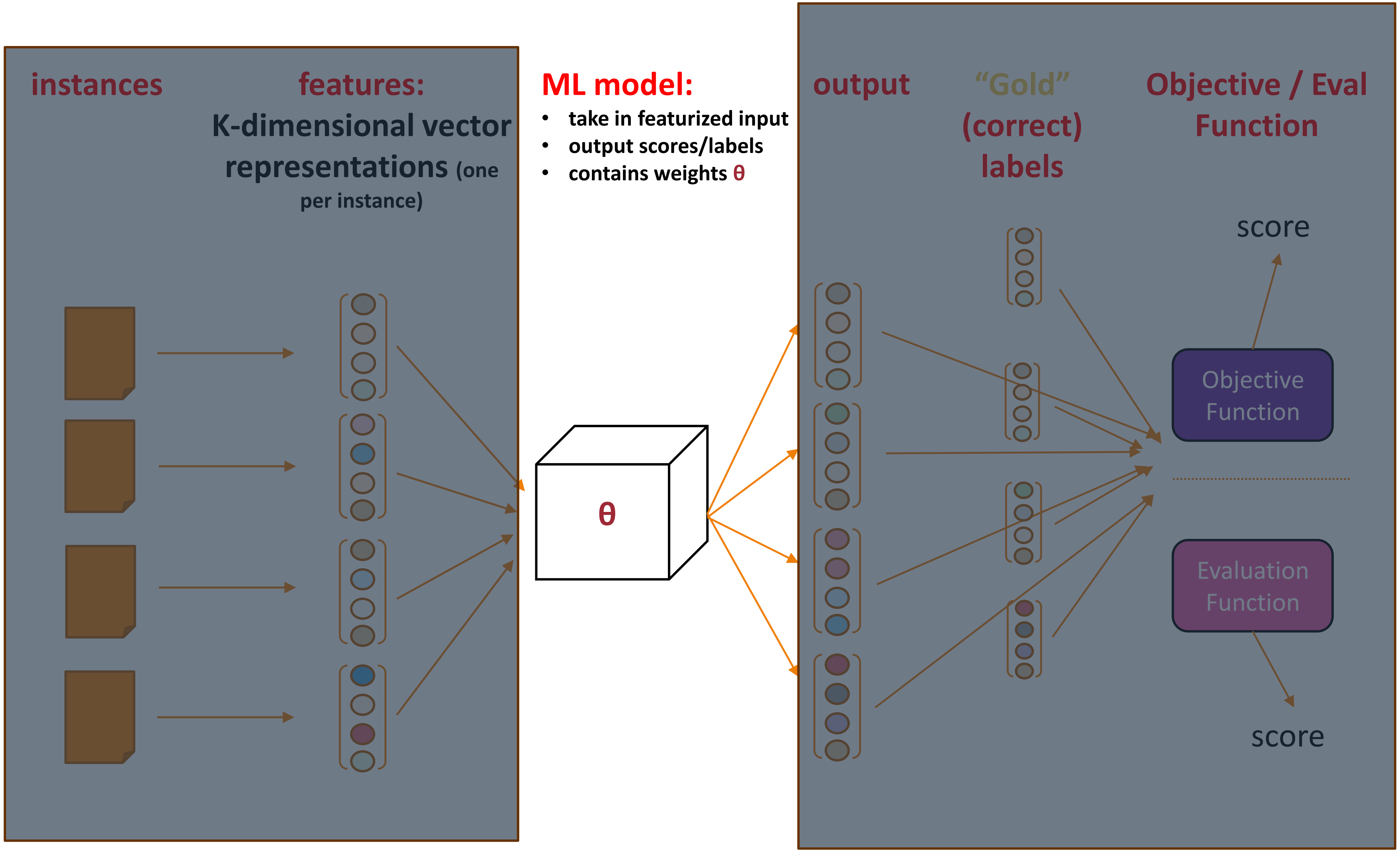
$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c) \cdot P(c)}{P(d)}$$

Naïve Bayes

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) \approx \underset{c \in C}{\operatorname{argmax}} P(d|c) \cdot P(c)$$

Naïve bayes is **generative** because we are sort of assuming this is how the data point is generated: pick a class *c* and then generate the words by sampling from P(d|c)

SLP 4.1



ML model:

- take in featurized input
- output scores/labels
- contains weights θ

Modeling

Classification/
Text Processing

$$P(y \mid x)$$

Language
Model (LM) /
Generation

$$P(w_t | w_{t-1}, w_{t-2} \cdots)$$

A language model is used to **generate** the next word(s) given a history of words.

Language Models

Maximum likelihood (MLE): simple counting

Other count-based models

- E.g., Laplace smoothing (add-1, add-λ)

Maxent n-gram models ← Featureful LMs

Neural n-gram models ← Feedforward LMs

Recurrent/autoregressive NNs ← Precursor to modern LMs

Transformers ← Modern LMs

Language Models

Maximum likelihood (MLE): simple counting

Other count-based models

- E.g., Laplace smoothing (add-1, add-λ)

Maxent n-gram models ← Featureful LMs

Neural n-gram models ← Feedforward LMs

Recurrent/autoregressive NNs ← Precursor to modern LMs

Transformers ← Modern LMs

Key Idea: Probability Chain Rule

$$p(w_1, w_2, \ldots, w_S) =$$
$$p(w_1)p(w_2\ |w_1)p(w_3\ |\ w_1, w_2) \cdots p(w_S|w_1, \ldots, w_{S-1}) =$$
$$\prod_i^S p(w_i|w_1, \ldots, w_{i-1})$$

N-Grams

Maintaining an entire *joint* inventory over sentences could be too much to ask

Store “smaller” pieces?

p(Colorless green ideas sleep furiously) =
*p(Colorless) **
*p(green | Colorless) **
*p(ideas | Colorless green) **
*p(sleep | Colorless green ideas) **
p(furiously | Colorless green ideas sleep)

N-Grams

p(furiously | Colorless green ideas sleep)

How much does “Colorless” influence the choice of “furiously?”

Remove history and contextual info

p(furiously | Colorless green ideas sleep) ≈

p(furiously | ideas sleep)

N-Grams

p(Colorless green ideas sleep furiously) =
p(Colorless) *
p(green | Colorless) *
p(ideas | Colorless green) *
p(sleep | ~~Colorless~~ green ideas) *
p(furiously | ~~Colorless green~~ ideas sleep)

Trigrams

p(Colorless green ideas sleep furiously) =
p(Colorless) *
p(green | Colorless) *
p(ideas | Colorless green) *
p(sleep | green ideas) *
p(furiously | ideas sleep)

Trigrams

p(Colorless green ideas sleep furiously) =

p(Colorless) *

p(green | Colorless) *

p(ideas | Colorless green) *

p(sleep | green ideas) *

p(furiously | ideas sleep)

Trigrams

p(Colorless green ideas sleep furiously) =
p(Colorless | *<BOS> <BOS>*) *
p(green | *<BOS>* Colorless) *
p(ideas | Colorless green) *
p(sleep | green ideas) *
p(furiously | ideas sleep)

Consistent notation: Pad the left with <BOS> (beginning of sentence) symbols

Trigrams

p(Colorless green ideas sleep furiously) =
p(Colorless | *<BOS> <BOS>*) *
p(green | *<BOS>* Colorless) *
p(ideas | Colorless green) *
p(sleep | green ideas) *
p(furiously | ideas sleep) *
p(*<EOS>* | sleep furiously)

Consistent notation: Pad the left with <BOS> (beginning of sentence) symbols
Fully proper distribution: Pad the right with a single <EOS> symbol

N-Gram Terminology

n	Commonly called	History Size (Markov order)	Example
1	unigram	0	p(furiously)
2	bigram	1	p(furiously | sleep)
3	trigram (3-gram)	2	p(furiously | ideas sleep)
4	4-gram	3	p(furiously | green ideas sleep)
n	n-gram	n-1	$p(w_i \mid w_{i-n+1} \ldots w_{i-1})$

Count-Based N-Grams (Unigrams)

$$p\left(\overset{\substack{\text{word type}\\ \downarrow}}{\mathrm{z}}\right) = \frac{count(\mathrm{z})}{\sum_v count\left(\underset{\substack{\uparrow\\ \text{word type}}}{\mathrm{v}}\right)}$$

Count-Based N-Grams (Unigrams)

word type

$$p(\mathrm{z}) \propto \frac{count(\mathrm{z})}{W}$$

number of tokens observed

Count-Based N-Grams (Unigrams)

The film got a great opening and the film went on to become a hit .

Word (Type) z	Raw Count count(z)	Normalization	Probability p(z)
The	1	16	1/16
film	2		1/8
got	1		1/16
a	2		1/8
great	1		1/16
opening	1		1/16
and	1		1/16
the	1		1/16
went	1		1/16
on	1		1/16
to	1		1/16
become	1		1/16
hit	1		1/16
.	1		1/16

Count-Based N-Grams (Trigrams)

order matters in conditioning

order matters in *count*

$$p(\mathrm{z}|\mathrm{x},\mathrm{y}) \propto count(\mathrm{x},\mathrm{y},\mathrm{z})$$

Count of the sequence of items “x y z”

Count-Based N-Grams (Trigrams)

order matters in conditioning

order matters in *count*

$$p(\mathrm{z}|\mathrm{x},\mathrm{y}) \propto count(\mathrm{x},\mathrm{y},\mathrm{z})$$

count(x, y, z) ≠ count(x, z, y) ≠ count(y, x, z) ≠ …

Count-Based N-Grams (Trigrams)

$$p(\mathrm{z}|\mathrm{x},\mathrm{y}) \propto count(\mathrm{x},\mathrm{y},\mathrm{z})$$

$$= \frac{count(\mathrm{x},\mathrm{y},\mathrm{z})}{\sum_{v} count(\mathrm{x},\mathrm{y},\mathrm{v})}$$

Count-Based N-Grams (Lowercased Trigrams)

the film got a great opening and the film went on to become a hit .

Context: x y	Word (Type): z	Raw Count	Normalization	Probability: p(z | x y)
the film	the	0	2	0/2
the film	film	0		0/2
the film	got	1		1/2
the film	went	1		1/2
…				
a great	great	0	1	0/1
a great	opening	1		1/1
a great	and	0		0/1
a great	the	0		0/1
…				

Language Models

Maximum likelihood (MLE): simple counting

Other count-based models

- E.g., **Laplace smoothing (add-1, add-λ)**

Maxent n-gram models ← Featureful LMs

Neural n-gram models ← Feedforward LMs

Recurrent/autoregressive NNs ← Precursor to modern LMs

Transformers ← Modern LMs

0s Are Not Your (Language Model's) Friend

$$p(\text{item}) \propto count(\text{item}) = 0 \rightarrow$$
$$p(\text{item}) = 0$$

0 probability → item is *impossible*

0s annihilate: x*y*z*0 = 0

Language is creative:

new words keep appearing

existing words could appear in known contexts

How much do you trust your data?

Add-λ estimation

Other names: Laplace smoothing, Lidstone smoothing

Pretend we saw each word λ more times than we did

$$p(\mathrm{z}) \propto count(\mathrm{z}) + \lambda$$

Add λ to all the counts

Add-λ estimation

Other names: Laplace smoothing, Lidstone smoothing

Pretend we saw each word λ more times than we did

Add λ to all the counts

$$p(\mathrm{z}) \propto count(\mathrm{z}) + \lambda$$

$$= \frac{count(\mathrm{z}) + \lambda}{\sum_v (count(\mathrm{v}) + \lambda)}$$

Add-λ estimation

Other names: Laplace smoothing, Lidstone smoothing

Pretend we saw each word λ more times than we did

Add λ to all the counts

$$p(\mathrm{z}) \propto count(\mathrm{z}) + \lambda$$

$$= \frac{count(\mathrm{z}) + \lambda}{\underbrace{W}_{\text{\# tokens}} + \underbrace{V}_{\text{\# types}}\lambda}$$

What are the tri-grams for “The film , a hit !”

Trigrams	MLE p(trigram)	UNK-ed trigrams	Smoothed p(trigram)
<BOS> <BOS> The	1	<BOS> <BOS> The	2/17
<BOS> The film	1	<BOS> The film	2/17
The film ,	0	The film <UNK>	1/17
film , a	0	film <UNK> a	1/16
, a hit	0	<UNK> a hit	1/16
a hit !	0	a hit <UNK>	1/17
hit ! <EOS>	0	hit <UNK> <EOS>	1/16

Language Models

Maximum likelihood (MLE): simple counting

Other count-based models

- E.g., Laplace smoothing (add-1, add-λ)

Maxent n-gram models ← Featureful LMs

Neural n-gram models ← Feedforward LMs

Recurrent/autoregressive NNs ← Precursor to modern LMs

Transformers ← Modern LMs

Text Generation
as *Classification Problem*?

I could eat so many delicious __________

I could eat so many juicy __________

Types	Probability
apples	.03
sandwiches	.02
pineapples	.004
houses	.00002
…	…

?

Maxent Models as Featureful n-gram Language Models

p(Colorless green ideas sleep furiously | Label) =
p(Colorless | Label, <BOS>) * … * p(<EOS> | Label , furiously)

Model each n-gram term with
a maxent model

$$p(x_i \mid y, x_{i-N+1:i-1}) =$$
$$\text{maxent}(y, x_{i-N+1:i-1}, x_i)$$

generatively trained:
learn to model *(class-specific) language*

Language Model with Maxent n-grams

$$p_n(\text{📄}|y) = \prod_{i=1}^{M} \text{maxent}(\overset{\substack{\text{label} \\ \downarrow}}{y}, \underbrace{x_{i-n+1:i-1}, x_i}_{\text{n-gram}})$$

$$= \prod_{i=1}^{M} \frac{\exp(\theta_{x_i}^{T} f(y, x_{i-n+1:i-1}))}{\sum_{x'} \exp(\theta_{x'}^{T} f(y, x_{i-n+1:i-1}))}$$

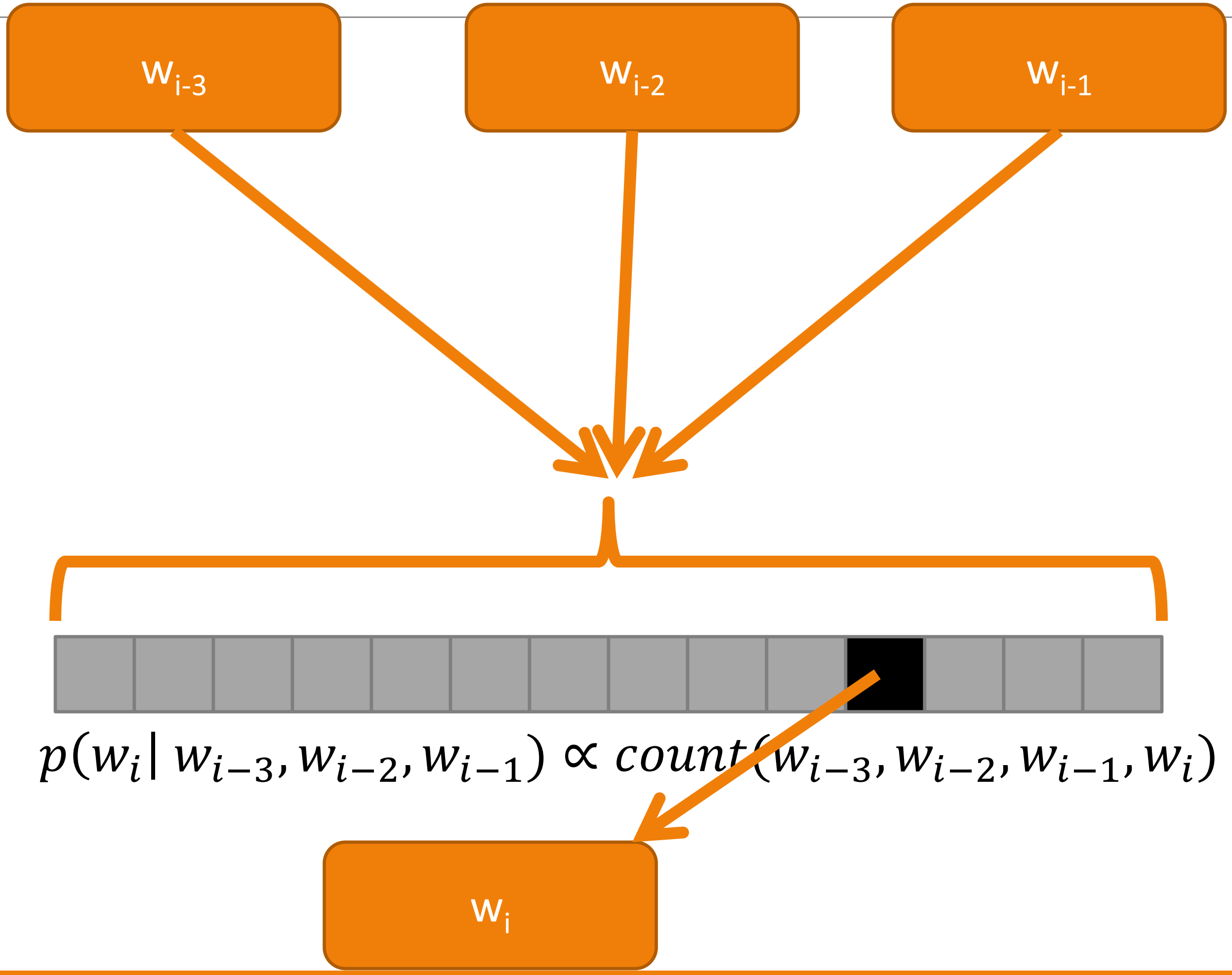
Iterate through all possible output vocab types x'---just like in count-based LMs

Count-based Language Models

given some context…

compute beliefs about what is likely…

predict the next word



$w_{i\text{-}3}$

$w_{i\text{-}2}$

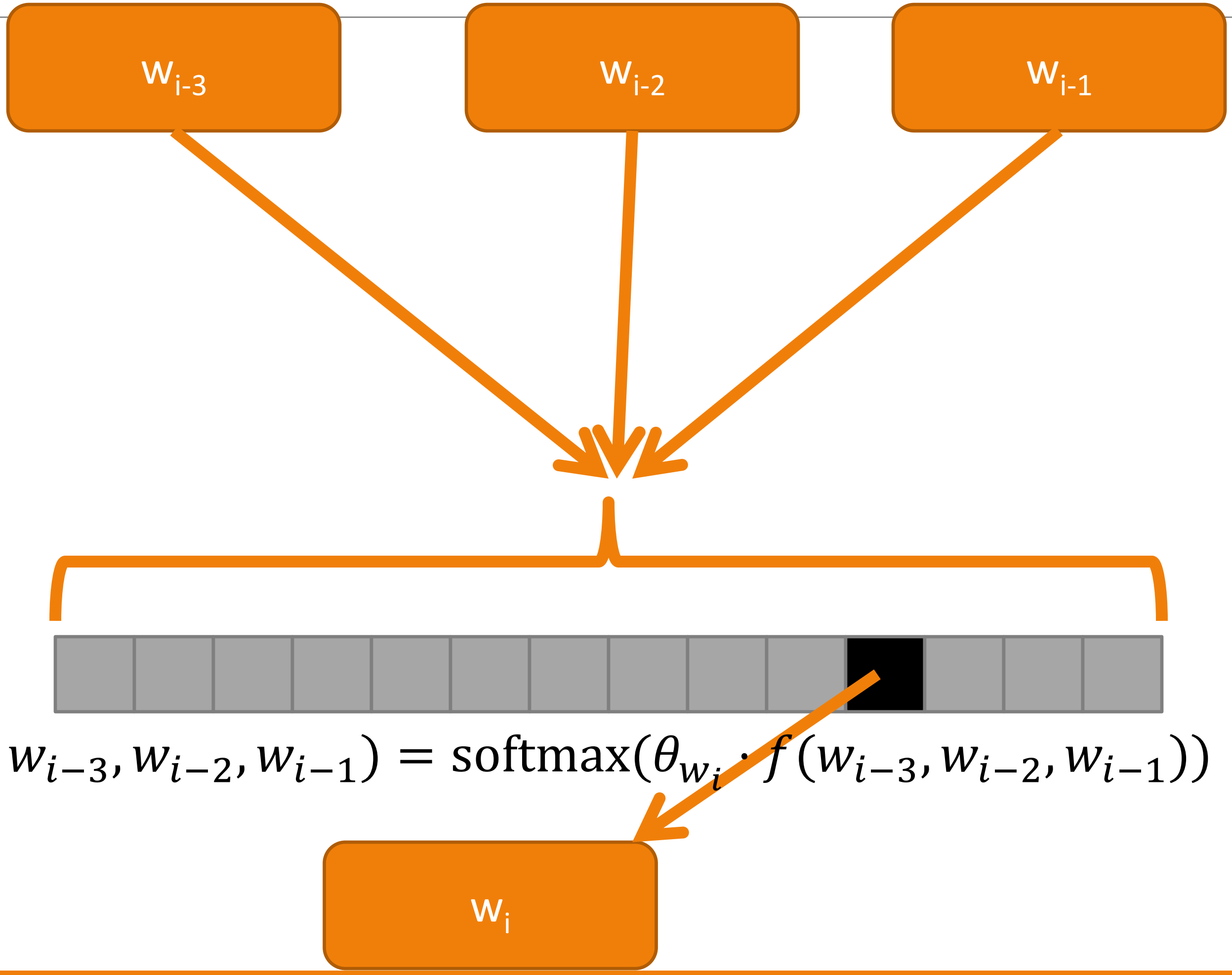
$w_{i\text{-}1}$

$$p(w_i|\, w_{i-3}, w_{i-2}, w_{i-1}) \propto count(w_{i-3}, w_{i-2}, w_{i-1}, w_i)$$

w_i

Maxent Language Models

given some context…



compute beliefs about what is likely…

$$p(w_i|\, w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))$$

predict the next word

ML/NLP Framework for Learning

instances

features: **K-dimensional vector representations** (one per instance)

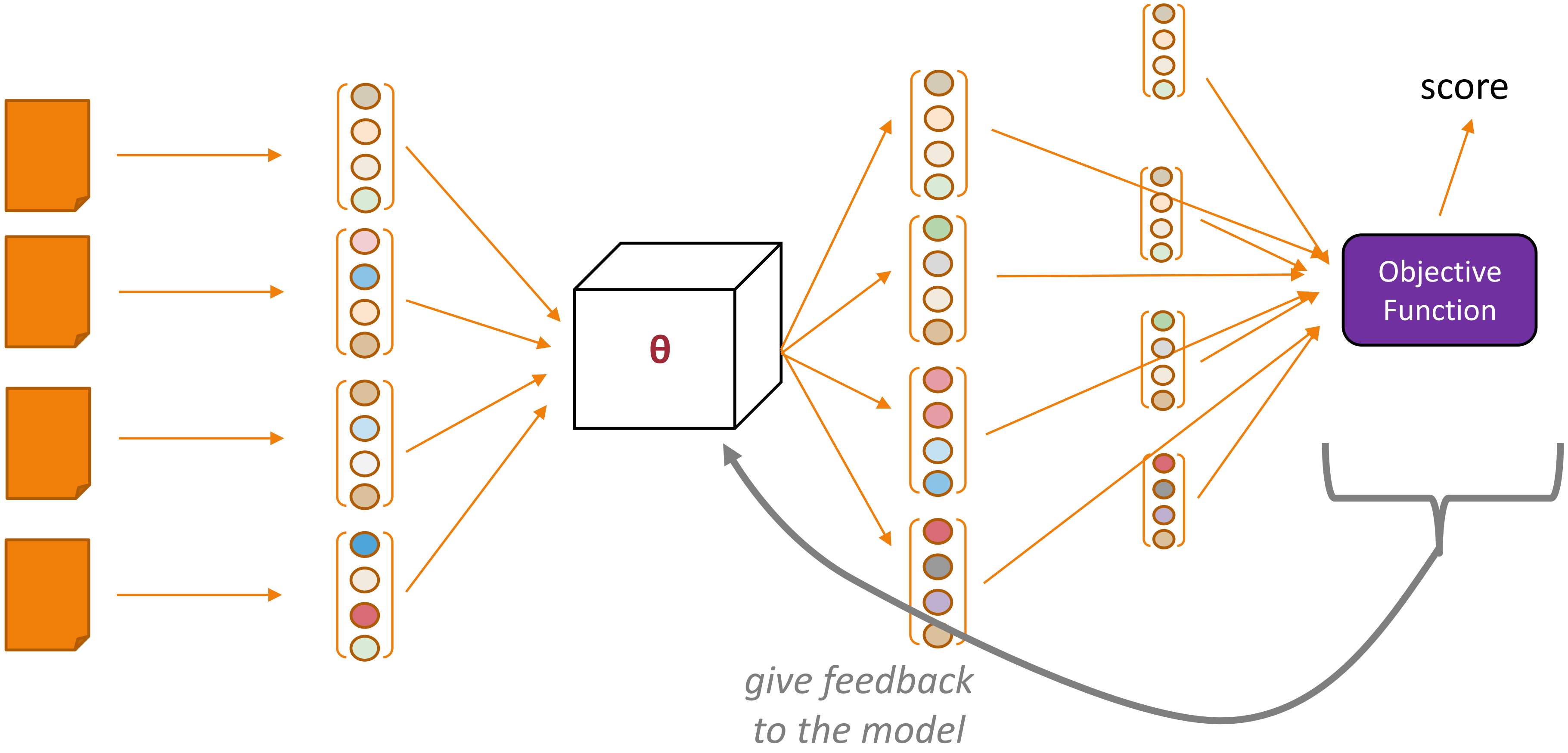
ML model:

