TTIC 31210:
Advanced Natural Language Processing

Kevin Gimpel

Spring 2017

Lecture 1: Introduction

- Please email me with the following:
  - name
  - email address
  - whether you are taking the course for credit

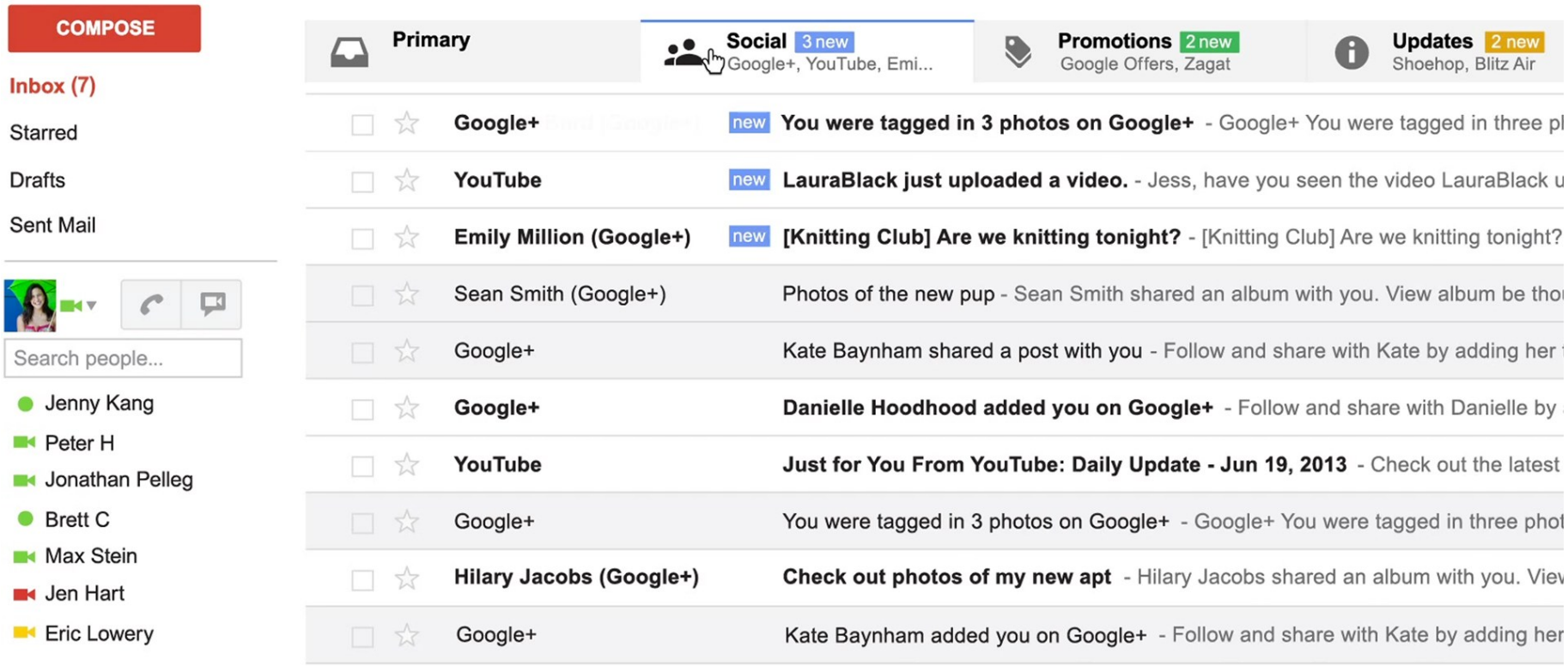
- I will use the email addresses for the course mailing list

What is natural language processing?