- take in featurized input
- output scores/labels
- contains weights θ

output

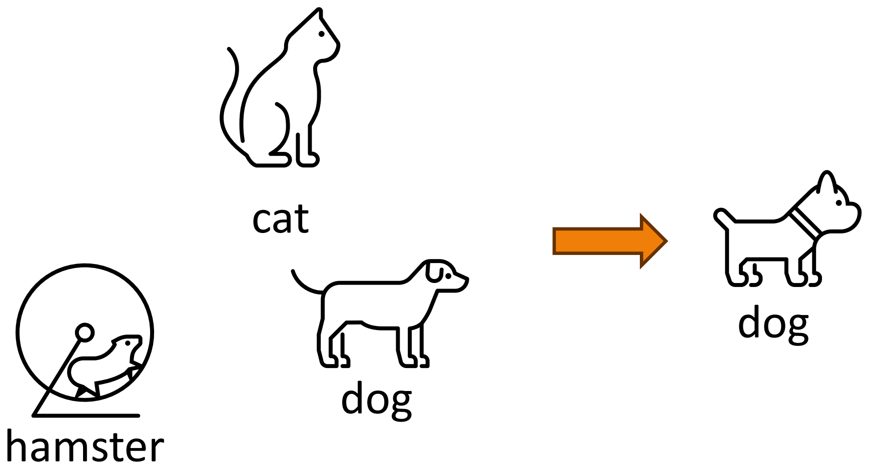
"Gold" (correct) labels

Objective Function

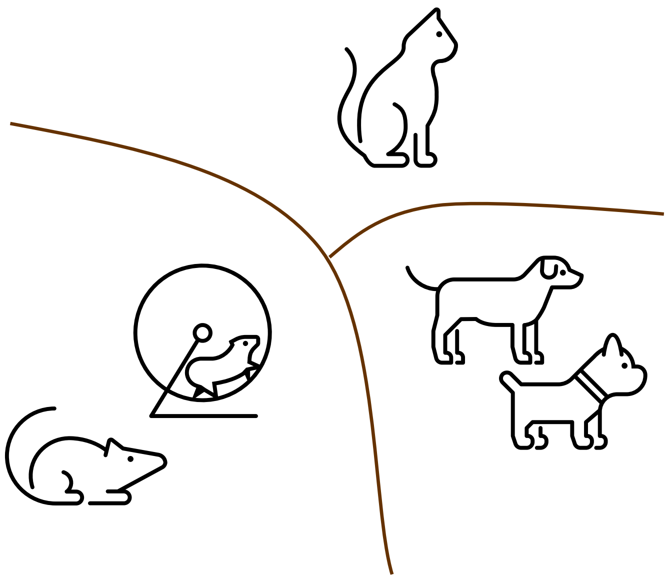


Types of Learning

SUPERVISED LEARNING



UNSUPERVISED LEARNING



Types of Learning

SUPERVISED LEARNING

Data has feedback (labels)

Data consists of input-output pairs

Learn mapping from input to output

Examples:

- Dataset classification
- How likely is it that this person will get into a car accident?

UNSUPERVISED LEARNING

No explicit feedback in data

Learn patterns directly from data

Examples:

- Clustering
- Do these people fall under multiple groups?

What are some other examples of these?

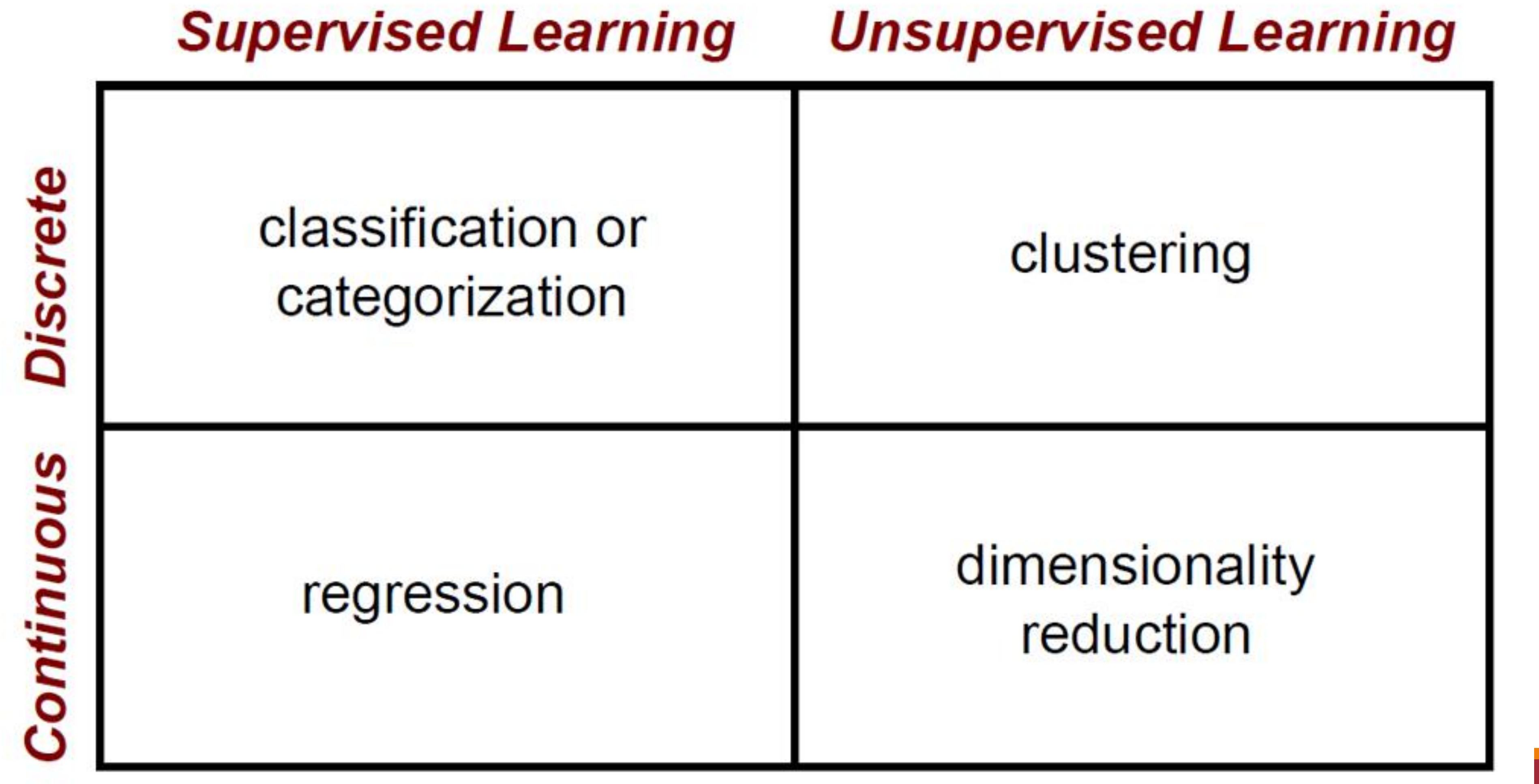
SUPERVISED LEARNING

- Machine translation
- Object segmentation (vision)
- Document classification

UNSUPERVISED LEARNING

- Clustering
- Language modeling

Types of Algorithms



	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

ML/NLP Framework for Learning

instances

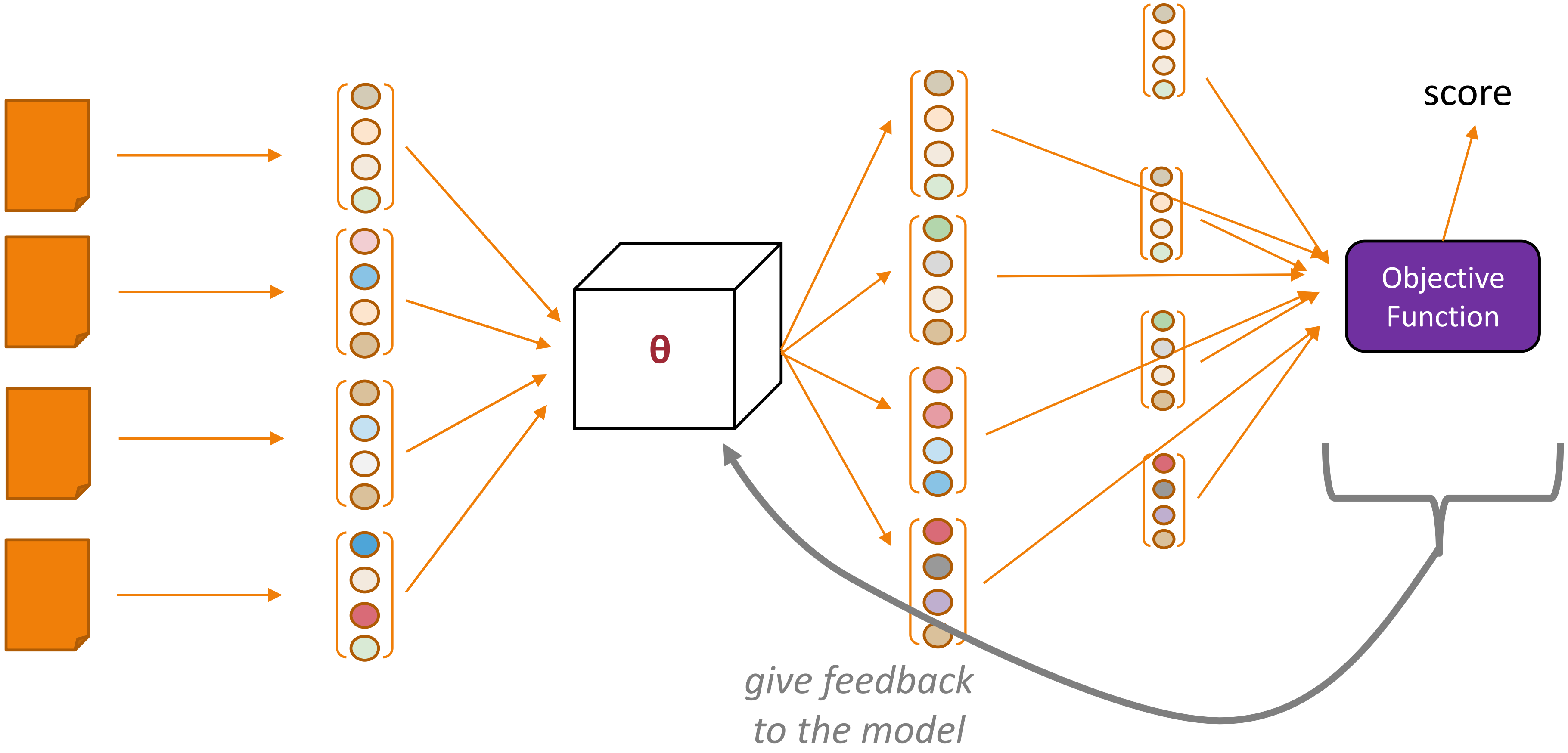
features:
K-dimensional vector representations (one per instance)

ML model:
- take in featurized input
- output scores/labels
- contains weights θ

output

"Gold" (correct) labels

Objective Function



give feedback to the model

$$p_\theta(y \mid x)$$ probabilistic model

$$\Downarrow$$

$$F(\theta; x, y)$$ **objective**

Primary Objective: Likelihood

Goal: *maximize* the score your model gives to the training data it observes

This is called the **likelihood of your data**

In classification, this is p(label | 🗎)

For language modeling, this is p(word | history of words)

Objective = Full Likelihood? (Classification)

$$\prod_i \overbrace{p_\theta(y_i|x_i)}^{\text{Our goal probability}} \propto \prod_i \overbrace{\exp(\theta_{y_i}^T f(x_i))}^{\text{Our maxent equation}}$$

These values can have very small magnitude ➔ underflow

Differentiating this product could be a pain

Logarithms

(0, 1] ➔ (-∞, 0]

Products ➔ Sums

log(ab) = log(a) + log(b)

log(a/b) = log(a) – log(b)

Inverse of exp

log(exp(x)) = x

How might you find the log of this?

$$\prod_{i} p_\theta(y_i | x_i)$$

Log-Likelihood (Classification)

Wide range of (negative) numbers
Sums are more stable

$$\log \prod_i p_\theta(y_i|x_i) = \sum_i \log p_\theta(y_i|x_i)$$

Products ➔ Sums

log(ab) = log(a) + log(b)

log(a/b) = log(a) – log(b)

Maximize Log-Likelihood (Classification)

Original maxent equation

$$\frac{\exp(\theta_y^T f(x))}{\sum_{y'} \exp(\theta_{y'}^T f(x))}$$

Differentiating this becomes nicer (even though Z depends on θ)

$$\log \prod_i p_\theta(y_i | x_i) = \sum_i \log p_\theta(y_i | x_i)$$

$$= \sum_i \overbrace{\theta_{y_i}^T f(x_i) - \log Z(x_i)}$$

Inverse of exp

log(exp(x)) = x

Log-Likelihood (Classification)

Wide range of (negative) numbers

Sums are more stable

$$\log \prod_i p_\theta(y_i|x_i) = \sum_i \log p_\theta(y_i|x_i)$$

$$= \sum_i \theta_{y_i}^T f(x_i) - \log Z(x_i)$$

$$= F(\theta)$$

Equivalent Version 2:
Minimize Cross Entropy Loss

Cross entropy:
How much $\hat{y}$ differs from the true y

loss uses y (random variable), or model's probabilities $\ell^{\text{xent}}\left(\overrightarrow{y^*}, p(y|x)\right)$

$$\ell^{\text{xent}}\left(\overrightarrow{y^*}, y\right) = -\sum_{k} \overrightarrow{y^*}[k] \log p(y = k|x)$$

$$\begin{pmatrix} 0 \\ 0 \\ \ldots \\ 1 \\ \ldots \\ 0 \end{pmatrix}$$

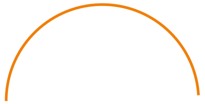
index of "1" indicates correct value

one-hot vector

Classification Log-likelihood ≅ Cross Entropy Loss

Log Likelihood

$$F(\theta) = \sum_i \theta_{y_i}^T f(x_i) - \log Z(x_i)$$



objective is concave

Cross Entropy Loss

$$\ell^{\text{xent}}\left(\overrightarrow{y^*}, y\right) = -\sum_k \overrightarrow{y^*}[k] \log p(y = k|x)$$

objective is convex

Preventing Extreme Values

Likelihood on its own can lead to overfitting and/or extreme values in the probability computation

$$F(\theta) = \sum_i \theta_{y_i}^T f(x_i) - \log Z(x_i)$$

Learn the parameters based on some (fixed) data/examples

Regularization: Preventing Extreme Values

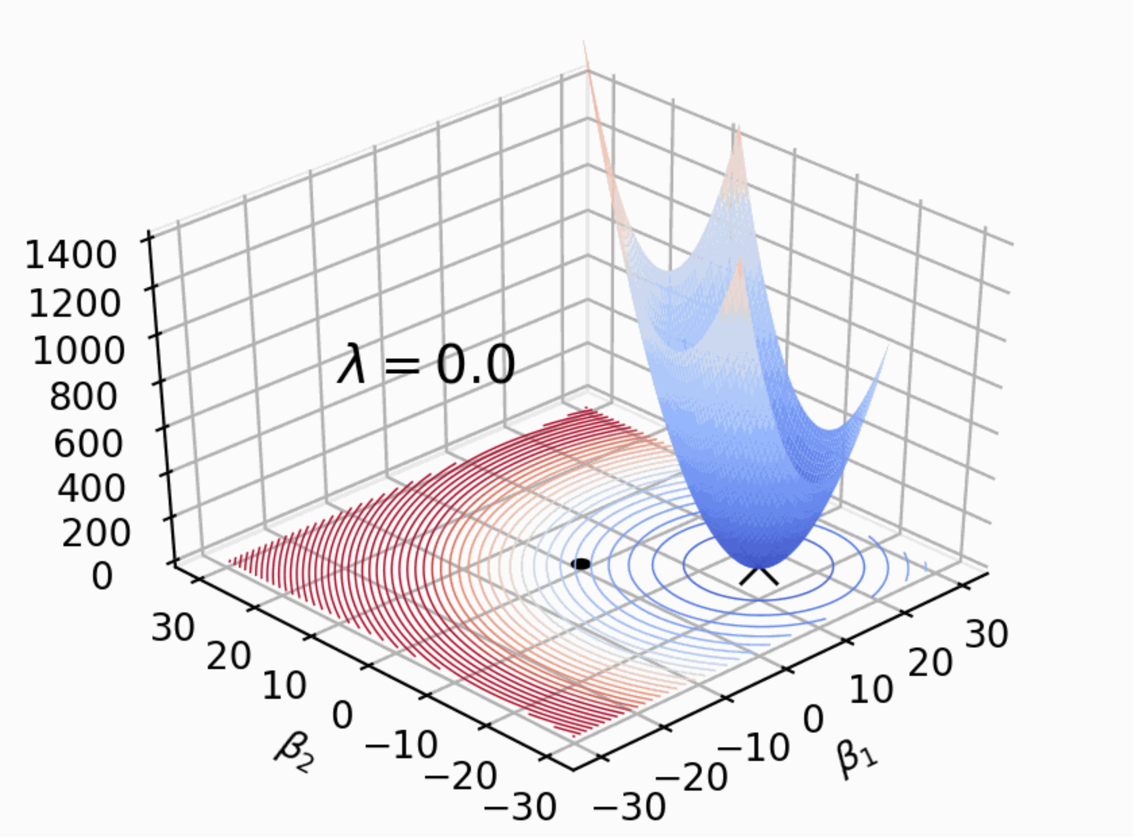
$$F(\theta) = \left(\sum_i \theta_{y_i}^T f(x_i) - \log Z(x_i) \right) - R(\theta)$$

With fixed/predefined features, the values of θ determine how "good" or "bad" our objective learning is

- Augment the objective with a **regularizer**
  - This regularizer places an inductive bias (or, prior) on the general "shape" and values of θ

(Squared) L2 Regularization

$$R(\theta) = \|\theta\|_2^2 = \sum_k \theta_k^2$$

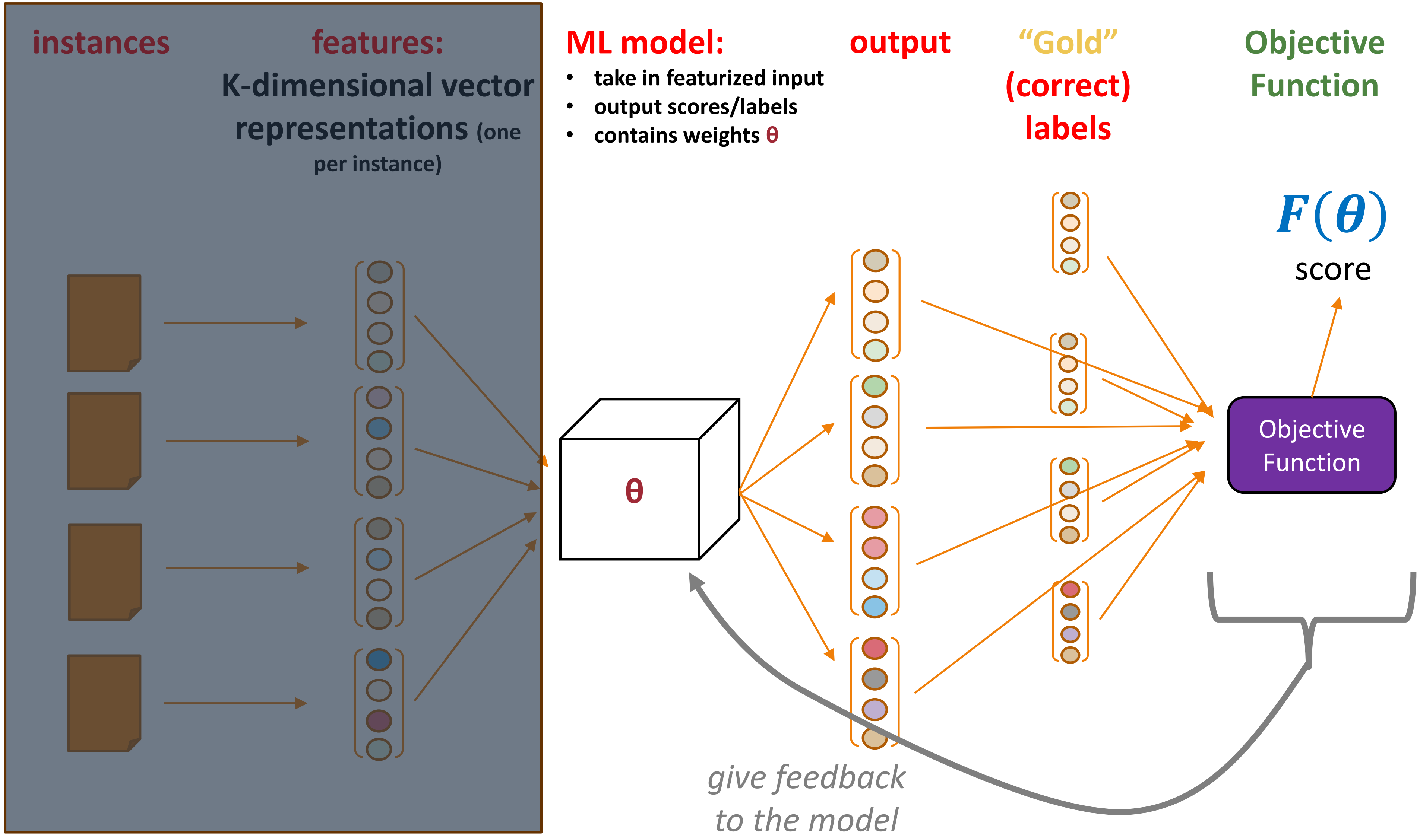


https://explained.ai/regularization/

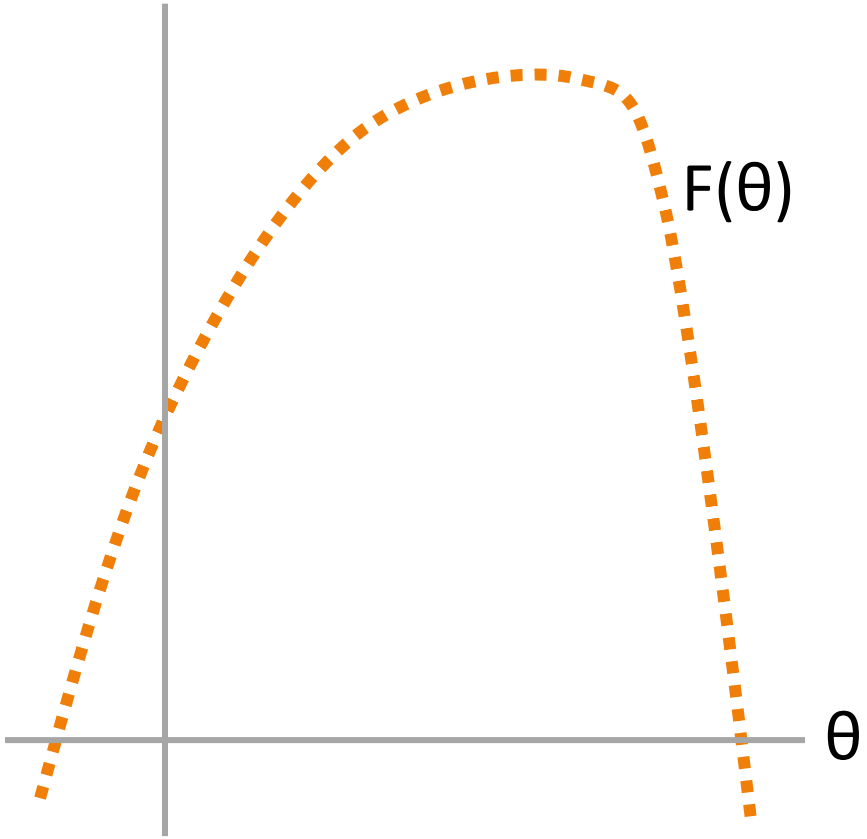
Regularization: Preventing Extreme Values

$$F(\theta) = \left(\sum_i \theta_{y_i}^T f(x_i) - \log Z(x_i) \right) - \sum_k \theta_k^2$$