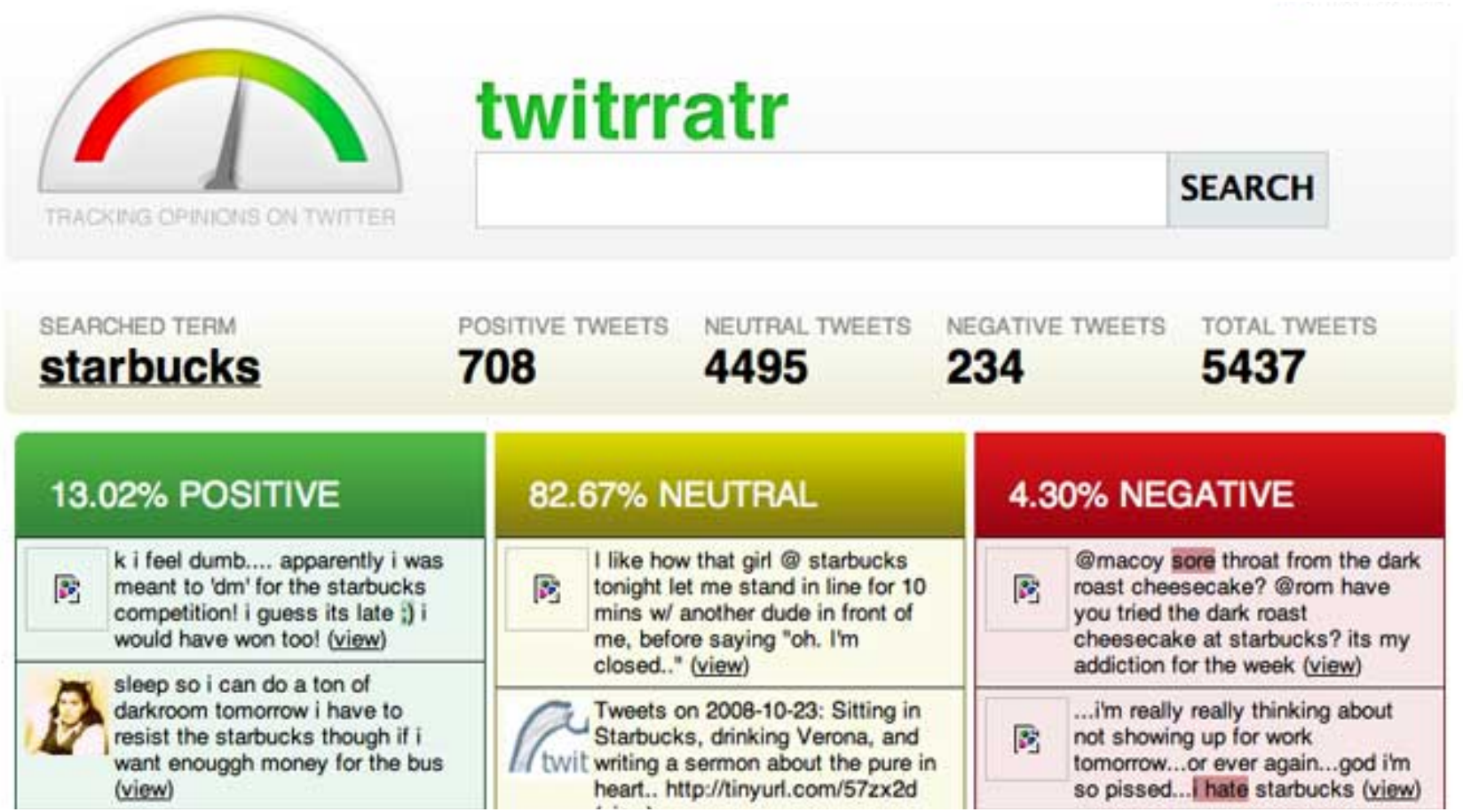
What is natural language processing?

an experimental computer science research area that includes problems and solutions pertaining to the understanding of human language

Text Classification



Sentiment Analysis



Machine Translation

14:11 Uhr · Apple Watch · fen

Neue Umfrage: Kaufen Sie eine Apple Watch?

Seit gestern ist auch die genaue Preisstruktur der Apple Watch bekannt und viele Nutzer **befassen sich daher mit der Frage**, ob sie eine Apple

New Poll: Will you buy an Apple Watch?

von Ihnen wissen, ob Sie schon eine Entscheidung getroffen haben - wird Ihre nächste Uhr eine Apple Watch und welches der drei Grundmodelle soll es dann sein? Oder hat Apple keine Chance, Sie als Käufer begrüßen zu können? Eine detaillierte Preisübersicht hatten wir in diesem Artikel zusammengestellt: @

Question Answering

Summarization

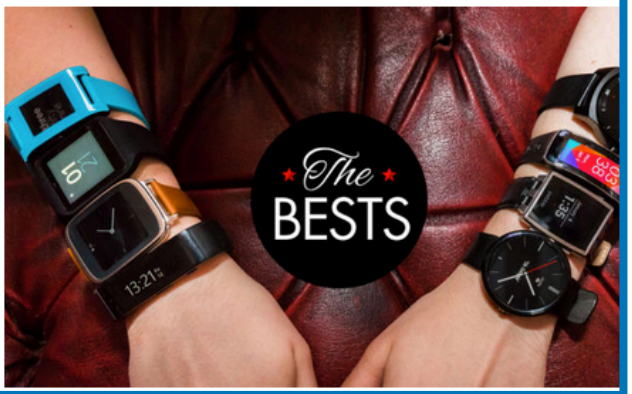
GIZMODO

+ FOLLOW

Eric Limer
Filed to: SMARTWATCHES Monday 4:31pm

175,377

The Best Smartwatches That Aren't the Apple Watch



Five things the Pebble Time can do that the Apple Watch can't

Summary: *The new Apple Watch isn't the only smartwatch to consider and if you own an iPhone then you should consider what the Pebble Time offers. Matthew lists five things to consider.*

By Matthew Miller for The Mobile Gadgeteer | March 12, 2015 -- 14:25 GMT (07:25 PDT)

Follow @palmsolo 8,013 followers

Get the ZDNet Microsoft newsletter now

Comments 5 | Share on Facebook 1 | Tweet 81 | Share 6 | more +

Apple Watch Has Big Drawbacks Interface, Reviews Say

reactions so far.

3.8K

11 twitter | 17 facebook | send via email | share

ted Apple Watch — a product developed behind a shroud of PR control and
dy for prime time. And reviews of the Apple Watch are pouring in. But a
pressions are not great.

The Apple Watch has drawbacks. There are other smartwatches that offer more capabilities.

Dialog Systems

user: Schedule a meeting with Matt and David on Thursday.
computer: Thursday won't work for David. How about Friday?
user: I'd prefer Monday then, but Friday would be ok if necessary.

Part-of-Speech Tagging

Some questioned if Tim Cook ’s first product would be a breakaway hit for Apple .

Part-of-Speech Tagging

determiner	verb (past)	prep.	proper noun	proper noun	poss.	adj.	noun
Some	questioned	if	Tim	Cook	’s	first	product

modal	verb	det.	adjective	noun	prep.	proper noun	punc.
would	be	a	breakaway	hit	for	Apple	.

Part-of-Speech Tagging

determiner	verb (past)	prep.	proper noun	proper noun	poss.	adj.	noun
Some	questioned	if	Tim	Cook	’s	first	product

modal	verb	det.	adjective	noun	prep.	proper noun	punc.
would	be	a	breakaway	hit	for	Apple	.