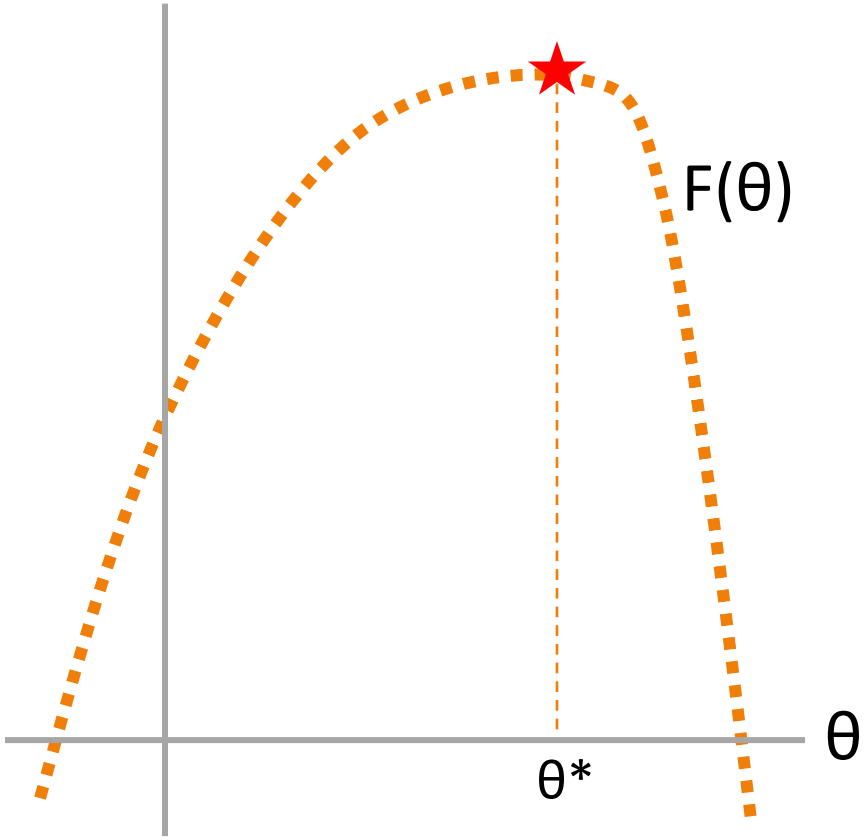
ML/NLP Framework for Learning



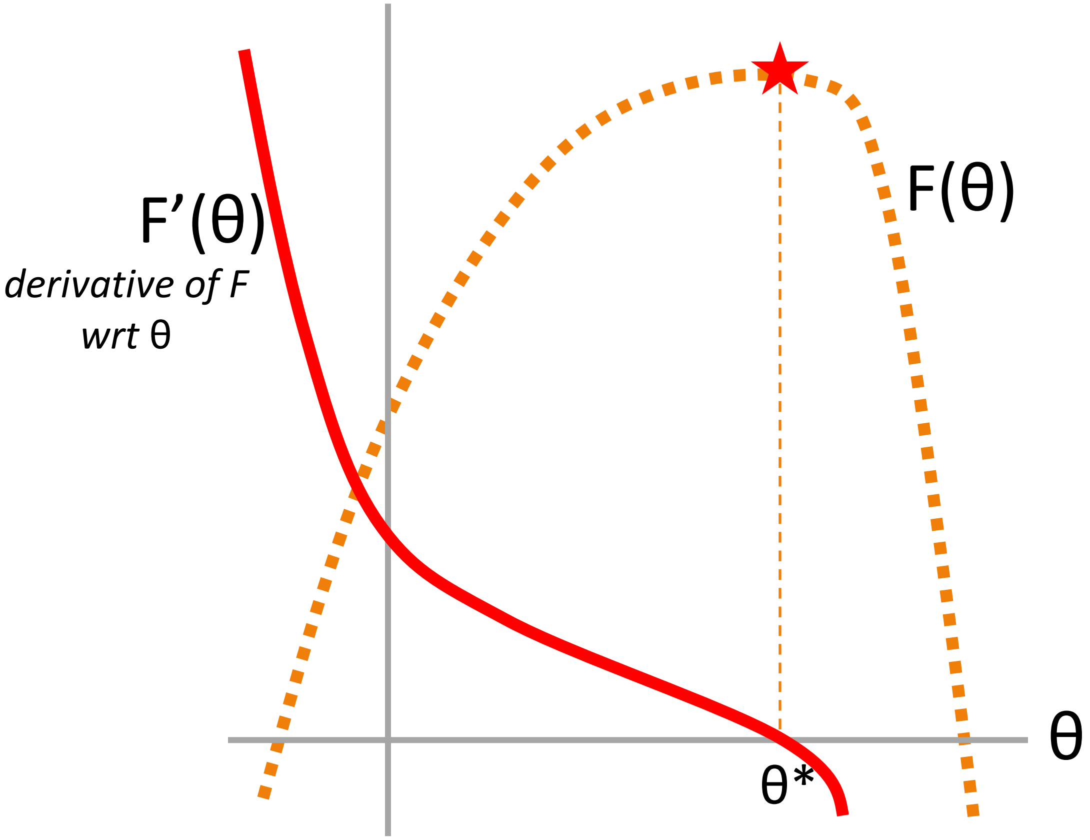
Optimizing F(θ)



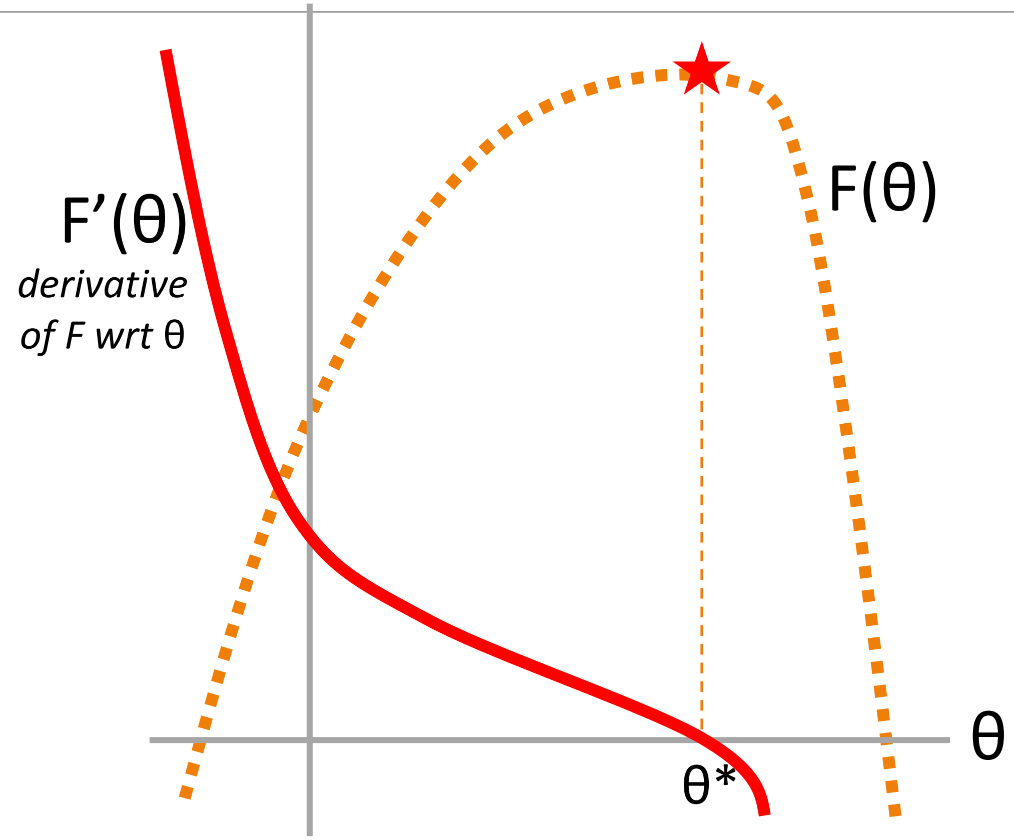
Optimizing F(θ)



Optimizing F(θ)



What if you can’t find the roots? Follow the derivative



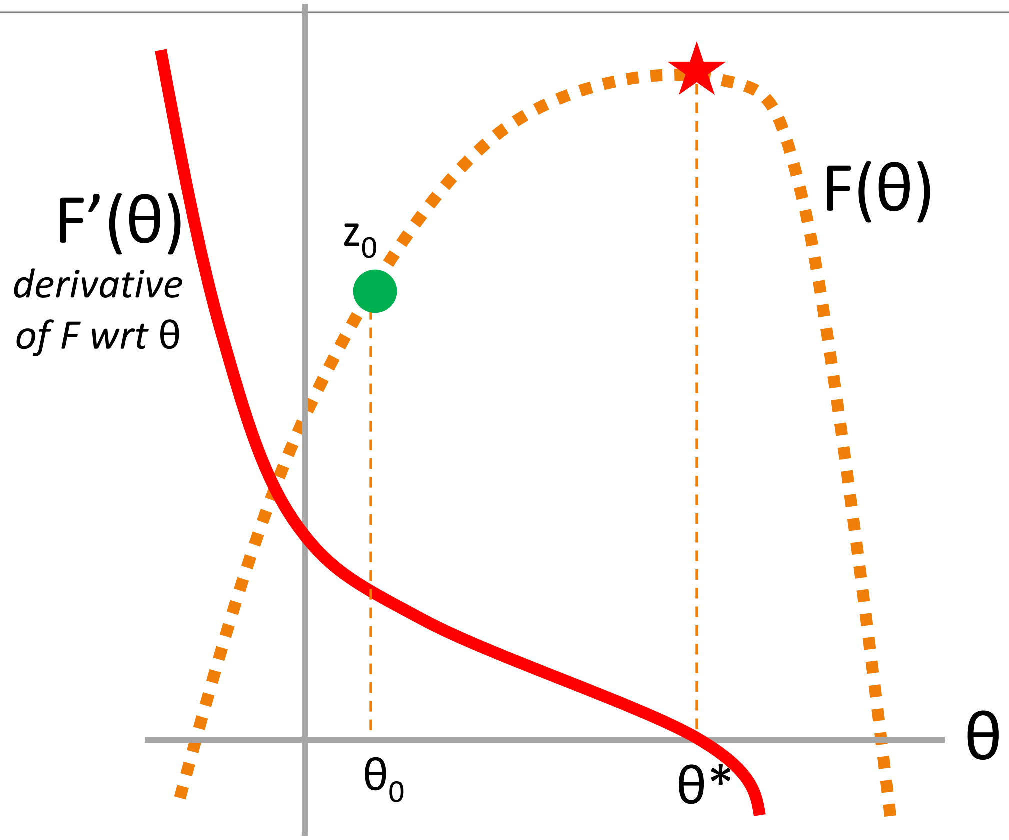
What if you can’t find the roots? Follow the derivative

Set t = 0

Pick a starting value θ_t

Until converged:

1. Get value $z_t = F(\theta_t)$



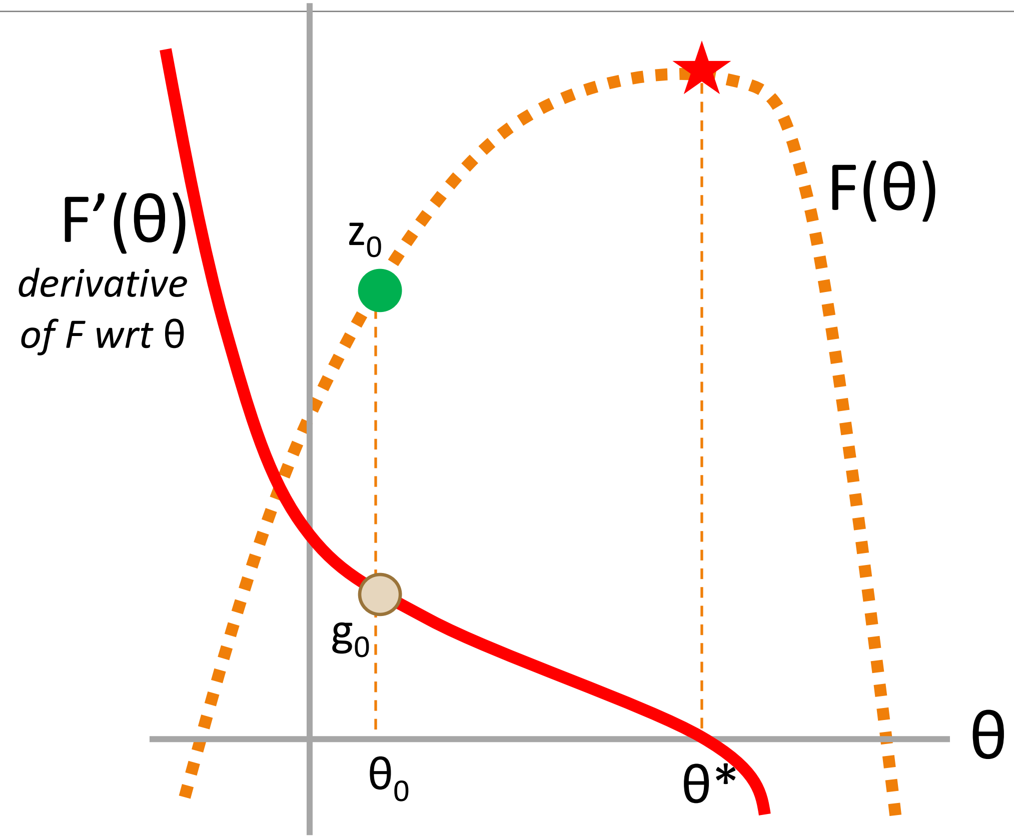
What if you can’t find the roots? Follow the derivative

Set t = 0

Pick a starting value θ_t

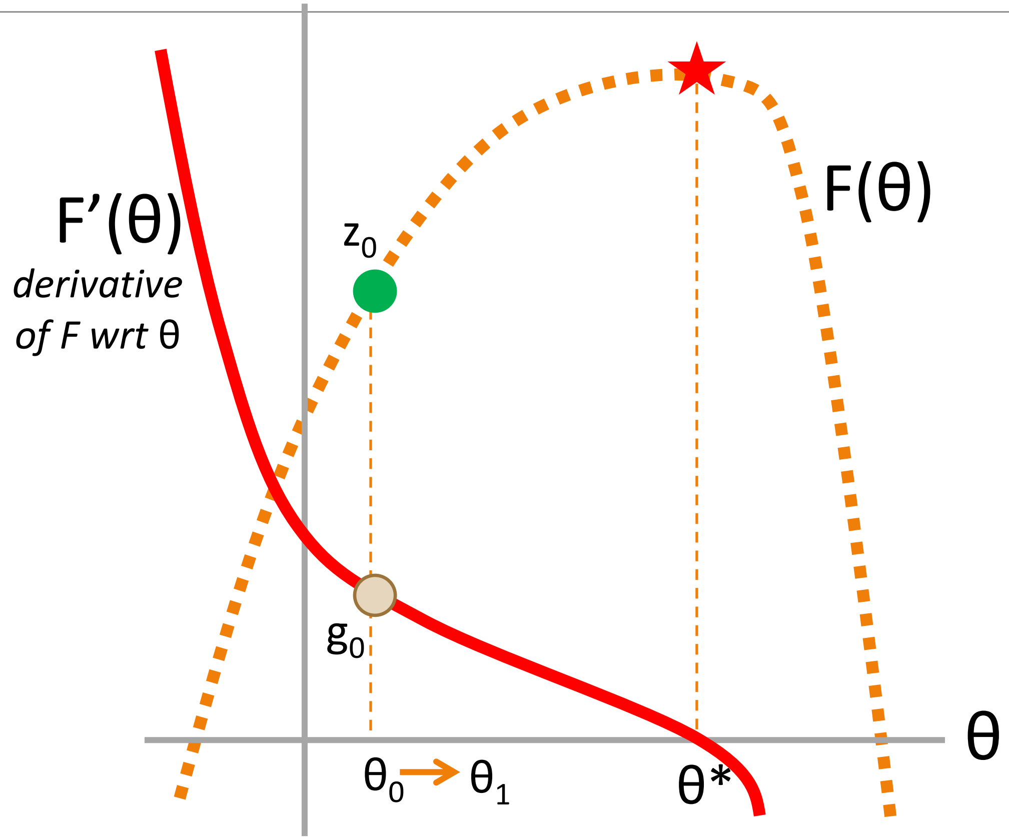
Until converged:

1. Get value $z_t = F(\theta_t)$
2. Get derivative $g_t = F'(\theta_t)$



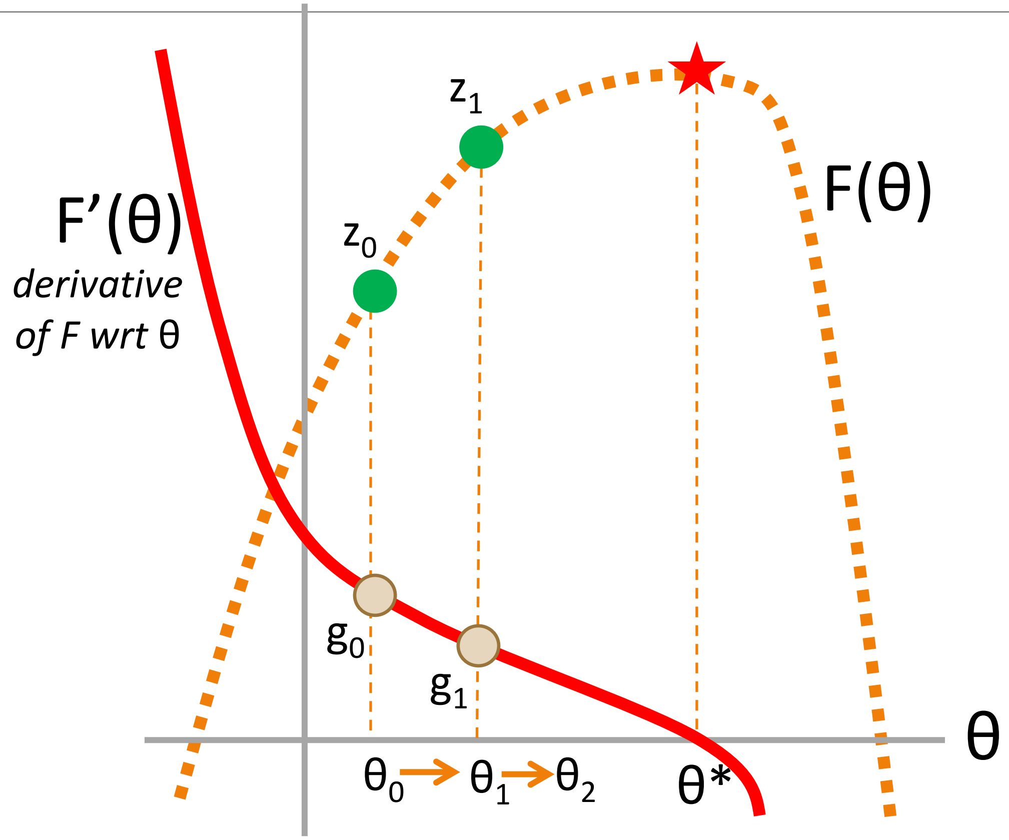
What if you can’t find the roots? Follow the derivative

```
Set t = 0
Pick a starting value $\theta_t$
Until converged:
    1. Get value $z_t = F(\theta_t)$
    2. Get derivative $g_t = F'(\theta_t)$
    3. Get scaling factor
       (learning rate) $\rho_t$
    4. Set $\theta_{t+1} = \theta_t + \rho_t * g_t$
    5. Set t += 1
```



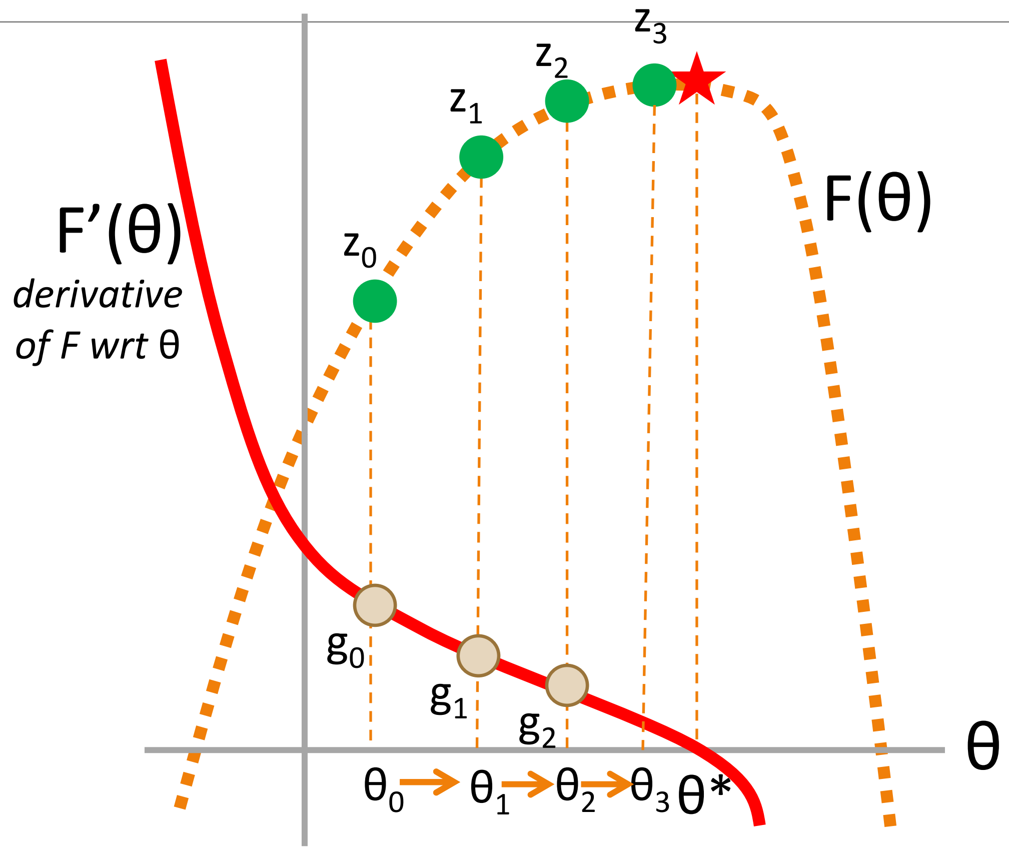
What if you can’t find the roots? Follow the derivative

```
Set t = 0
Pick a starting value $\theta_t$
Until converged:
    1. Get value $z_t = F(\theta_t)$
    2. Get derivative $g_t = F'(\theta_t)$
    3. Get scaling factor
       (learning rate) $\rho_t$
    4. Set $\theta_{t+1} = \theta_t + \rho_t * g_t$
    5. Set t += 1
```



What if you can't find the roots? Follow the derivative

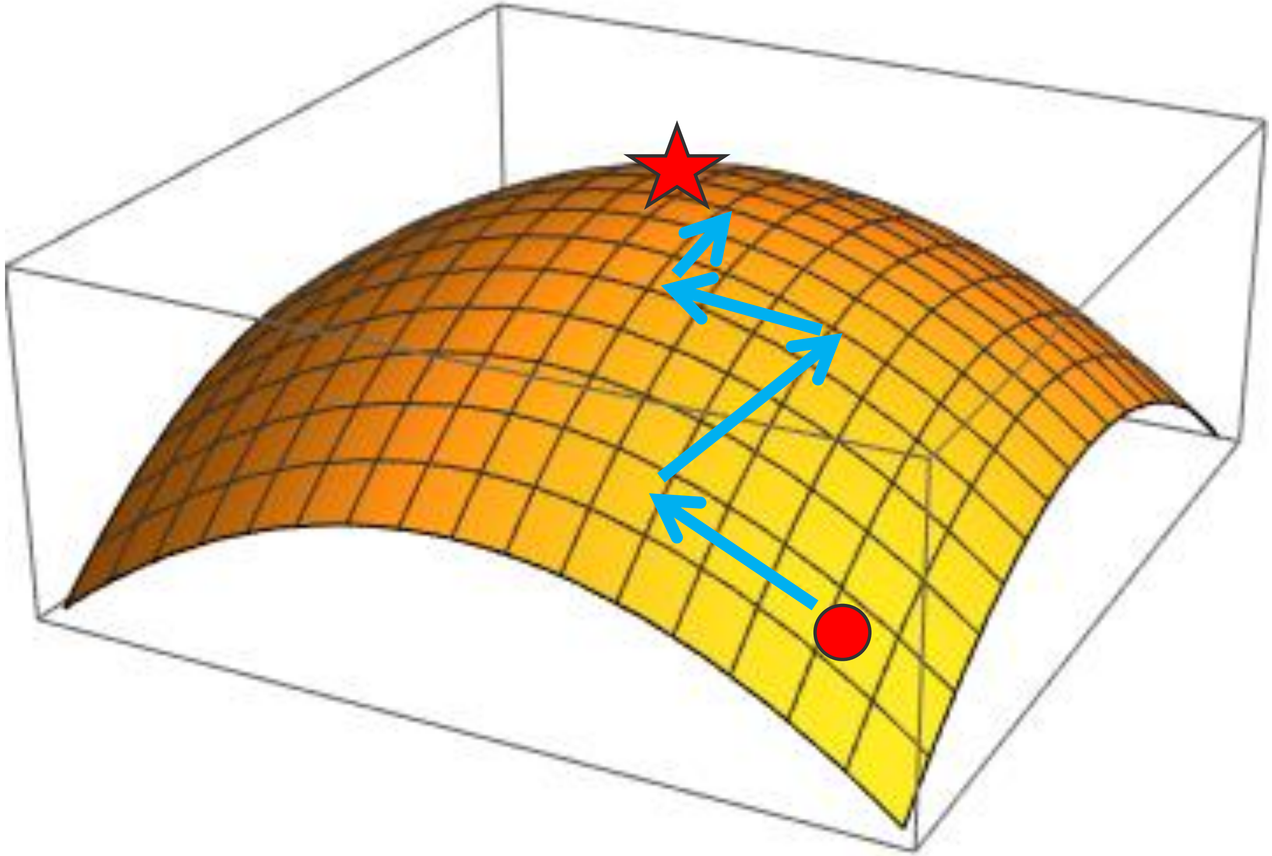
```
Set t = 0
Pick a starting value $\theta_t$
Until converged:
    1. Get value $z_t = F(\theta_t)$
    2. Get derivative $g_t = F'(\theta_t)$
    3. Get scaling factor
       (learning rate) $\rho_t$
    4. Set $\theta_{t+1} = \theta_t + \rho_t * g_t$
    5. Set t += 1
```



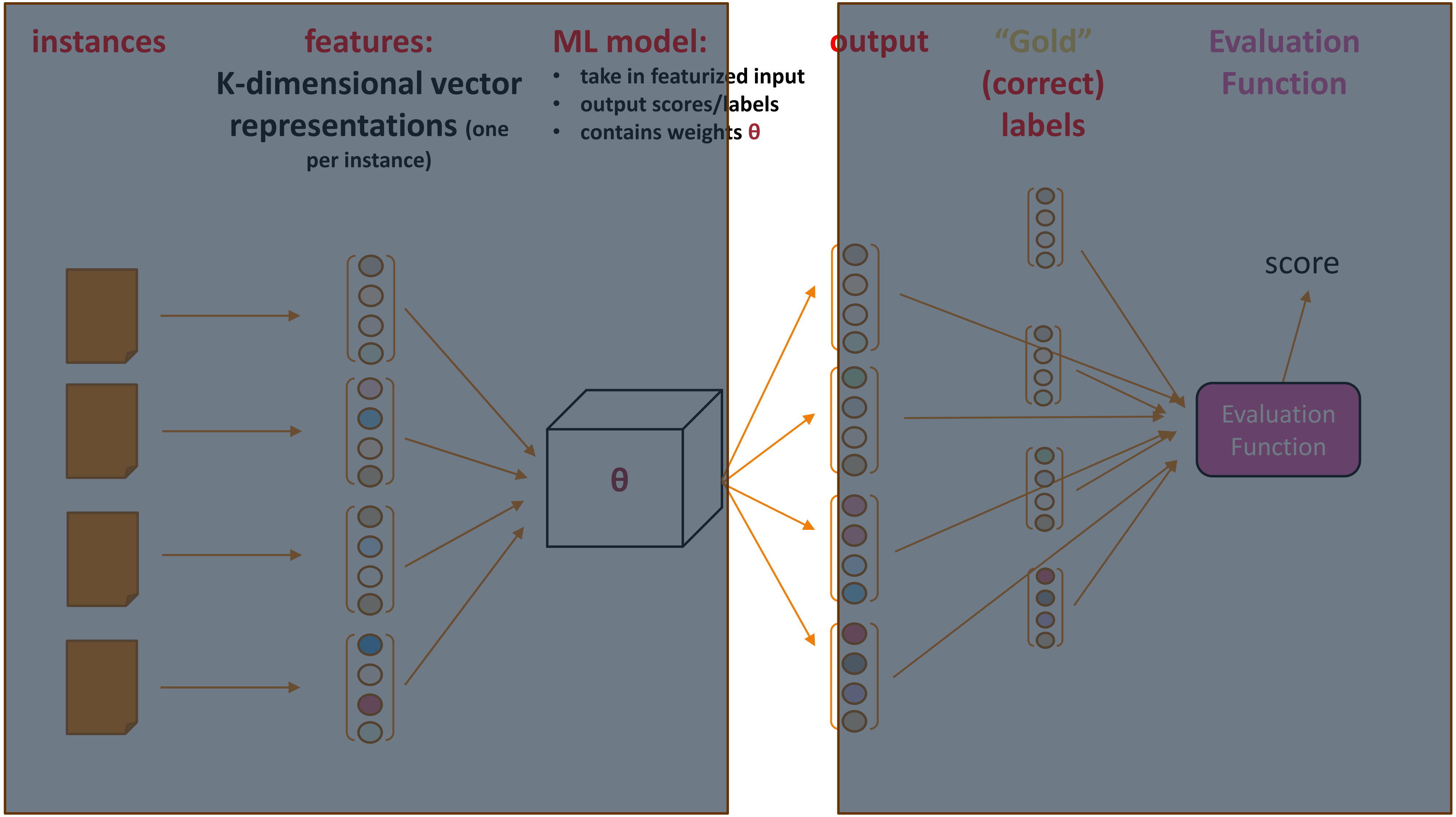
Gradient = Multi-variable derivative

$$\nabla_\theta F\left(\overset{\substack{\text{K-dimensional input}\\ \downarrow}}{\theta}\right) = \left(\underbrace{\frac{\partial F}{\partial \theta_1}, \frac{\partial F}{\partial \theta_2}, \ldots, \frac{\partial F}{\partial \theta_K}}_{\text{K-dimensional output}}\right)$$

Gradient Ascent



ML/NLP Framework for Prediction



- take in featurized input
- output scores/labels
- contains weights θ

Getting Labels from the Classifier

Given X, our classifier produces a score for each possible label

p(●|X) vs. p(○|X)

Can turn a probability ("regression") model into a classification model

$$\text{best label} = \arg\max_{\text{label}} P(\text{label}|\text{example})$$

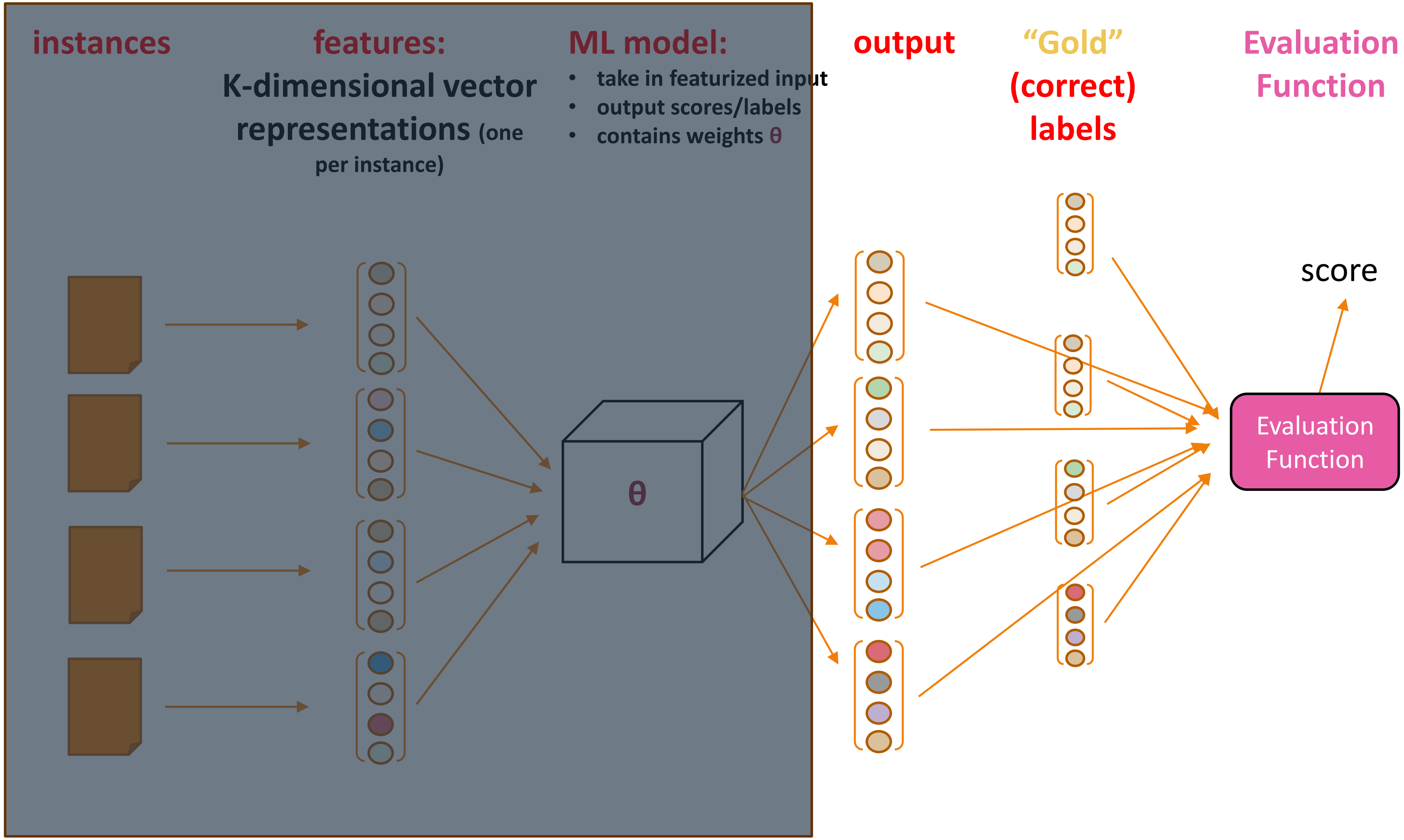
Example of `argmax`

Amazon acquired MGM in 2022, taking over a sprawling library that includes more than 4,000 feature films and 17,000 television shows. The tech behemoth also earned the rights to distribute all the Bond movies, but the new deal solidifies the company's oversight of Bond's big-screen future.

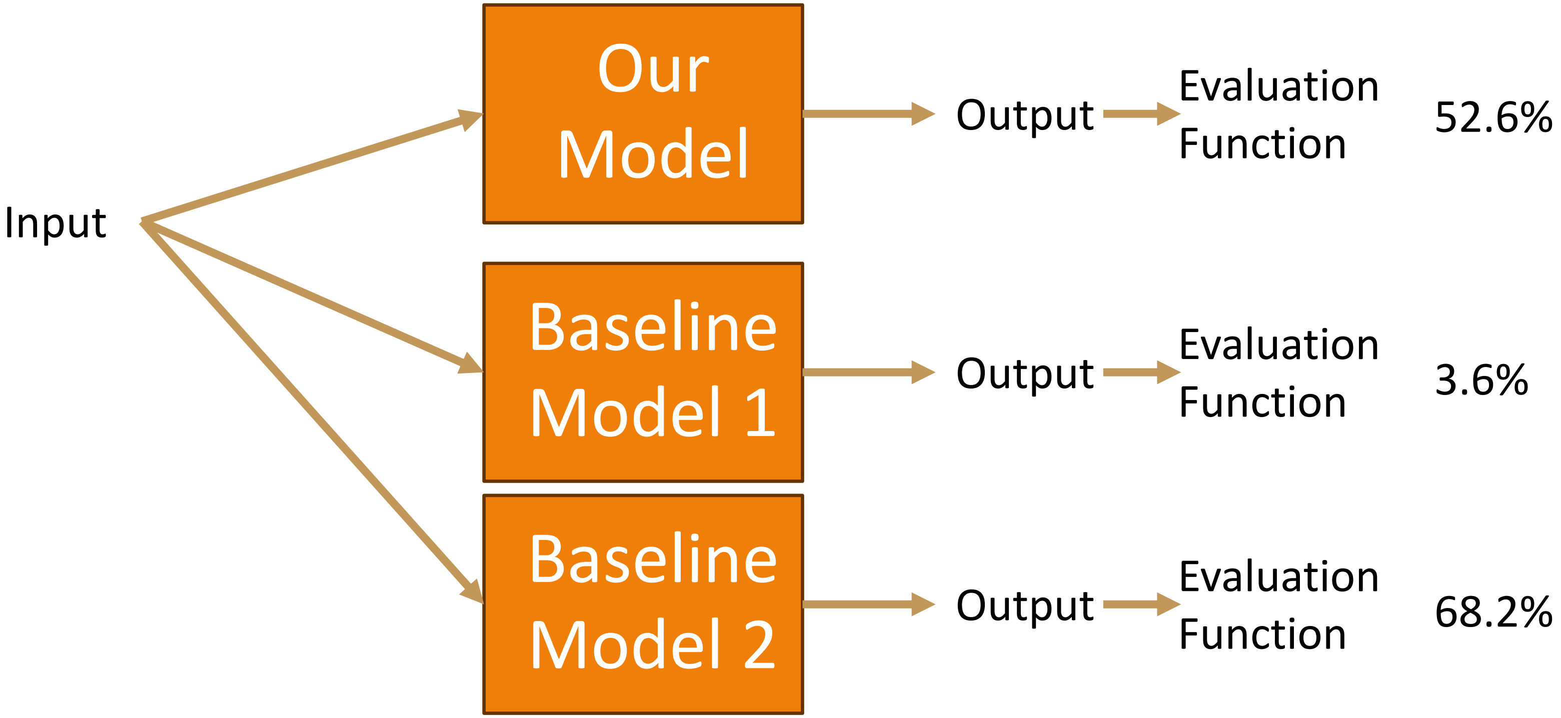
POLITICS	.002
MOVIES	**.48**
SPORTS	.0001
TECH	.39
HEALTH	.0001
FINANCE	.05
…	

Adapted from https://www.nbcnews.com/pop-culture/movies/amazon-taking-control-james-bond-movie-franchise-rcna192967

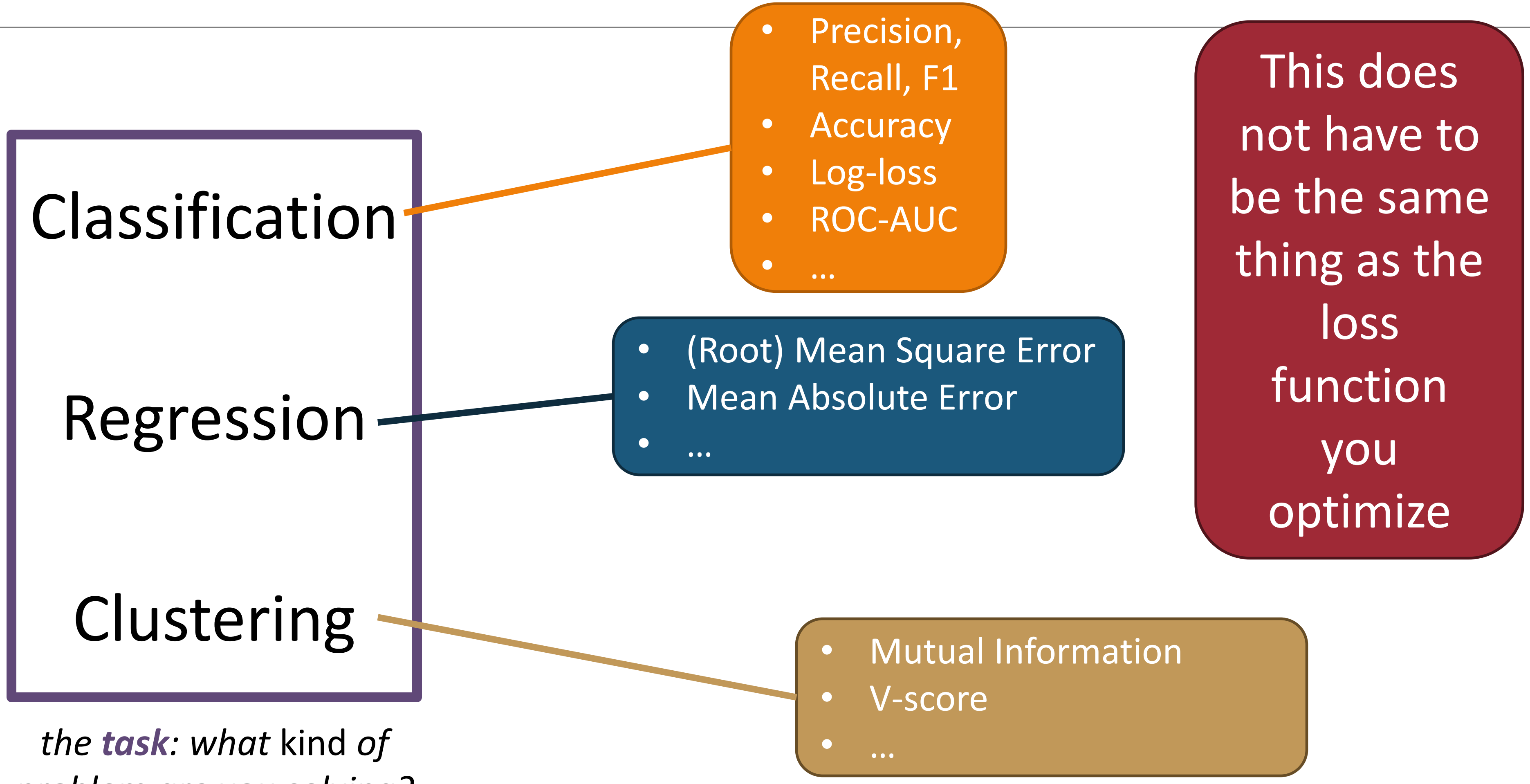
ML/NLP Framework for Prediction



Determining how good a model is: Baselines



Central Question: How Well Are We Doing?



Classification

- Precision, Recall, F1
- Accuracy
- Log-loss
- ROC-AUC
- …

Regression

- (Root) Mean Square Error
- Mean Absolute Error
- …

Clustering

- Mutual Information
- V-score
- …

This does not have to be the same thing as the loss function you optimize

*the **task**: what kind of problem are you solving?*

Evaluating Classification

Classification Evaluation:
the 2-by-2 contingency table

Assumption 1: There are two classes/labels

● ○

Assumption 2: ● is the “positive” label

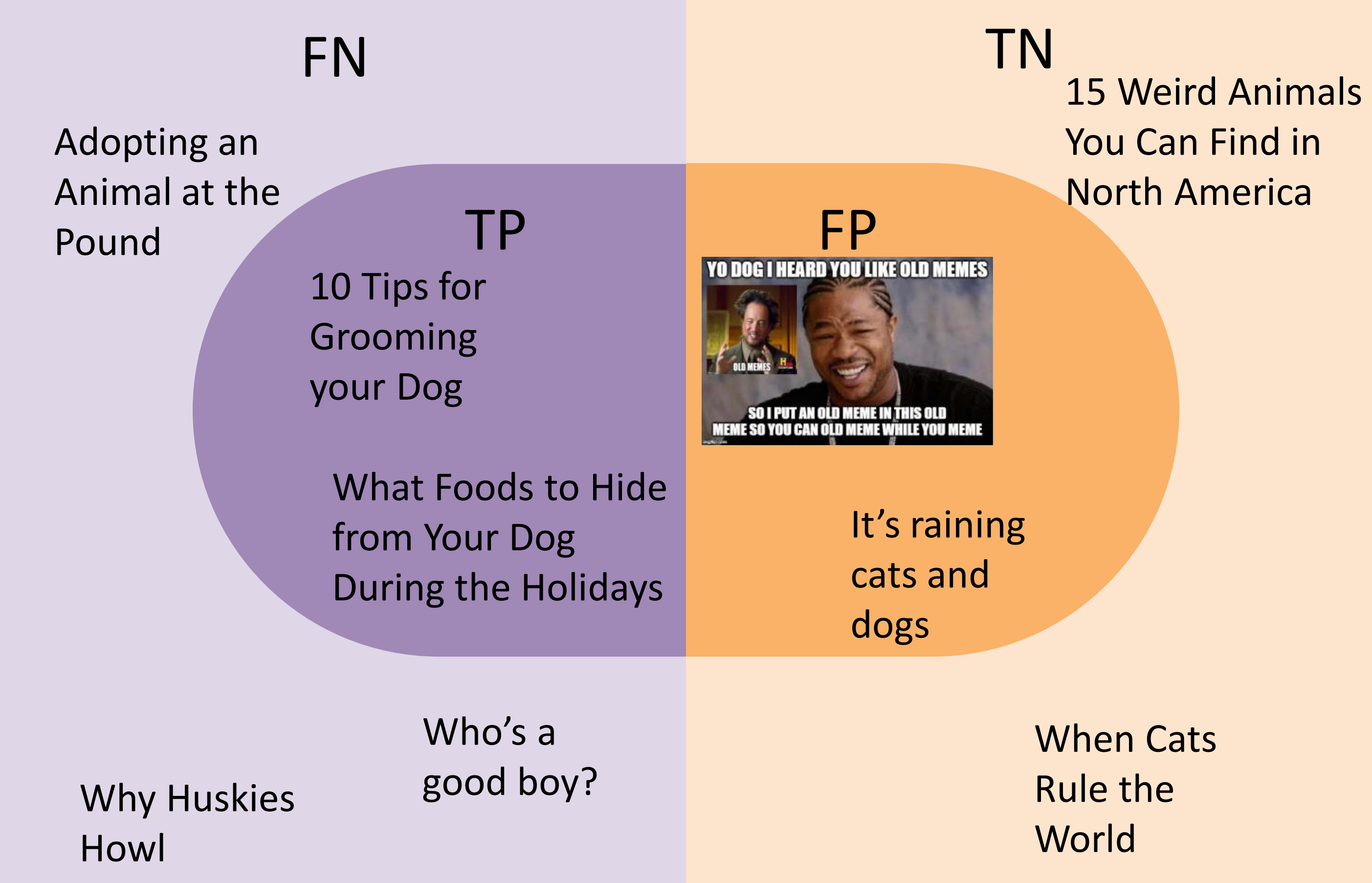
Classification Evaluation:
the 2-by-2 contingency table

	What is the actual label?	
What label does our system predict? (↓)	Actual Target Class ("●")	Not Target Class ("○")
Selected/ Guessed ("●")	True Positive ● (TP) ● *Actual* *Guessed*	False Positive ○ (FP) ● *Actual* *Guessed*
Not selected/ not guessed ("○")	False Negative ● (FN) ○ *Actual* *Guessed*	True Negative ○ (TN) ○ *Actual* *Guessed*

Contingency Table (out of table form)

Query:
Articles about dogs

Simple model classifies based on presence of “dog” or “dogs”



FN

Adopting an Animal at the Pound

Why Huskies Howl

Who’s a good boy?