Named Entity Recognition

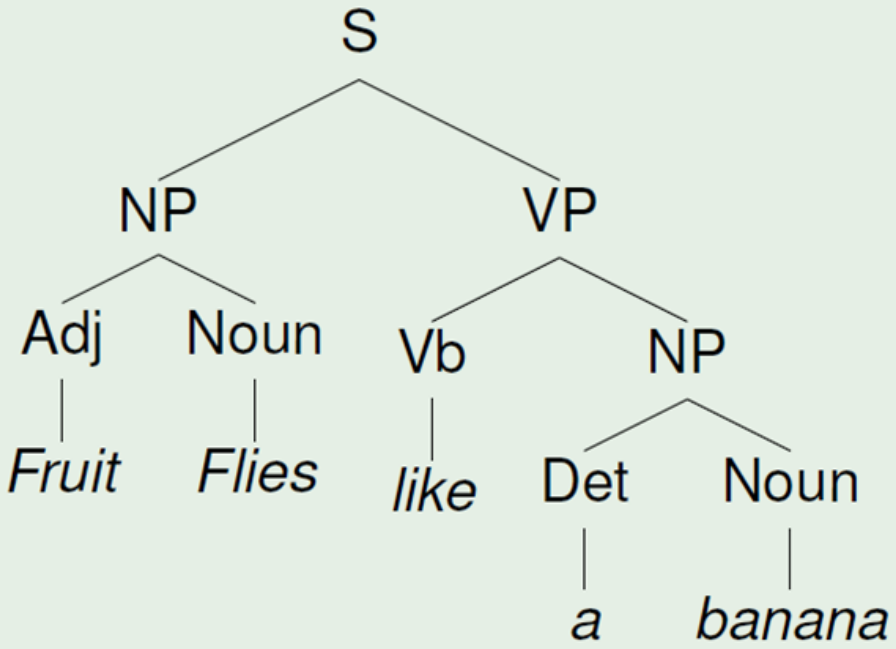
Some questioned if Tim Cook’s first product would be a breakaway hit for Apple.

PERSON **ORGANIZATION**

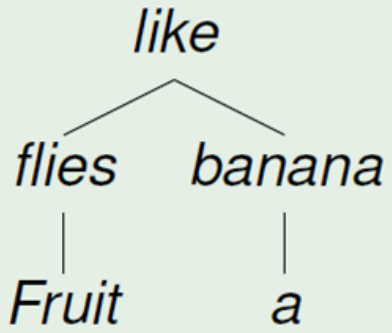
Syntactic Parsing

Fruit flies like a banana

Constituency Structure



Dependency Structure



Entity Linking

Coreference Resolution

figure credit: Durrett & Klein (2014)

“Winograd Schema” Coreference Resolution

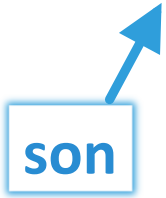
The man couldn't lift his son because **he** was so weak.

The man couldn't lift his son because **he** was so heavy.

"Winograd Schema" Coreference Resolution

The man couldn't lift his son because **he** was so weak.

The man couldn't lift his son because **he** was so heavy.



Reading Comprehension

Once there was a boy named Fritz who loved to draw. He drew everything. In the morning, he drew a picture of his cereal with milk. His papa said, “Don’t draw your cereal. Eat it!”

After school, Fritz drew a picture of his bicycle. His uncle said, “Don't draw your bicycle. Ride it!”

…

What did Fritz draw first?

A) the toothpaste

B) his mama

C) cereal and milk

D) his bicycle

Course Overview

- New course, first time being offered
- Prerequisite: TTIC 31190 (NLP)
- Aimed at senior graduate students
- My office hours: by appointment, TTIC 531
- Teaching assistant: John Wieting, TTIC PhD student

Grading

- 3 assignments (10%, 15%, 15%)
- course project (30%)
- class participation (30%)
- no final

Course Philosophy

- goal: use our time well
  - maximum amount learned for minimum time investment
- our in-class time is very important

Course Philosophy

- goal: use our time well
  - maximum amount learned for minimum time investment
- our in-class time is very important
- some class meetings will be more interactive, involving programming exercises, pen-and-paper exercises, data analysis in small groups

Class Participation

- class participation is worth 30%
- your participation grade $\propto$ number of wrong answers you give

- if you have good reason to miss class, let me know!