TP

10 Tips for Grooming your Dog

What Foods to Hide from Your Dog During the Holidays

FP

It’s raining cats and dogs

TN

15 Weird Animals You Can Find in North America

When Cats Rule the World

YO DOG I HEARD YOU LIKE OLD MEMES
OLD MEMES
SO I PUT AN OLD MEME IN THIS OLD MEME SO YOU CAN OLD MEME WHILE YOU MEME

Meme from: *https://www.reddit.com/r/AdviceAnimals/comments/ck8xh0/yo_dawg_i_heard_you_like_old_memes/*

Contingency Table Example

Predicted:	○	●	●	●	○	●
Actual:	●	●	●	○	○	○

	What is the actual label?	
What label does our system predict? (↓)	Actual Target Class ("●")	Not Target Class ("○")
Selected/ Guessed ("●")	True Positive (TP)	False Positive (FP)
Not selected/ not guessed ("○")	False Negative (FN)	True Negative (TN)

Contingency Table Example

Predicted: ○ ● ● ● ○ ●

Actual: ● ● ● ○ ○ ○

	What is the actual label?	
What label does our system predict? (↓)	Actual Target Class ("●")	Not Target Class ("○")
Selected/ Guessed ("●")	True Positive (TP) = 2	False Positive (FP) = 2
Not selected/ not guessed ("○")	False Negative (FN) = 1	True Negative (TN) = 1

Classification Evaluation: Accuracy, Precision, and Recall

Accuracy: % of items correct

$$\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}$$

	Actually Target	Actually Not Target
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Classification Evaluation: Accuracy, Precision, and Recall

Accuracy: % of items correct

$$\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}$$

Precision: % of selected items that are correct

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

"**Precision** measures the percentage of the items that precision the system detected (i.e., the system labeled as positive) that are in fact positive (i.e., are positive according to the human gold labels"

SLP, ch. 4

	Actually Target	Actually Not Target
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Classification Evaluation: Accuracy, Precision, and Recall

Accuracy: % of items correct

$$\frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Precision: % of selected items that are correct

$$\frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall: % of correct items that are selected

$$\frac{\text{TP}}{\text{TP} + \text{FN}}$$

	Actually Target	Actually Not Target
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

> "**Recall** measures the percentage of items actually present in the input that were correctly identified by the system."
> SLP, ch. 4

Classification Evaluation: Accuracy, Precision, and Recall

Accuracy: % of items correct

$$\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{FN}+\mathrm{TN}}$$

Precision: % of selected items that are correct

$$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$$

Recall: % of correct items that are selected

$$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$$

Min: 0 ☹
Max: 1 😀

	Actually Target	**Actually Not Target**
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Comparing Accuracy & F1

Accuracy: % of items correct

$$\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}$$

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

Accuracy takes everything in consideration

F-Score is focused on TP

When would you want to use accuracy vs F1?

Accuracy works better if the dataset is balanced

	Actually Target	Actually Not Target
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

P/R/F in a Multi-class Setting: Micro- vs. Macro-Averaging

Macroaveraging: Compute performance for each class, then average.

$$\text{macroprecision} = \frac{1}{C}\sum_{c}\frac{\mathrm{TP_c}}{\mathrm{TP_c}+\mathrm{FP_c}} = \frac{1}{C}\sum_{c}\text{precision}_{c}$$

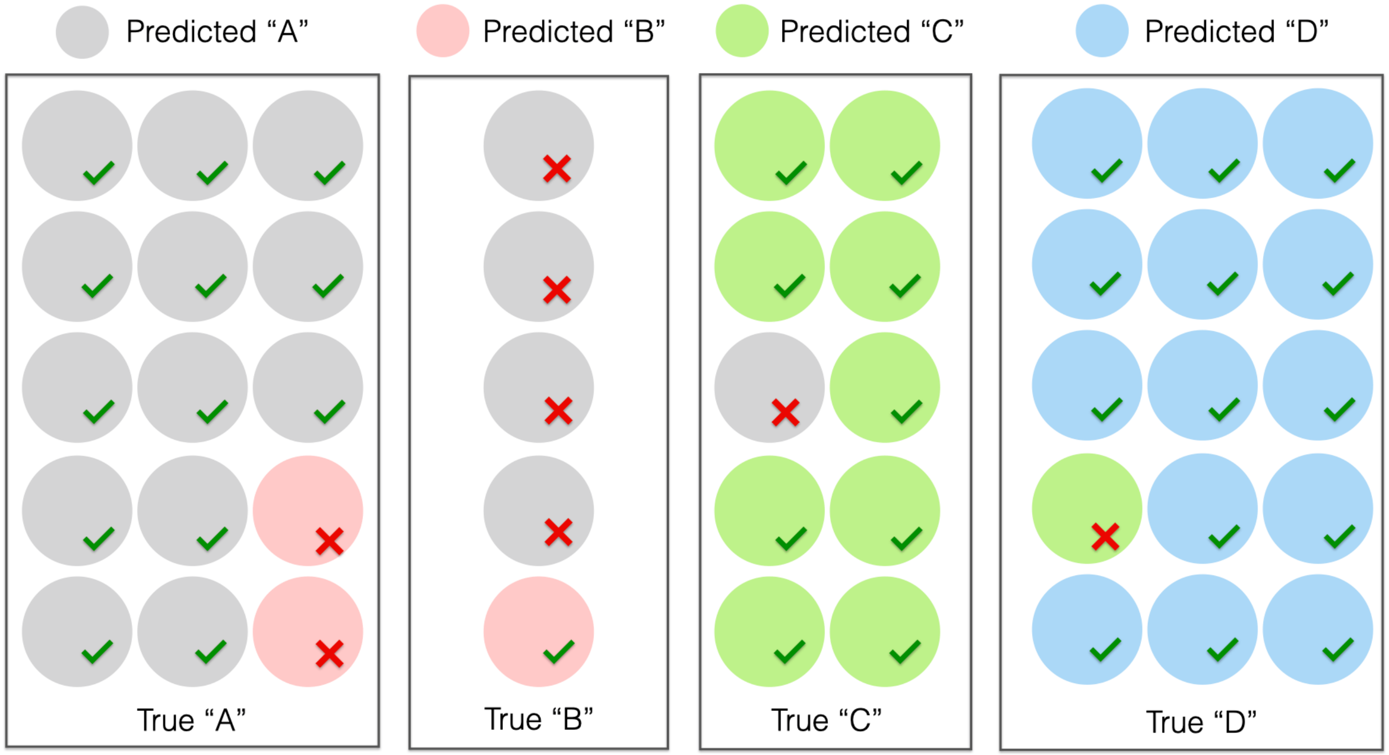
$$\text{macrorecall} = \frac{1}{C}\sum_{c}\frac{\mathrm{TP_c}}{\mathrm{TP_c}+\mathrm{FN_c}} = \frac{1}{C}\sum_{c}\text{recall}_{\mathrm{c}}$$

Microaveraging: Collect decisions for all classes, compute contingency table, evaluate.

$$\text{microprecision} = \frac{\sum_{\mathrm{c}}\mathrm{TP_c}}{\sum_{\mathrm{c}}\mathrm{TP_c}+\sum_{\mathrm{c}}\mathrm{FP_c}}$$

$$\text{microrecall} = \frac{\sum_{\mathrm{c}}\mathrm{TP_c}}{\sum_{\mathrm{c}}\mathrm{TP_c}+\sum_{\mathrm{c}}\mathrm{FN_c}}$$

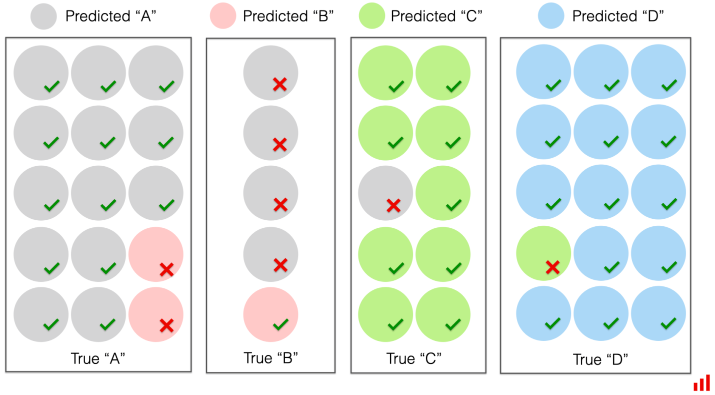
Macro/Micro Example

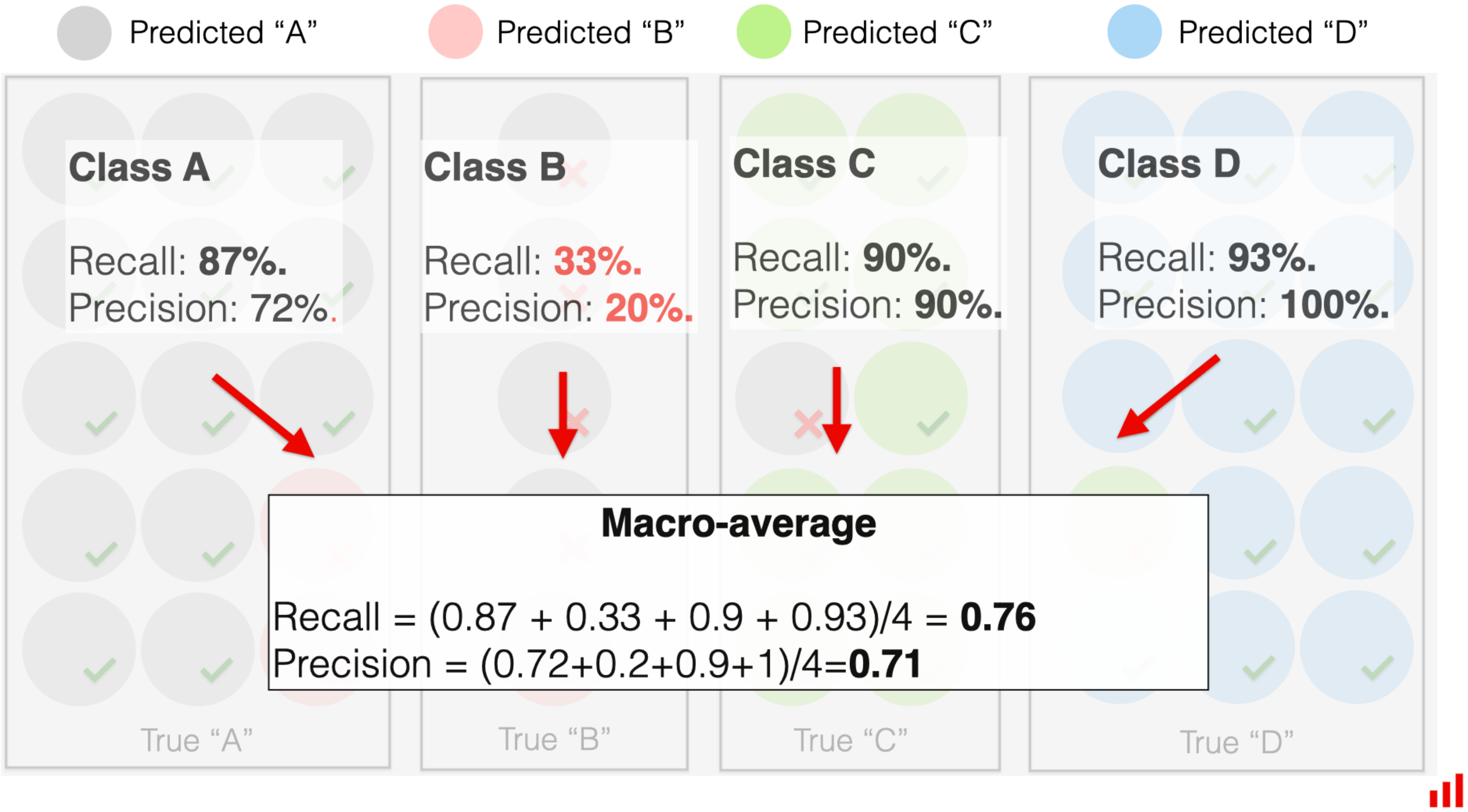


https://www.evidentlyai.com/classification-metrics/multi-class-metrics

Each *class* has equal weight

Macro-Average





Class A

Recall: **87%.**
Precision: 72%.

Class B

Recall: **33%.**
Precision: **20%.**

Class C

Recall: **90%.**
Precision: **90%.**

Class D

Recall: **93%.**
Precision: **100%.**

Macro-average

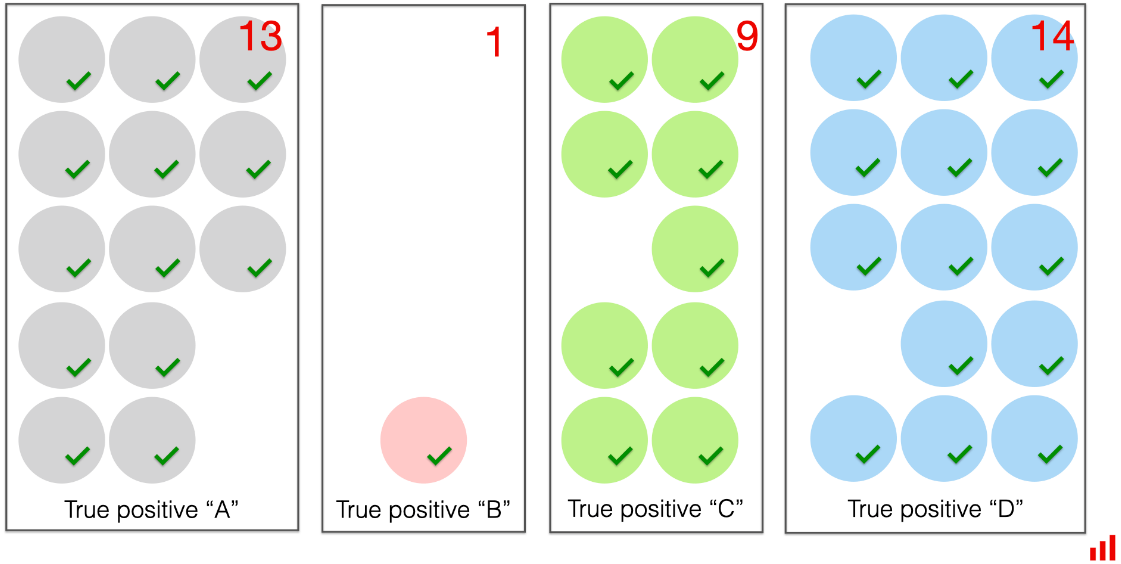
Recall = (0.87 + 0.33 + 0.9 + 0.93)/4 = **0.76**
Precision = (0.72+0.2+0.9+1)/4=**0.71**

https://www.evidentlyai.com/classification-metrics/multi-class-metrics

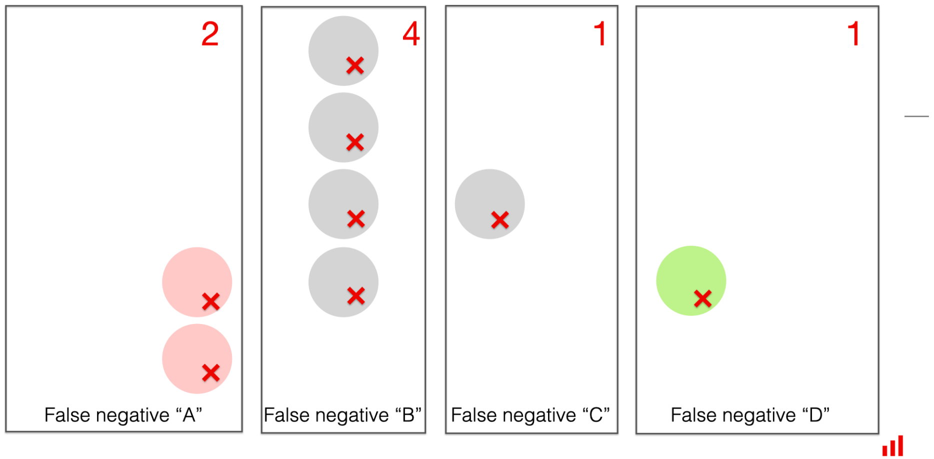
Each *instance* has equal weight

Micro-Average

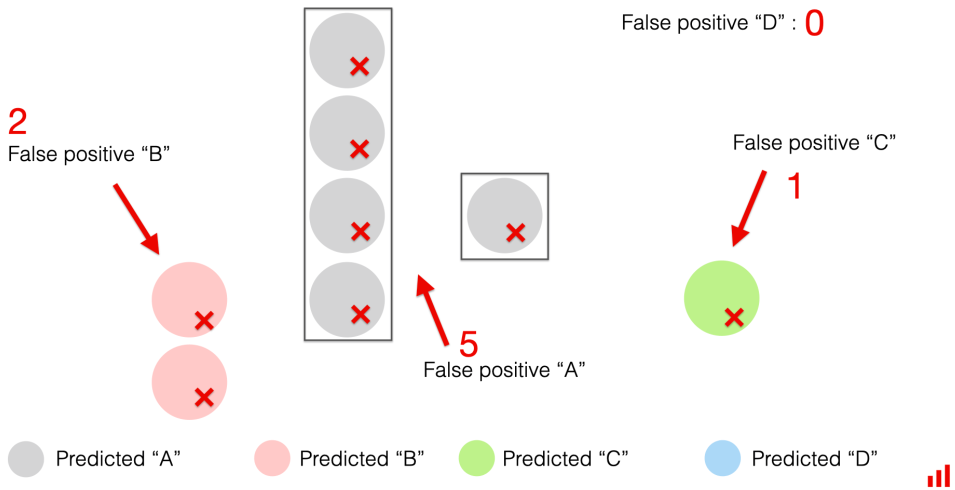
All true positives



All false negatives



All false positives



Total TP Total FP Total FN

$$\text{Precision}_{\text{Micro-average}} = \frac{13+1+9+14}{13+1+9+14 + \boxed{2+5+1+0}} = 0.82$$

$$\text{Recall}_{\text{Micro-average}} = \frac{13+1+9+14}{13+1+9+14 + 2+4+1+1} = 0.82$$

https://www.evidentlyai.com/classification-metrics/multi-class-metrics

But how do we compute stats for multiple classes?

We already saw how the "polarity" affects the stats we compute…

Two main approaches. Either:

1. Compute "one-vs-all" 2x2 tables. OR
2. Generalize the 2x2 tables and compute per-class TP / FP / FN based on the diagonals and off-diagonals

1. Compute “one-vs-all” 2x2 tables

Predicted	●	○	▭	●	○	▭	●	○	▭
Actual	●	○	○	▭	○	▭	●	●	●

Look for ●	Actually Target	Actually Not Target
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Look for ○	Actually Target	Actually Not Target
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Look for ▭	Actually Target	Actually Not Target
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

1. Compute “one-vs-all” 2x2 tables

Predicted	●	○	▭	●	○	▭	●	○	▭
Actual	●	○	○	▭	○	▭	●	●	●

Look for ●	Actually Target	Actually Not Target
Selected/Guessed	2	1
Not select/not guessed	2	4

Look for ○	Actually Target	Actually Not Target
Selected/Guessed	2	1
Not select/not guessed	1	5

Look for ▭	Actually Target	Actually Not Target
Selected/Guessed	1	2
Not select/not guessed	1	5

2. Generalizing the 2-by-2 contingency table

		Correct Value		
		●	○	▭
Guessed Value	●	#	#	#
	○	#	#	#
	▭	#	#	#

This is also called a **Confusion Matrix**

2. Generalizing the 2-by-2 contingency table

Predicted	●	○	▭	●	○	▭	●	○	▭
Actual	●	○	○	▭	○	▭	●	●	●

		Correct Value		
		●	○	▭
Guessed Value	●	2	0	1
	○	1	2	0
	▭	1	1	1

This is also called a **Confusion Matrix**

2. Generalizing the 2-by-2 contingency table

Predicted	●	○	▭	●	○	▭	●	○	▭
Actual	●	○	○	▭	○	▭	●	●	●

		Correct Value		
		●	○	▭
Guessed Value	●	A 2	B 0	C 1
	○	D 1	E 2	F 0
	▭	G 1	H 1	I 1

How do you compute $TP_{\bullet}$?

2. Generalizing the 2-by-2 contingency table

Predicted	●	○	▭	●	○	▭	●	○	▭
Actual	●	○	○	▭	○	▭	●	●	●

		Correct Value		
		●	○	▭
Guessed Value	●	A 2	B 0	C 1
	○	D 1	E 2	F 0
	▭	G 1	H 1	I 1

How do you compute $FN_{●}$?

2. Generalizing the 2-by-2 contingency table

Predicted	●	○	▭	●	○	▭	●	○	▭
Actual	●	○	○	▭	○	▭	●	●	●

		Correct Value		
		●	○	▭
Guessed Value	●	A 2	B 0	C 1
	○	D 1	E 2	F 0
	▭	G 1	H 1	I 1

How do you compute $FP_{\square}$?

Evaluating Generation

Evaluating Language Models

What is “correct?”

What is working “well?”

Extrinsic: Evaluate LM in downstream task

Test an MT, ASR, etc. system and see which LM does better

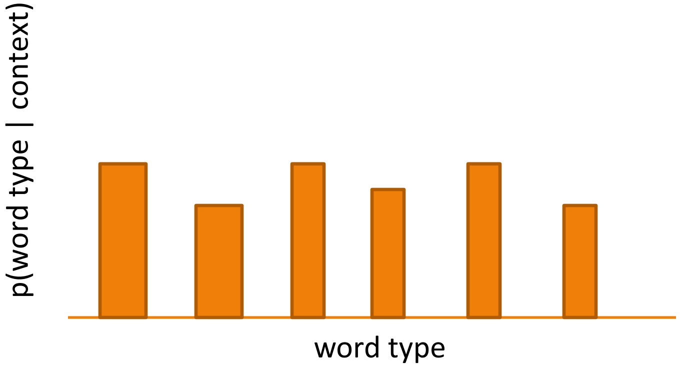
Issue: Propagate & conflate errors

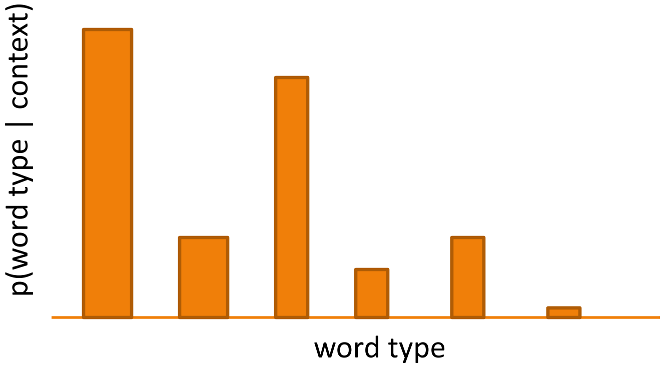
Intrinsic: Treat LM as its own downstream task

Use perplexity (from information theory)

Perplexity: Average "Surprisal"

Lower is better : lower perplexity ➔ less surprised





Less certain ➔
More surprised ➔
Higher perplexity

More certain ➔
Less surprised ➔
Lower perplexity

Perplexity

Lower is better : lower perplexity ➔ less surprised

$$\text{perplexity} = \exp(\text{avg crossentropy})$$

Perplexity

Lower is better : lower perplexity ➔ less surprised

$$\text{perplexity} = \exp\left(\frac{-1}{M}\log p(w_1, \ldots, w_M)\ \right)$$

Perplexity

Lower is better : lower perplexity ➔ less surprised

e.g., n-gram history
(n-1 items)

$$\text{perplexity} = \exp\left(\frac{-1}{M} \sum_{i=1}^{M} \log p(w_i \mid h_i)\right)$$

Example perplexity for trigram model

"The film , a hit !"

Trigrams	MLE p(trigram)
<BOS> <BOS> The	1
<BOS> The film	1
The film ,	0
film , a	0
, a hit	0
a hit !	0
hit ! <EOS>	0
Perplexity	Infinity

$$\text{perplexity} = \exp\left(\frac{-1}{M}\sum_{i=1}^{M}\log p(w_i \mid h_i)\right)$$

Example perplexity for trigram model

Trigrams	MLE p(trigram)	Smoothed p(trigram)
<BOS> <BOS> The	1	2/17
<BOS> The film	1	2/17
The film ,	0	1/17
film , a	0	1/16
, a hit	0	1/16
a hit !	0	1/17
hit ! <EOS>	0	1/16
Perplexity	Infinity	13.59

"The film , a hit !"

$$\text{perplexity} = \exp\left(\frac{-1}{M}\sum_{i=1}^{M}\log p(w_i \mid h_i)\right)$$

instances

features:
K-dimensional vector representations (one per instance)

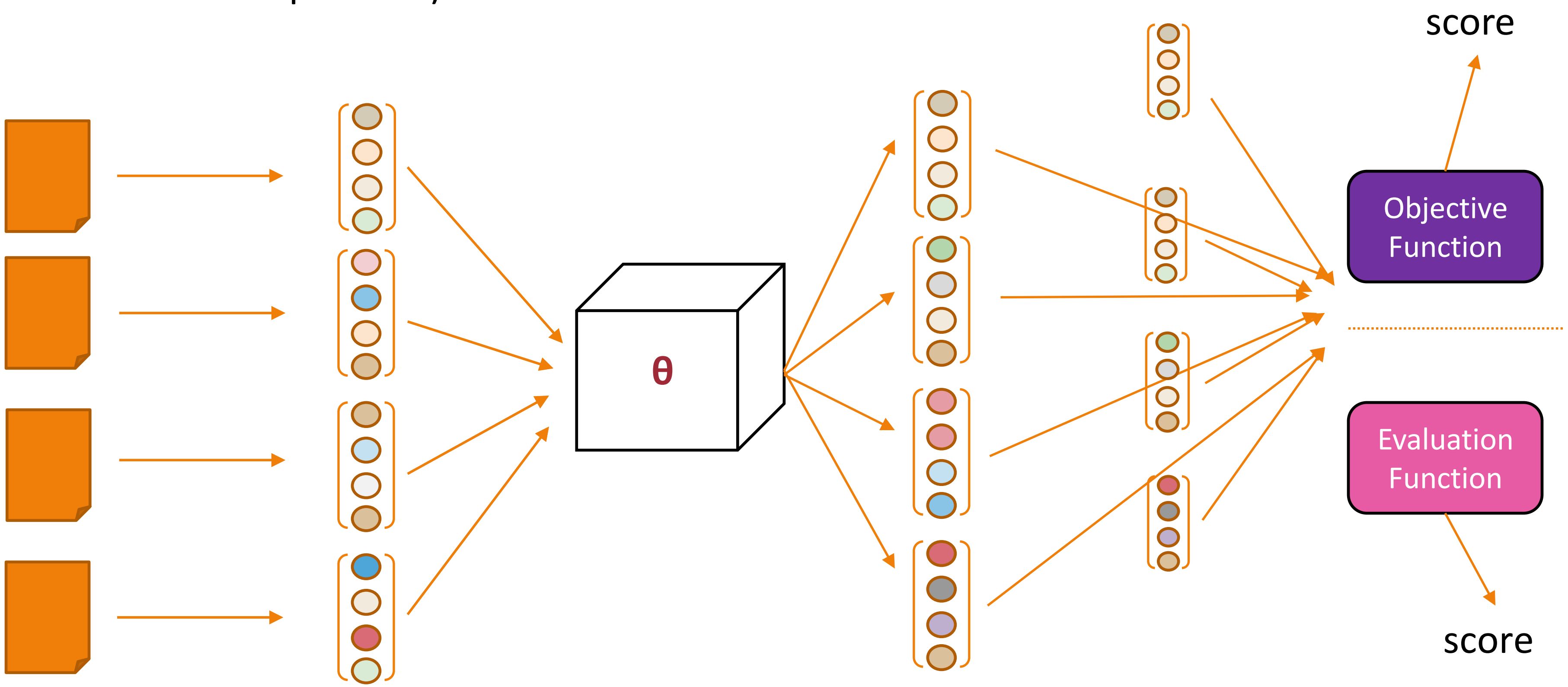
ML model:

- take in featurized input
- output scores/labels
- contains weights θ

output

“Gold” (correct) labels

Objective / Eval Function



Text Annotation Tasks (“Classification” Tasks)

1. Classify the entire document (“text categorization”)
2. Classify word tokens individually
3. Classify word tokens in a sequence
4. Identify phrases (“chunking”)
5. Syntactic annotation (parsing)
6. Semantic annotation

Slide courtesy Jason Eisner, with mild edits

Text Annotation Tasks (“Classification” Tasks)

1. Classify the entire document (“text categorization”)
2. Classify word tokens individually
3. Classify word tokens in a sequence
4. Identify phrases (“chunking”)
5. Syntactic annotation (parsing)
6. Semantic annotation

Slide courtesy Jason Eisner, with mild edits

Document Classification

Assigning subject categories, topics, or genres

Spam detection

Authorship identification

Language Identification

Sentiment analysis

…

Document Classification

Assigning subject categories, topics, or genres

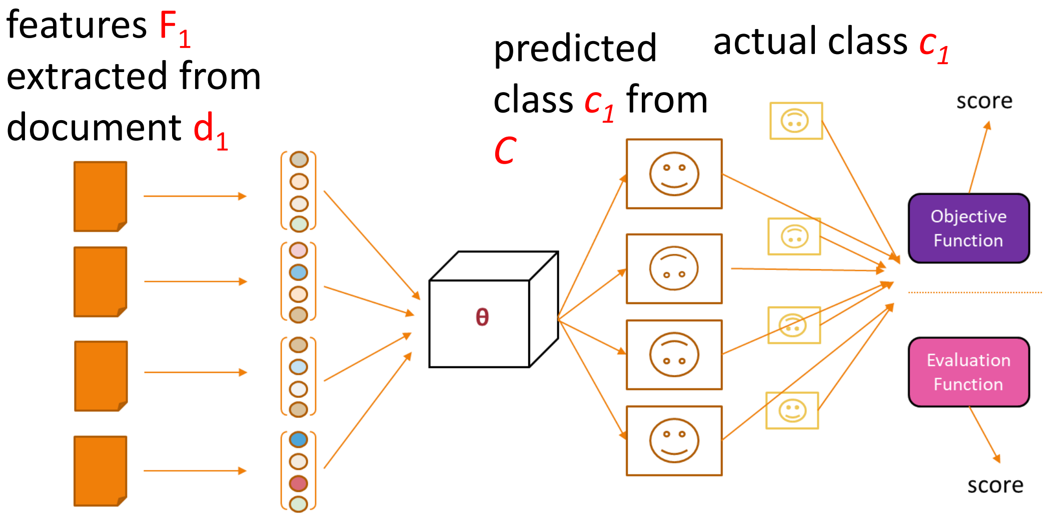
Spam detection

Authorship identification

Language Identification

Sentiment analysis

…


features F_1 extracted from document d_1
predicted class c_1 from C
actual class c_1
θ
score
Objective Function
Evaluation Function
score

Document Classification

Assigning subject categories, topics, or genres

Spam detection

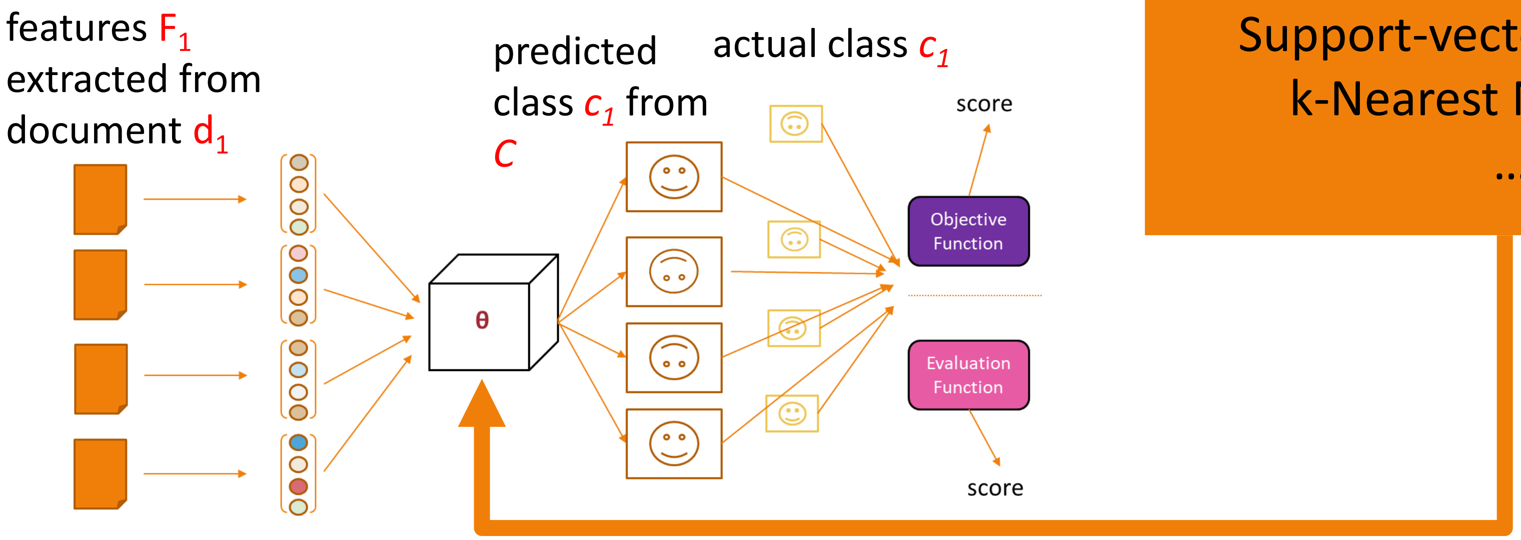
Authorship identification

Language Identification

Sentiment analysis

…

Naïve Bayes
Logistic regression
Neural network
Support-vector machines
k-Nearest Neighbors
…



features F_1 extracted from document d_1

predicted class c_1 from C

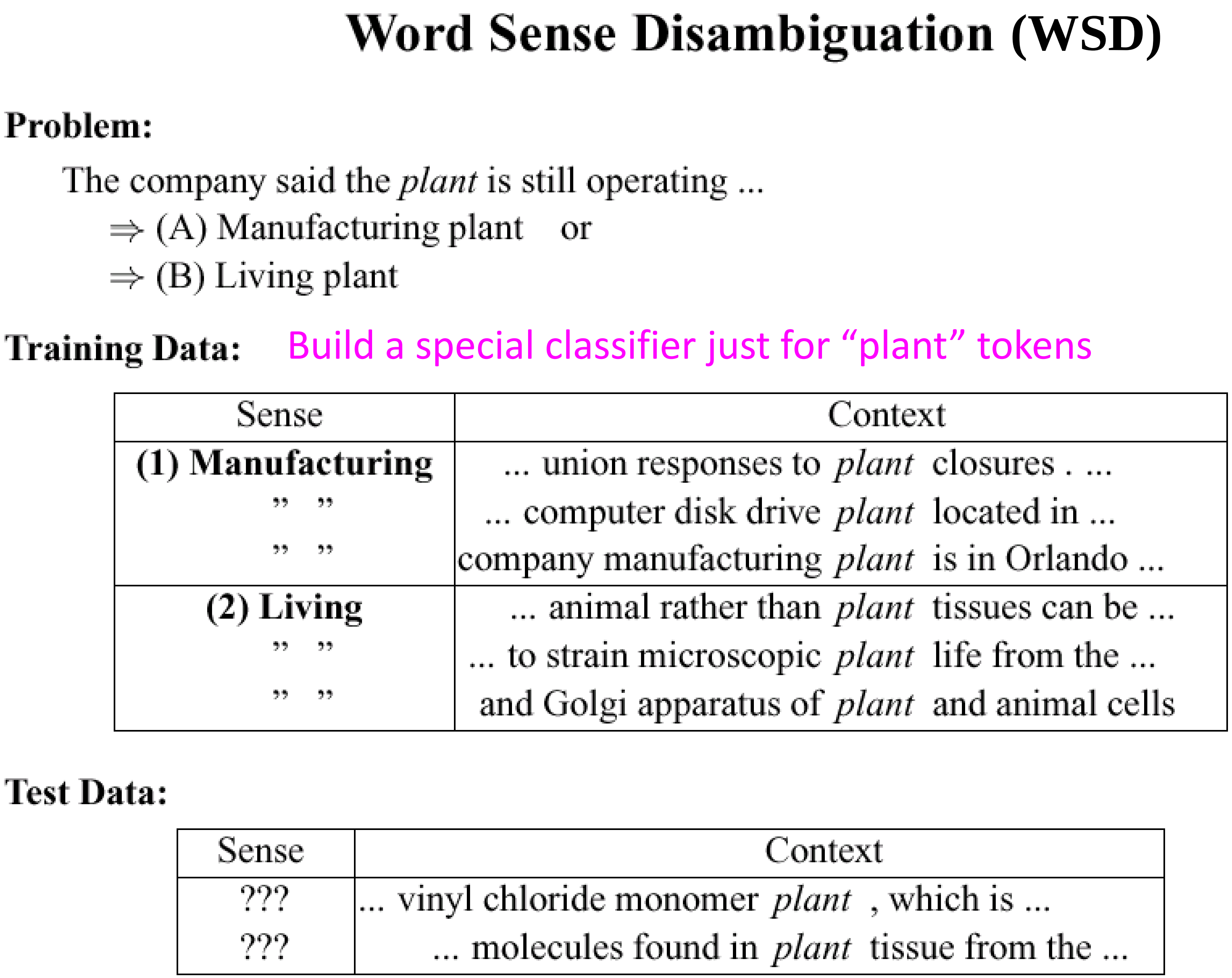
actual class c_1

Text Annotation Tasks (“Classification” Tasks)

1. Classify the entire document (“text categorization”)
2. Classify word tokens individually
3. Classify word tokens in a sequence
4. Identify phrases (“chunking”)
5. Syntactic annotation (parsing)
6. Semantic annotation

Slide courtesy Jason Eisner, with mild edits

p(class | token in context)



Word Sense Disambiguation (WSD)

Problem:

The company said the *plant* is still operating ...

⇒ (A) Manufacturing plant or

⇒ (B) Living plant

Training Data: Build a special classifier just for “plant” tokens

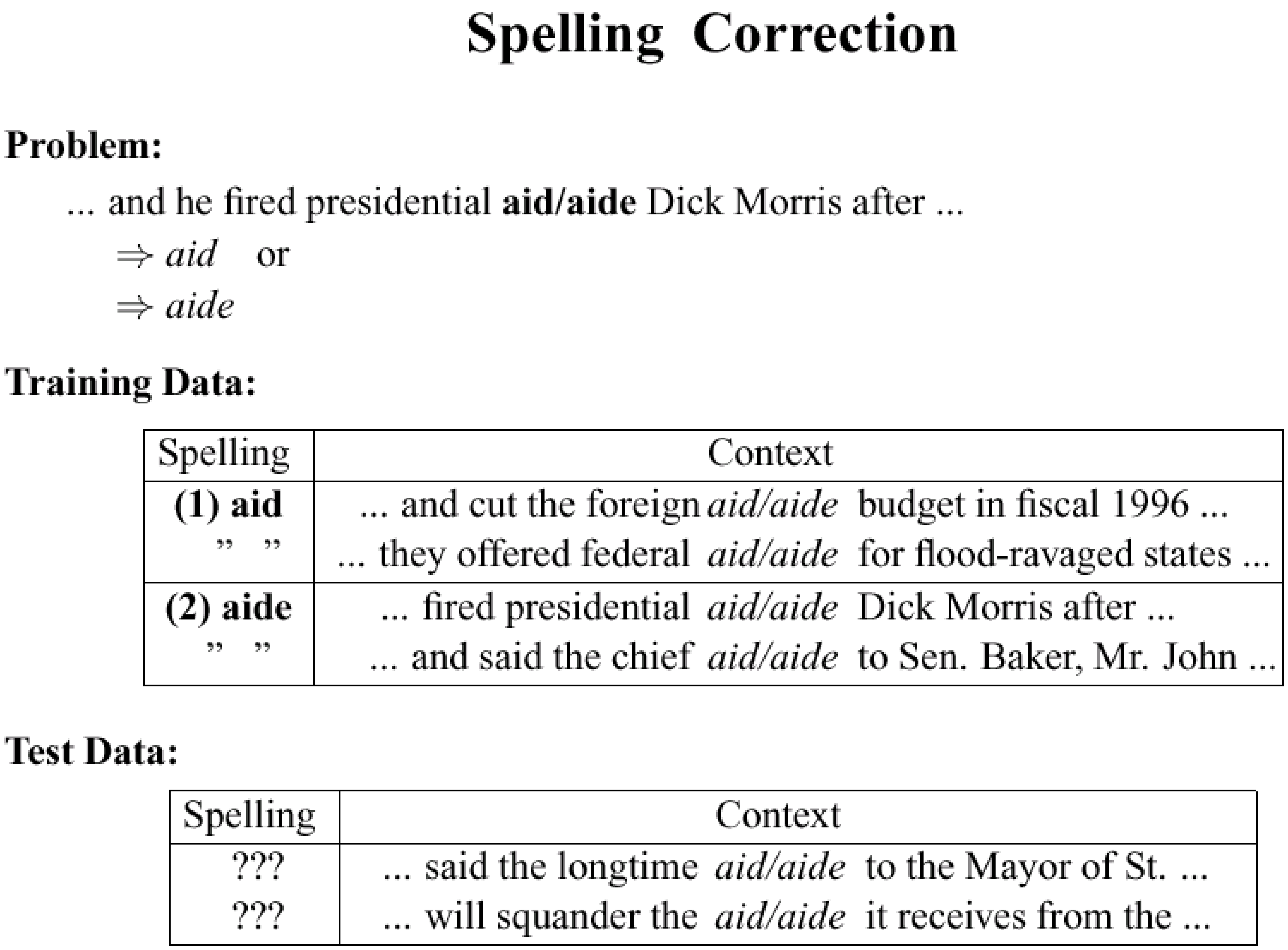
Sense	Context
(1) Manufacturing	... union responses to *plant* closures . ...
” ”	... computer disk drive *plant* located in ...
” ”	company manufacturing *plant* is in Orlando ...
(2) Living	... animal rather than *plant* tissues can be ...
” ”	... to strain microscopic *plant* life from the ...
” ”	and Golgi apparatus of *plant* and animal cells

Test Data:

Sense	Context
???	... vinyl chloride monomer *plant* , which is ...
???	... molecules found in *plant* tissue from the ...

slide courtesy of D. Yarowsky (modified)

p(class | token in context)



Spelling Correction

Problem:

... and he fired presidential **aid/aide** Dick Morris after ...

⇒ *aid* or

⇒ *aide*

Training Data:

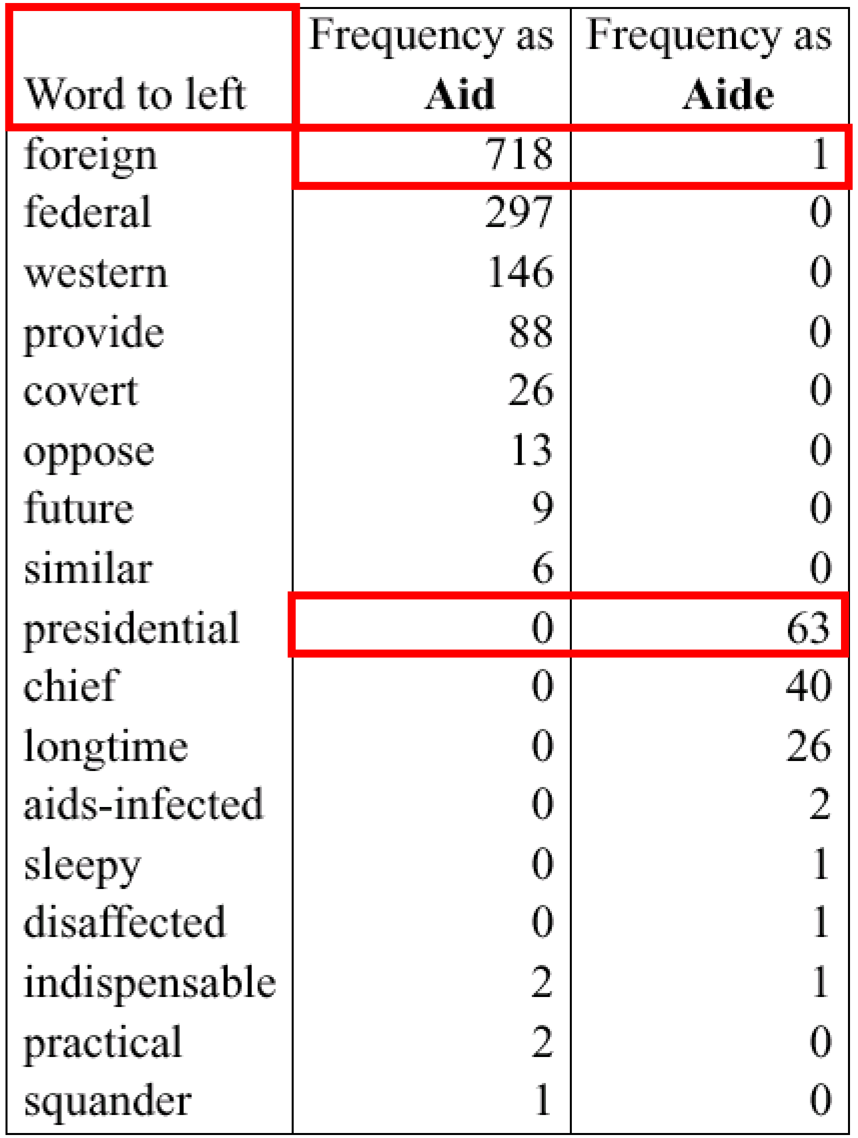
Spelling	Context
(1) aid	... and cut the foreign *aid/aide* budget in fiscal 1996 ...
” ”	... they offered federal *aid/aide* for flood-ravaged states ...
(2) aide	... fired presidential *aid/aide* Dick Morris after ...
” ”	... and said the chief *aid/aide* to Sen. Baker, Mr. John ...