Assignments

- Mini-research projects: formal exposition, implementation, experimentation, analysis, developing new methods
- Assignment 1 has been posted; due April 10
- It's a (relatively) short warm-up assignment that will help you catch up if you didn't take the prerequisite

Project

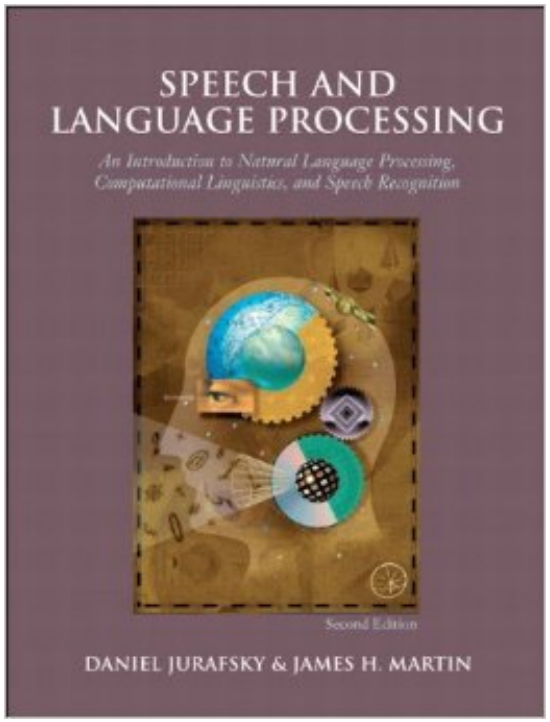
- Replicate [part of] a published NLP paper, or define your own project.
- The project may be done individually or in a group of two. Each group member will receive the same grade.
- More details to come.

Collaboration Policy

- You are welcome to discuss assignments with others in the course, but solutions and code must be written individually

Textbooks

- All are optional
- Speech and Language Processing, 2nd Ed.
  - some chapters of 3rd edition are online
- Bayesian Analysis in NLP by Shay Cohen
  - will be available in the TTIC library



Roadmap

- review of TTIC 31190 (week 1)
- deep learning for NLP (weeks 2-4)
- generative models & Bayesian inference (week 5)
- Bayesian nonparametrics in NLP (week 6)
- EM for unsupervised NLP (week 7)
- syntax/semantics and structure prediction (weeks 8-9)
- applications (week 10)

The First Couple Weeks

- I will be away Wed. March 29 and Wed. April 5
- Sorry about this ☹
- Wed. March 29:
  - Class will be optional
  - TA will hold an office hour during class for anyone who has questions about Assignment 1, deep learning toolkits, python, TTIC 31190, etc.
- Wed. April 5:
  - Class will be canceled

Why is NLP hard?

- ambiguity and variability of linguistic expression:
  - **variability**: many forms can mean the same thing
  - **ambiguity**: one form can mean many things

- there are many different kinds of ambiguity
- each NLP task has to address a distinct set of kinds

What is a classifier?

- a function from inputs *x* to classification labels *y*

What is a classifier?

- a function from inputs *x* to classification labels *y*
- one simple type of classifier:
  - for any input *x*, assign a score to each label *y*, parameterized by vector $\boldsymbol{\theta}$:

$$\text{score}(x, y, \boldsymbol{\theta})$$

What is a classifier?

- a function from inputs *x* to classification labels *y*
- one simple type of classifier:
  - for any input *x*, assign a score to each label *y*, parameterized by vector $\boldsymbol{\theta}$:

$$\text{score}(x, y, \boldsymbol{\theta})$$

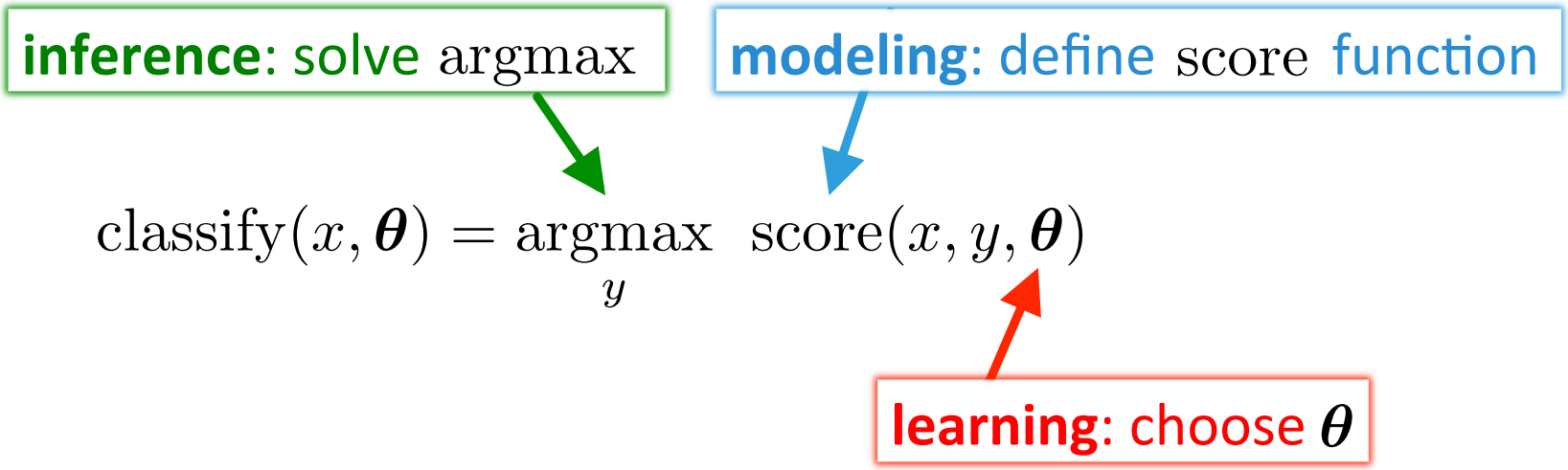
  - classify by choosing highest-scoring label:

$$\text{classify}(x, \boldsymbol{\theta}) = \underset{y}{\text{argmax}} \ \ \text{score}(x, y, \boldsymbol{\theta})$$

Modeling, Inference, Learning

$$\text{classify}(x, \boldsymbol{\theta}) = \underset{y}{\text{argmax}} \;\; \text{score}(x, y, \boldsymbol{\theta})$$

Modeling, Inference, Learning



- **We will use this same paradigm throughout the course, even when the output space size is exponential in the size of the input or is unbounded (e.g., machine translation)**

Notation

- We'll use boldface for vectors:

$$\boldsymbol{\theta}$$

- Individual entries will use subscripts and no boldface, e.g., for entry i:

$$\theta_i$$

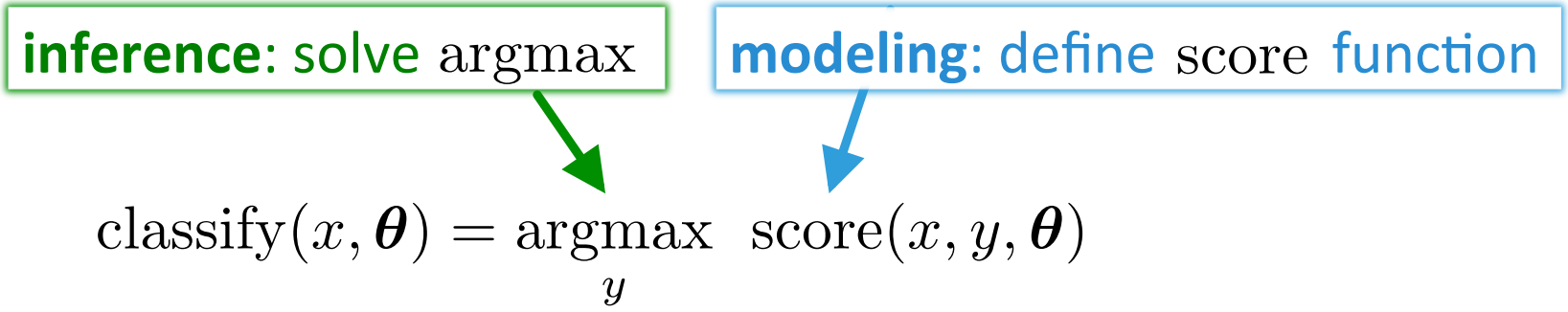
Modeling, Inference, Learning

modeling: define score function

$$\text{classify}(x, \boldsymbol{\theta}) = \underset{y}{\text{argmax}} \;\; \text{score}(x, y, \boldsymbol{\theta})$$