Test Data:

Spelling	Context
???	... said the longtime *aid/aide* to the Mayor of St. ...
???	... will squander the *aid/aide* it receives from the ...

slide courtesy of D. Yarowsky (modified)

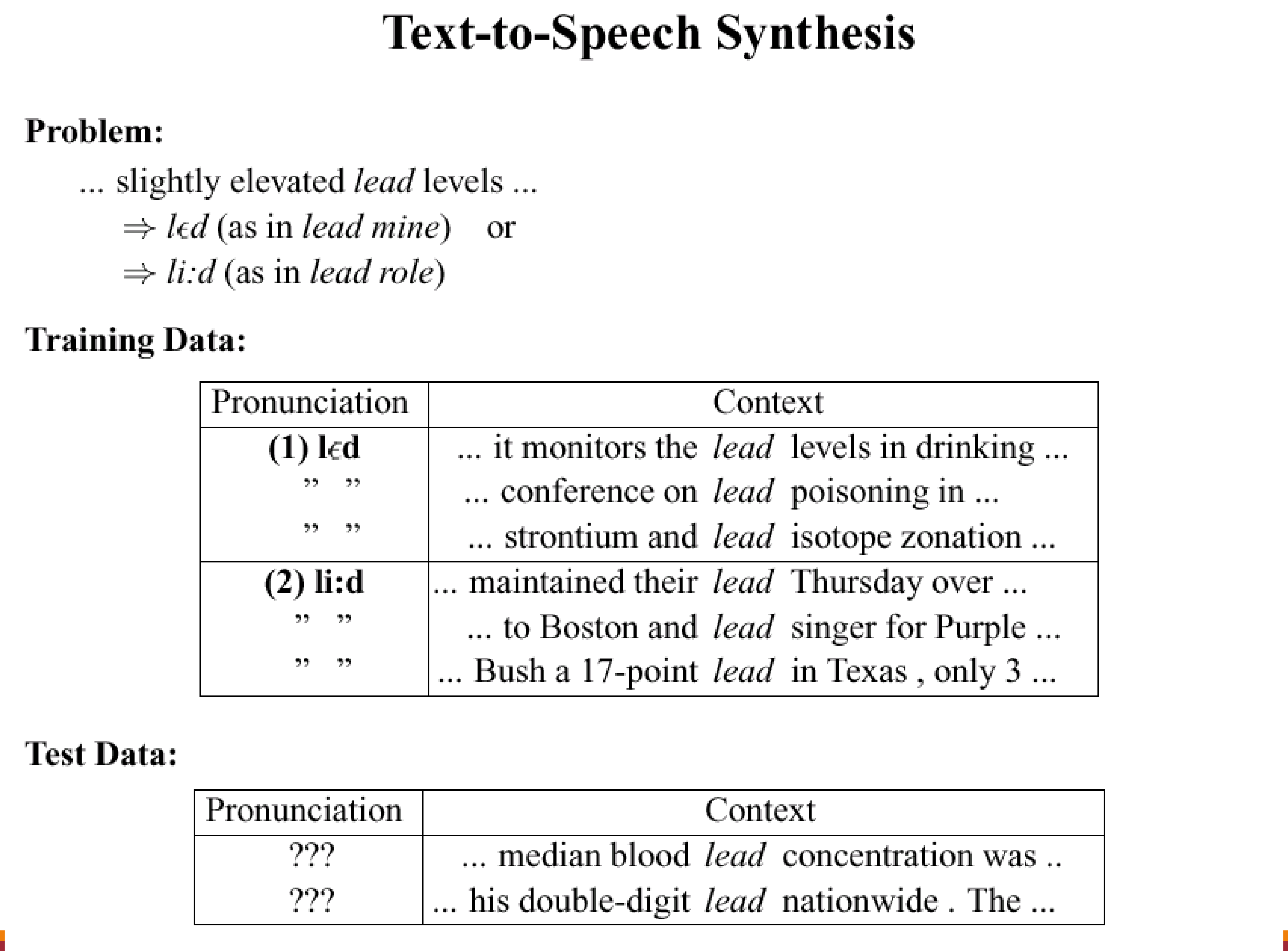
What features? Example: “word to [the] left [of correction]”



Word to left	Frequency as **Aid**	Frequency as **Aide**
foreign	718	1
federal	297	0
western	146	0
provide	88	0
covert	26	0
oppose	13	0
future	9	0
similar	6	0
presidential	0	63
chief	0	40
longtime	0	26
aids-infected	0	2
sleepy	0	1
disaffected	0	1
indispensable	2	1
practical	2	0
squander	1	0

slide courtesy of D. Yarowsky (modified)

p(class | token in context)



Text-to-Speech Synthesis

Problem:

... slightly elevated *lead* levels ...

⇒ *lɛd* (as in *lead mine*) or

⇒ *li:d* (as in *lead role*)

Training Data:

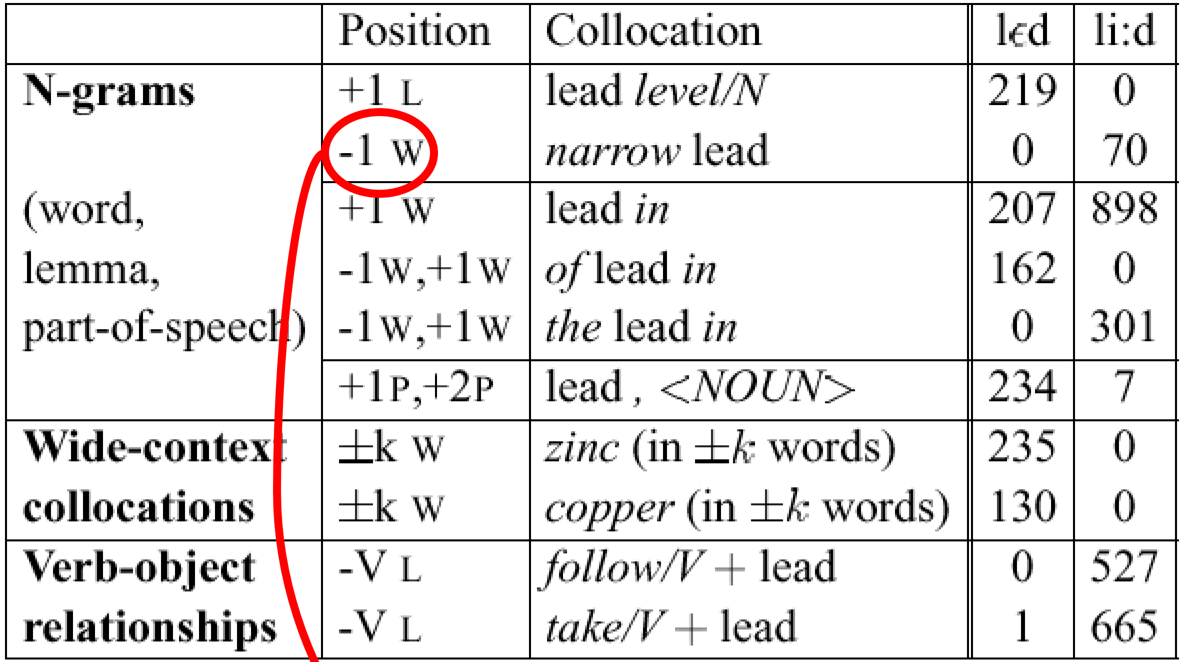
Pronunciation	Context
(1) lɛd	... it monitors the *lead* levels in drinking ...
” ”	... conference on *lead* poisoning in ...
” ”	... strontium and *lead* isotope zonation ...
(2) li:d	... maintained their *lead* Thursday over ...
” ”	... to Boston and *lead* singer for Purple ...
” ”	... Bush a 17-point *lead* in Texas , only 3 ...

Test Data:

Pronunciation	Context
???	... median blood *lead* concentration was ..
???	... his double-digit *lead* nationwide . The ...

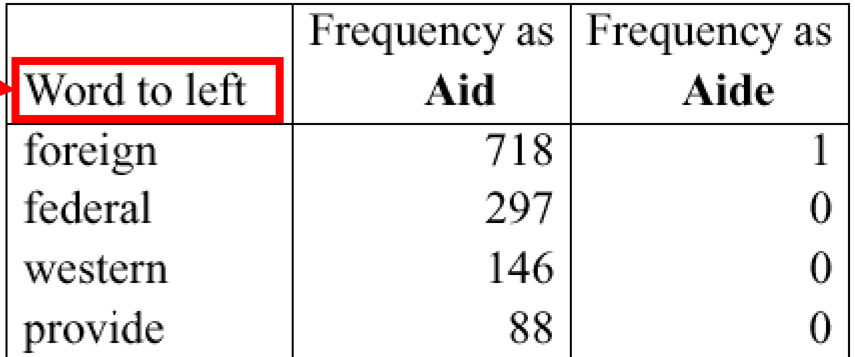
slide courtesy of D. Yarowsky (modified)

An assortment of possible cues …



	Position	Collocation	lɛd	li:d
N-grams	+1 L	lead *level/N*	219	0
	-1 W	*narrow* lead	0	70
(word,	+1 W	lead *in*	207	898
lemma,	-1W,+1W	*of* lead *in*	162	0
part-of-speech)	-1W,+1W	*the* lead *in*	0	301
	+1P,+2P	lead *, <NOUN>*	234	7
Wide-context	±k W	*zinc* (in ±*k* words)	235	0
collocations	±k W	*copper* (in ±*k* words)	130	0
Verb-object	-V L	*follow/V* + lead	0	527
relationships	-V L	*take/V* + lead	1	665

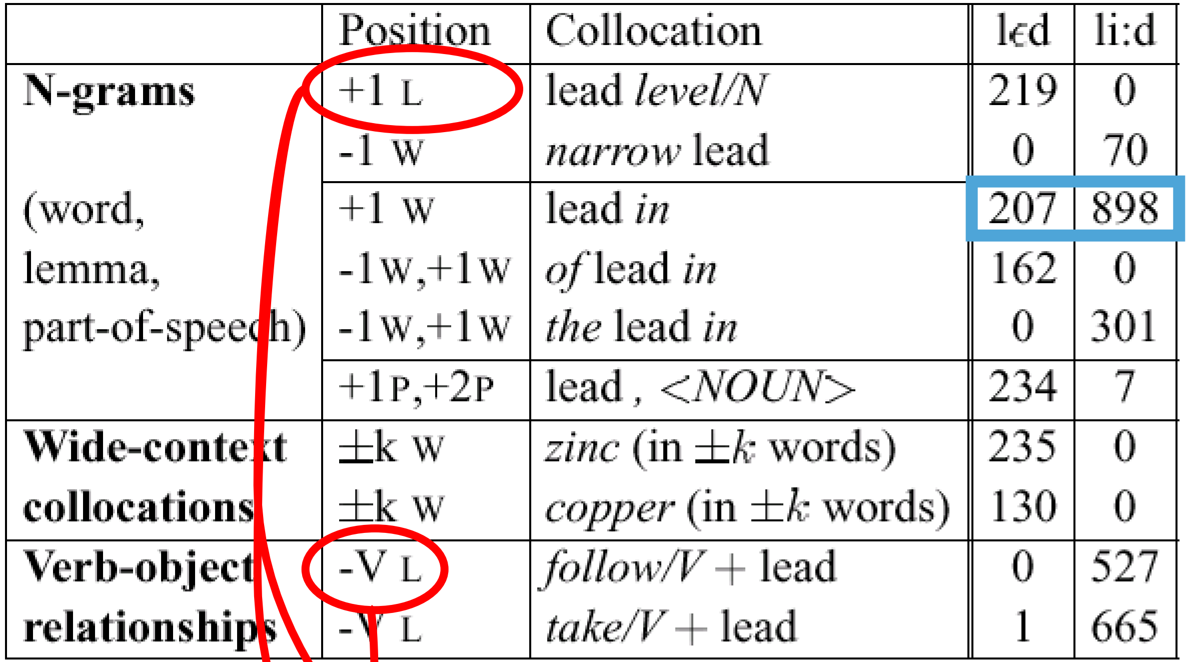
generates a whole bunch of potential cues – use data to find out which ones work best



Word to left	Frequency as **Aid**	Frequency as **Aide**
foreign	718	1
federal	297	0
western	146	0
provide	88	0

slide courtesy of D. Yarowsky (modified)

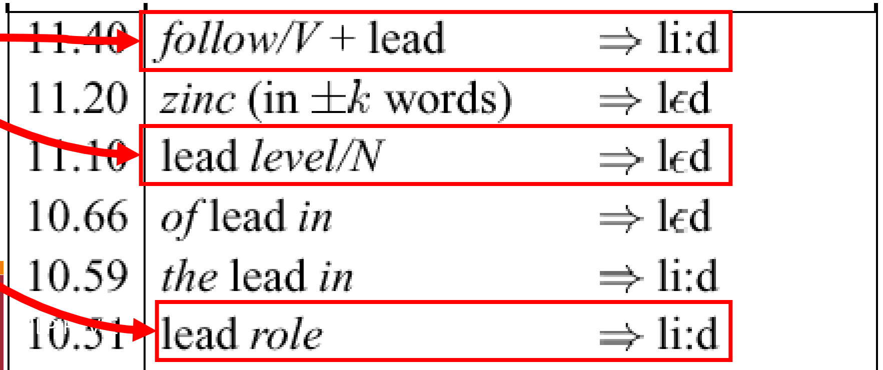
An assortment of possible cues ...



	Position	Collocation	lɛd	li:d
N-grams	+1 L	lead *level/N*	219	0
	-1 W	*narrow* lead	0	70
(word,	+1 W	lead *in*	207	898
lemma,	-1W,+1W	*of* lead *in*	162	0
part-of-speech)	-1W,+1W	*the* lead *in*	0	301
	+1P,+2P	lead , <*NOUN*>	234	7
Wide-context collocations	±k W	*zinc* (in ±*k* words)	235	0
	±k W	*copper* (in ±*k* words)	130	0
Verb-object relationships	-V L	*follow/V* + lead	0	527
	-V L	*take/V* + lead	1	665

This feature is relatively weak, but weak features are still useful, especially since very few features will fire in a given context.

merged ranking of all cues of all these types



11.40	*follow/V* + lead	⇒ li:d
11.20	*zinc* (in ±*k* words)	⇒ lɛd
11.10	lead *level/N*	⇒ lɛd
10.66	*of* lead *in*	⇒ lɛd
10.59	*the* lead *in*	⇒ li:d
10.51	lead *role*	⇒ li:d

slide courtesy of D. Yarowsky (modified)

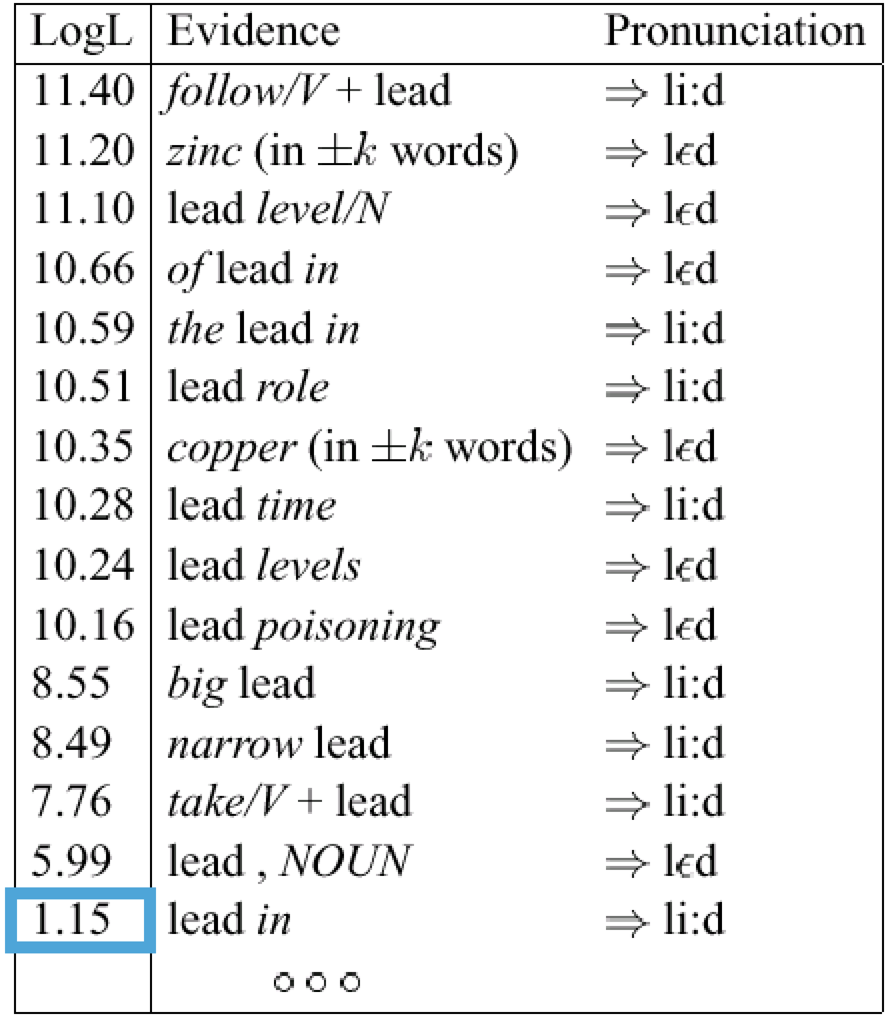
Final decision list for ***lead*** (abbreviated)

What are the input/output?
What are the features?
What types of applications?

List of all features, ranked by their weight.

(These weights are for a simple "decision list" model where the single highest-weighted feature that fires gets to make the decision all by itself.

However, a log-linear model, which adds up the weights of all features that fire, would be roughly similar.)



LogL	Evidence	Pronunciation
11.40	*follow/V* + lead	⇒ li:d
11.20	*zinc* (in $\pm k$ words)	⇒ lɛd
11.10	lead *level/N*	⇒ lɛd
10.66	*of* lead *in*	⇒ lɛd
10.59	*the* lead *in*	⇒ li:d
10.51	lead *role*	⇒ li:d
10.35	*copper* (in $\pm k$ words)	⇒ lɛd
10.28	lead *time*	⇒ li:d
10.24	lead *levels*	⇒ lɛd
10.16	lead *poisoning*	⇒ lɛd
8.55	*big* lead	⇒ li:d
8.49	*narrow* lead	⇒ li:d
7.76	*take/V* + lead	⇒ li:d
5.99	lead , *NOUN*	⇒ lɛd
1.15	lead *in*	⇒ li:d
	○○○	

slide courtesy of D. Yarowsky (modified)

Token Classification

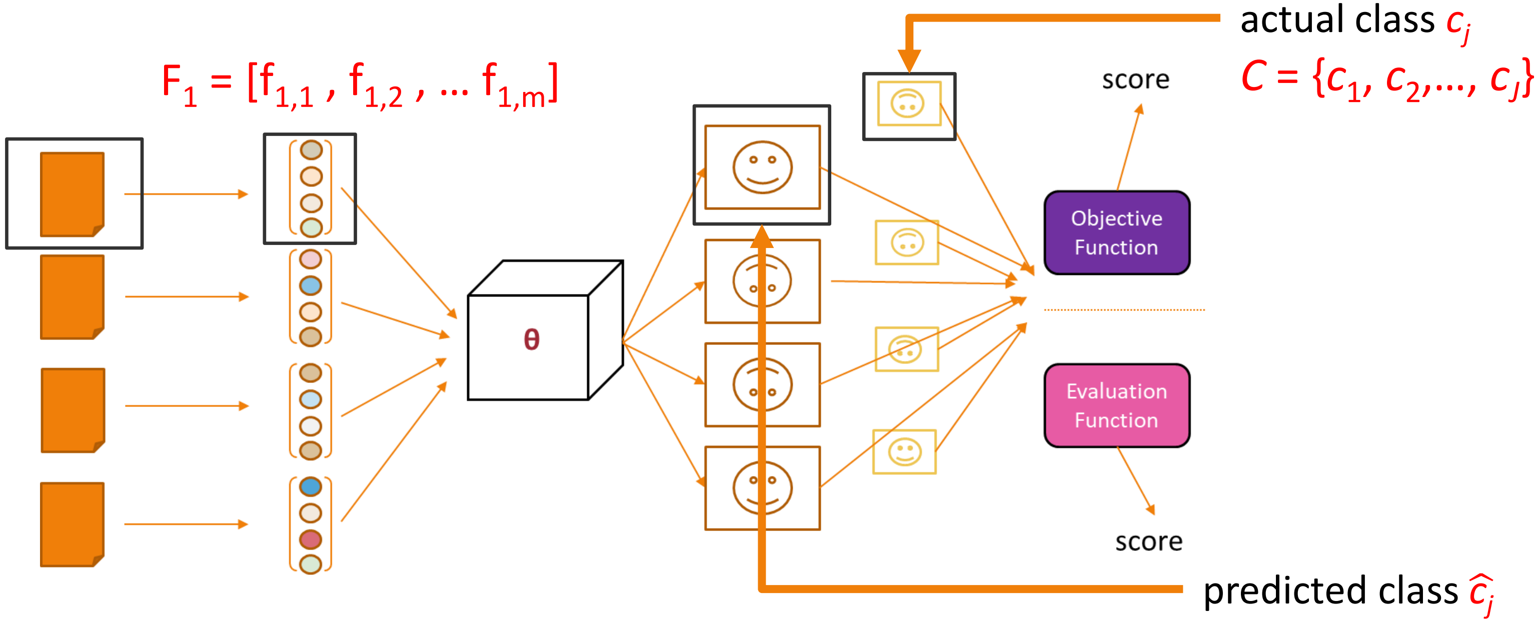
Word pronunciation

Word sense disambiguation (WSD) within or across languages

Accent restoration

Other examples?

features F_1 extracted from word w_1 and its surrounding words (context)



$F_1 = [f_{1,1}, f_{1,2}, \ldots f_{1,m}]$
θ
actual class c_j
$C = \{c_1, c_2, \ldots, c_J\}$
score
Objective Function
Evaluation Function
score
predicted class $\hat{c}_j$

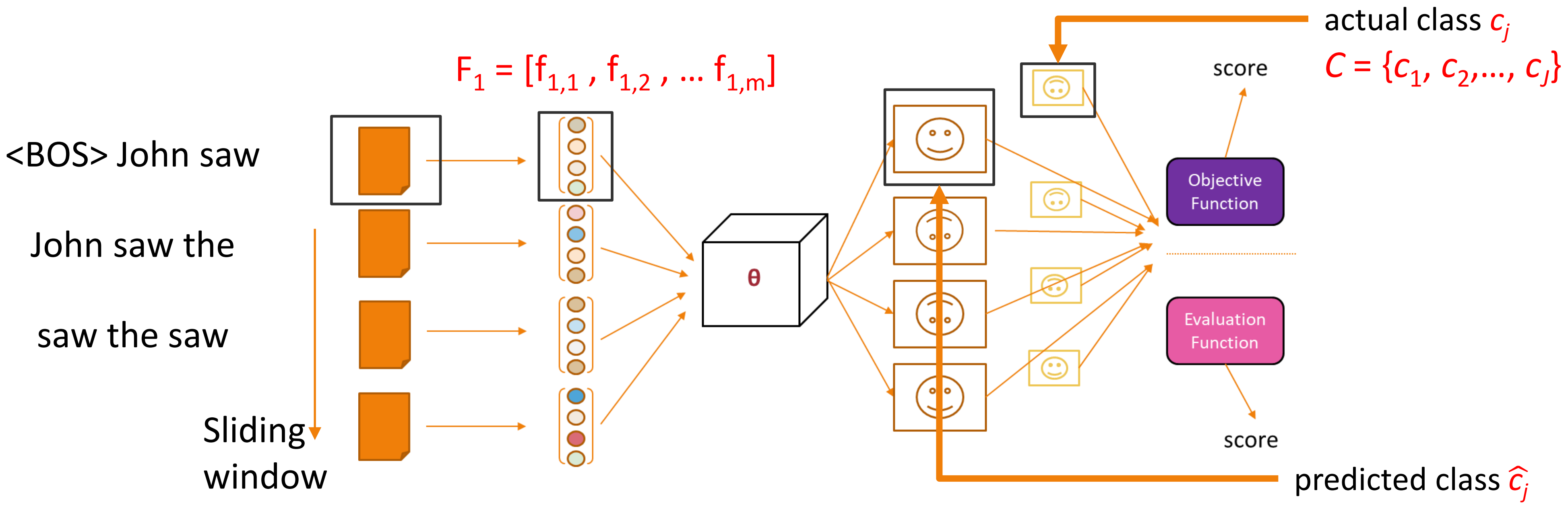
Text Annotation Tasks (“Classification” Tasks)

1. Classify the entire document (“text categorization”)
2. Classify word tokens individually
3. Classify word tokens in a sequence (i.e., order matters)
4. Identify phrases (“chunking”)
5. Syntactic annotation (parsing)
6. Semantic annotation

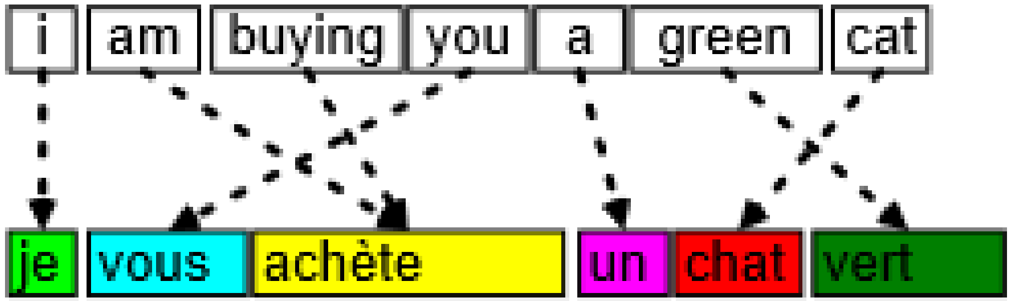
Slide courtesy Jason Eisner, with mild edits

Part of Speech Tagging

<BOS>	John	saw	the	saw	and	decided	to	take	it	to	the	table	.
	NNP	VBD	DT	NN	CC	VBD	TO	VB	PRP	IN	DT	NN	PUNCT



Machine Translation: Word Alignment



https://towardsdatascience.com/machine-translation-a-short-overview-91343ff39c9f

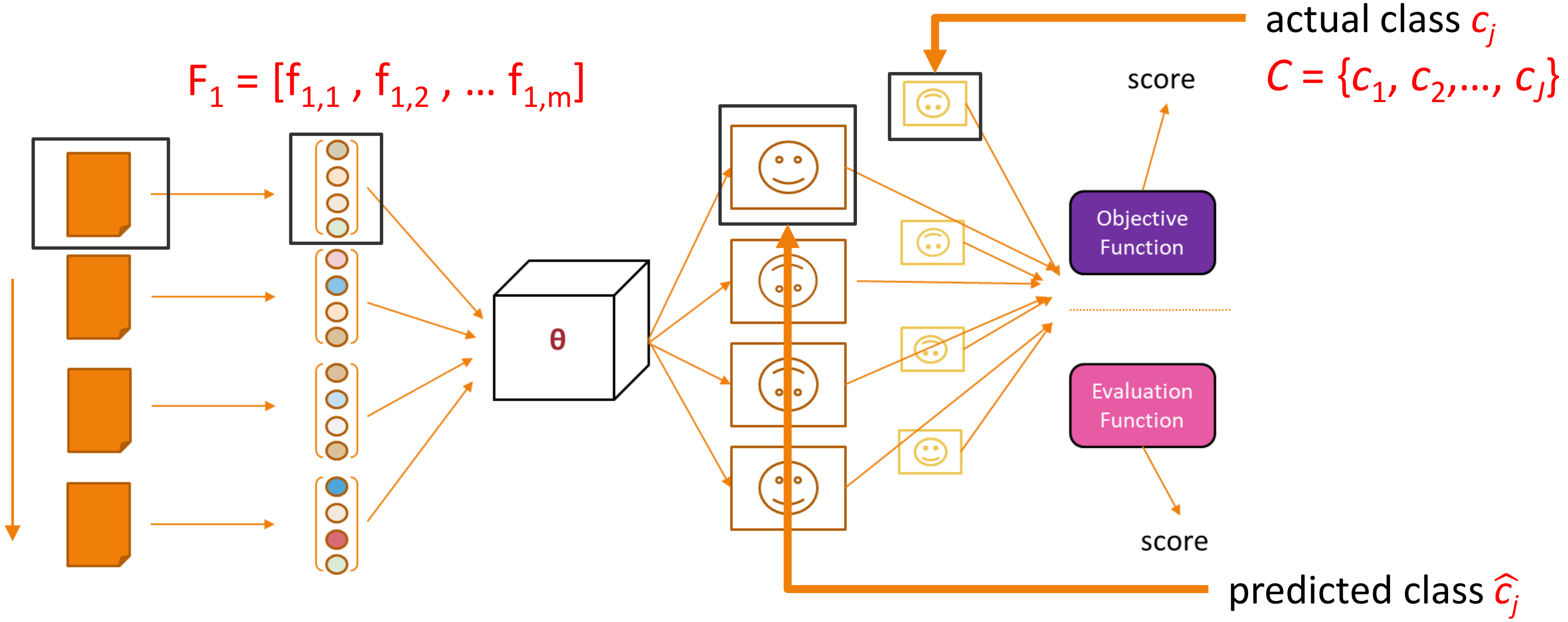
Token Classification in a Sequence

Part of speech tagging

Word alignment

features F_1 extracted from word w_1 and its surrounding words (context)

Sliding window



$F_1 = [f_{1,1}, f_{1,2}, \ldots f_{1,m}]$

θ

actual class c_j

$C = \{c_1, c_2, \ldots, c_J\}$

score

Objective Function

Evaluation Function

score

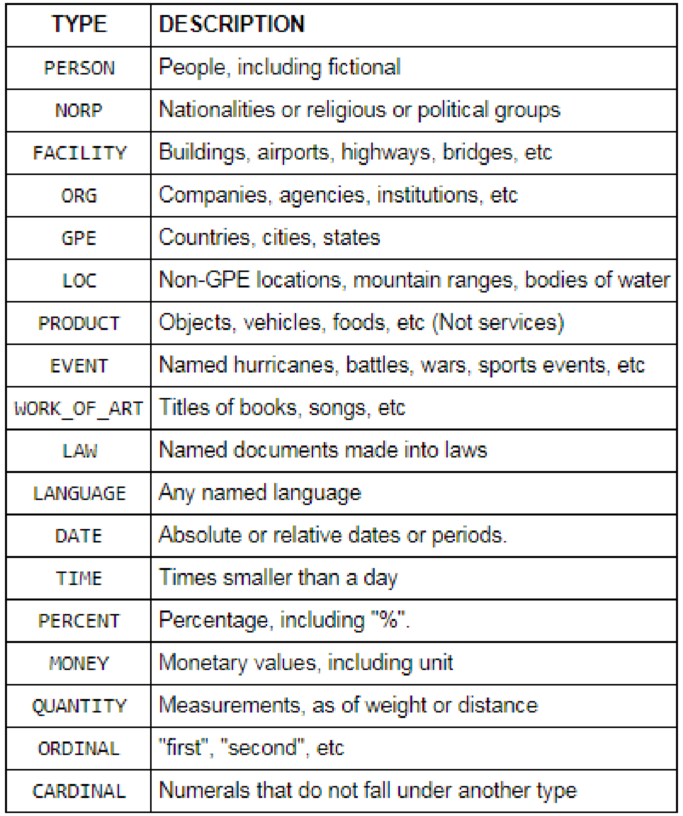
predicted class $\hat{c}_j$

Text Annotation Tasks (“Classification” Tasks)

1. Classify the entire document (“text categorization”)
2. Classify word tokens individually
3. Classify word tokens in a sequence
4. Identify phrases (“chunking”)
5. Syntactic annotation (parsing)
6. Semantic annotation

Slide courtesy Jason Eisner, with mild edits

Named Entity Recognition



TYPE	DESCRIPTION
PERSON	People, including fictional
NORP	Nationalities or religious or political groups
FACILITY	Buildings, airports, highways, bridges, etc
ORG	Companies, agencies, institutions, etc
GPE	Countries, cities, states
LOC	Non-GPE locations, mountain ranges, bodies of water
PRODUCT	Objects, vehicles, foods, etc (Not services)
EVENT	Named hurricanes, battles, wars, sports events, etc
WORK_OF_ART	Titles of books, songs, etc
LAW	Named documents made into laws
LANGUAGE	Any named language
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day
PERCENT	Percentage, including "%".
MONEY	Monetary values, including unit
QUANTITY	Measurements, as of weight or distance
ORDINAL	"first", "second", etc
CARDINAL	Numerals that do not fall under another type

CHICAGO (AP) — Citing high fuel prices, United Airlines said Friday it has increased fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit AMR, immediately matched the move, spokesman Tim Wagner said. United, a unit of UAL, said the increase took effect Thursday night and applies to most routes where it competes against discount carriers, such as Chicago to Dallas and Atlanta and Denver to San Francisco, Los Angeles and New York.

Slide courtesy Jim Martin

Example Use: Information Extraction

As a task: **Filling slots in a database from sub-segments of text.**

Note:
IE is a task on its own but it can be an application of NER

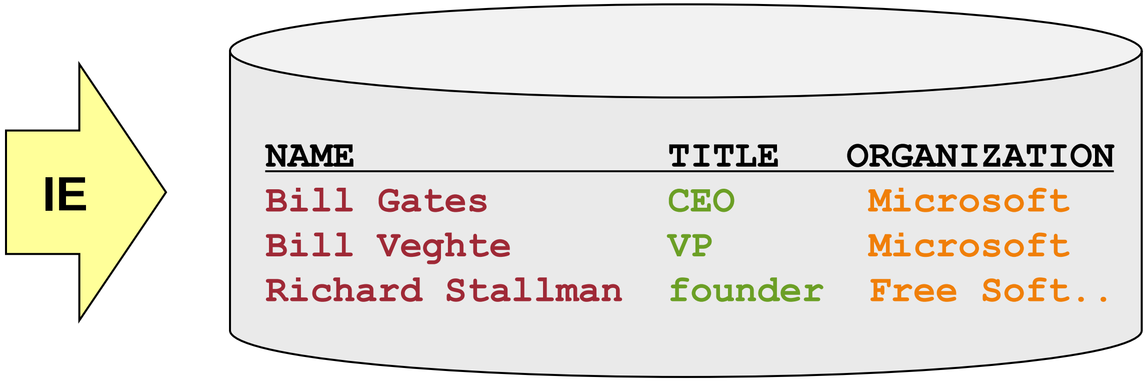
October 14, 2002, 4:00 a.m. PT

For years, Microsoft Corporation CEO Bill Gates railed against the economic philosophy of open-source software with Orwellian fervor, denouncing its communal licensing as a "cancer" that stifled technological innovation.

Today, Microsoft claims to "love" the open-source concept, by which software code is made public to encourage improvement and development by outside programmers. Gates himself says Microsoft will gladly disclose its crown jewels--the coveted code behind the Windows operating system--to select customers.

"We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access.“

Richard Stallman, founder of the Free Software Foundation, countered saying…



NAME	TITLE	ORGANIZATION
Bill Gates	CEO	Microsoft
Bill Veghte	VP	Microsoft
Richard Stallman	founder	Free Soft..

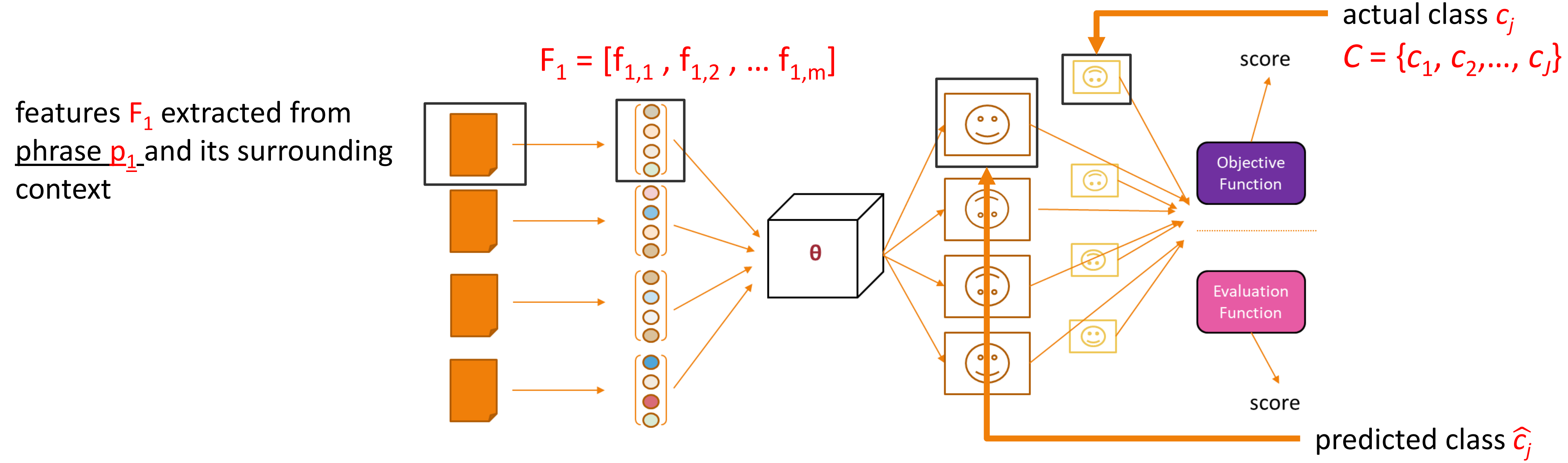
Slide from Chris Brew, adapted from slide by William Cohen

Chunking

Named entity recognition

Information extraction

Identifying idioms



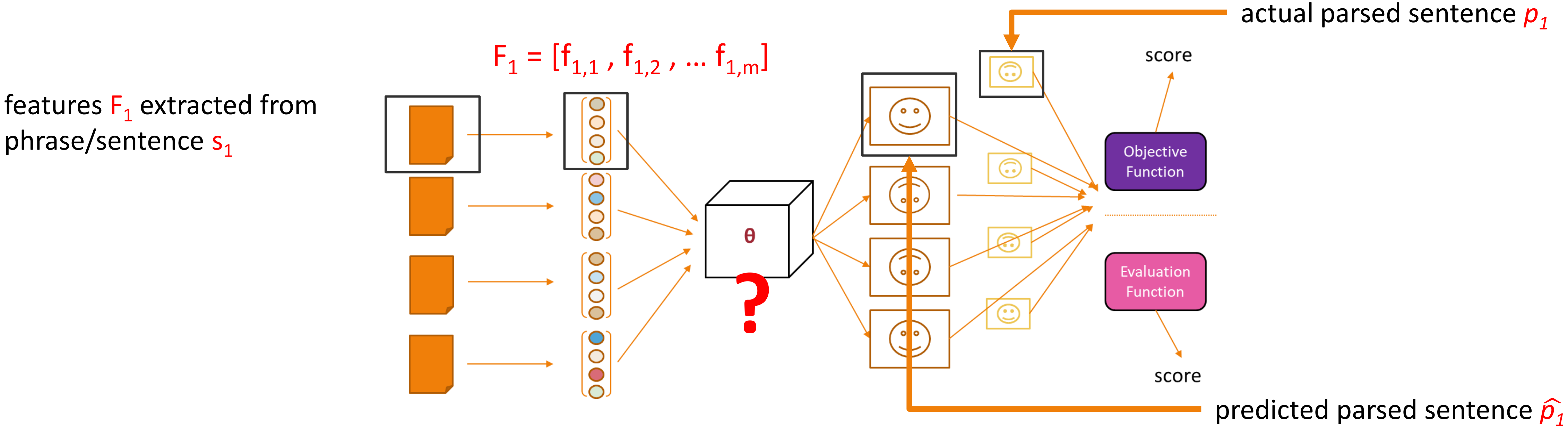
features F_1 extracted from phrase p_1 and its surrounding context
$F_1 = [f_{1,1} , f_{1,2} , \ldots f_{1,m}]$
θ
actual class c_j
$C = \{c_1, c_2, \ldots, c_J\}$
score
Objective Function
Evaluation Function
score
predicted class $\hat{c}_j$

Text Annotation Tasks (“Classification” Tasks)

1. Classify the entire document (“text categorization”)
2. Classify word tokens individually
3. Classify word tokens in a sequence
4. Identify phrases (“chunking”)
5. Syntactic annotation (syntax parsing)
6. Semantic annotation

Slide courtesy Jason Eisner, with mild edits

Syntax Parsing



features F_1 extracted from phrase/sentence s_1

$F_1 = [f_{1,1}, f_{1,2}, \dots f_{1,m}]$

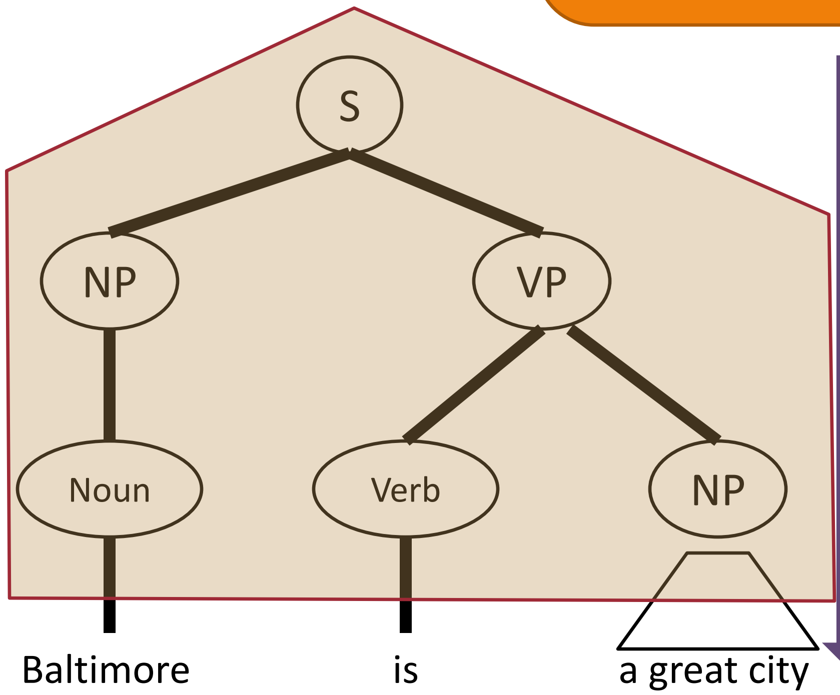
actual parsed sentence p_1

predicted parsed sentence $\hat{p}_1$

Assign Structure (Parse) with a Context Free Grammar

S → NP VP
NP → Det Noun
NP → Noun
NP → Det AdjP
NP → NP PP

PP → P NP
AdjP → Adj Noun
VP → V NP
Noun → Baltimore
…



Baltimore is a great city

[$_{S}$ [$_{NP}$ [$_{Noun}$ Baltimore]] [$_{VP}$ [$_{Verb}$ is] [$_{NP}$ a great city]]]

bracket notation

```
(S (NP (Noun Baltimore))
 (VP (V is)
   (NP a great city)))
```

S-expression

Text Annotation Tasks (“Classification” Tasks)

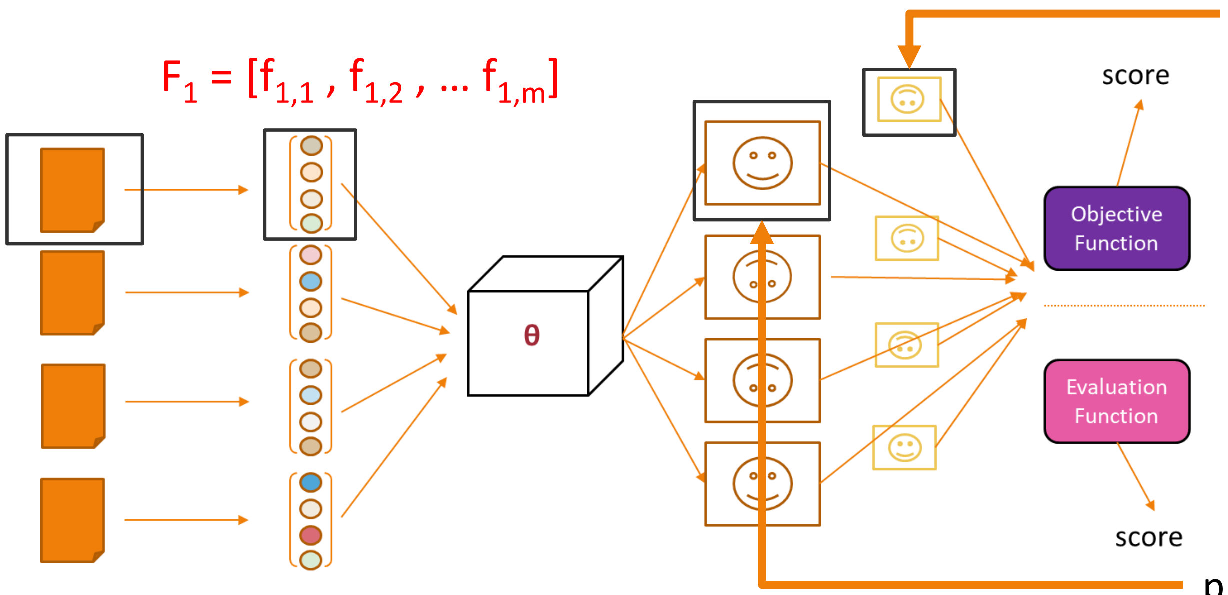
1. Classify the entire document (“text categorization”)
2. Classify word tokens individually
3. Classify word tokens in a sequence
4. Identify phrases (“chunking”)
5. Syntactic annotation (syntax parsing)
6. Semantic annotation

Slide courtesy Jason Eisner, with mild edits

Semantic Parsing

Semantic role labeling (SRL)

features F_1 extracted from phrase/sentence s_1 and its surrounding context



$F_1 = [f_{1,1} , f_{1,2} , \ldots f_{1,m}]$

actual semantic parse p_1

predicted semantic parse $\hat{p}_1$

What is a semantic parse?

Semantic Role Labeling (SRL)

For each predicate (e.g., verb)

1. find its arguments (e.g., NPs)
2. determine their **semantic roles**

John drove Mary from Austin to Dallas in his Toyota Prius.

- agent: Actor of an action
- patient: Entity affected by the action
- source: Origin of the affected entity
- destination: Destination of the affected entity
- instrument: Tool used in performing action.
- beneficiary: Entity for whom action is performed

Slide thanks to Ray Mooney (modified)

Semantic Role Labeling (SRL)

For each predicate (e.g., verb)

1. find its arguments (e.g., NPs)
2. determine their **semantic roles**

John drove Mary from Austin to Dallas in his Toyota Prius.

agent patient source destination instrument

Slide thanks to Ray Mooney (modified)