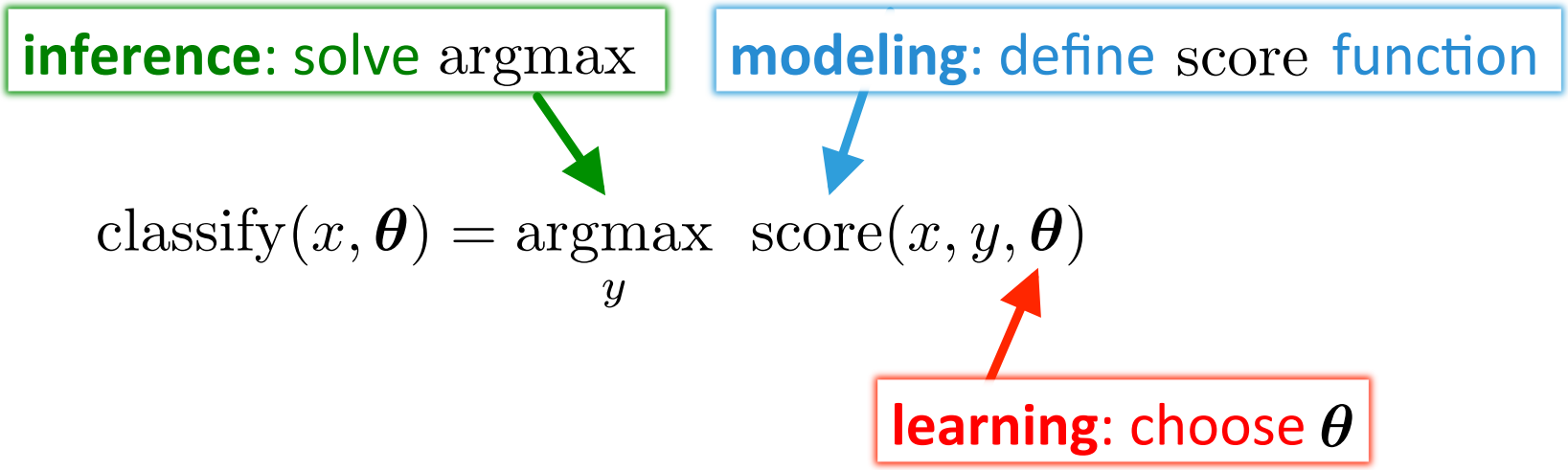
- **Modeling**: How do we assign a score to an (*x*,*y*) pair using parameters $\boldsymbol{\theta}$?

Modeling, Inference, Learning



- **Inference**: How do we efficiently search over the space of all labels?

Modeling, Inference, Learning



- **Learning**: How do we choose $\boldsymbol{\theta}$?

Applications of our Classification Framework

text classification:

$$\text{classify}_{\text{text}}^{\text{linear}}(\boldsymbol{x}, \boldsymbol{\theta}) = \underset{y \in \mathcal{L}}{\text{argmax}} \sum_i \theta_i f_i(\boldsymbol{x}, y)$$

$\mathcal{L}$ = {objective, subjective}

x	*y*
the hulk is an anger fueled monster with incredible strength and resistance to damage .	objective
in trying to be daring and original , it comes off as only occasionally satirical and never fresh .	subjective

Applications of our Classification Framework

word sense classifier for *bass*:

$$\text{classify}^{\text{linear}}_{\text{bassWSD}}(\boldsymbol{x}, \boldsymbol{\theta}) = \underset{y \in \mathcal{L}_{\text{bass}}}{\text{argmax}} \sum_i \theta_i f_i(\boldsymbol{x}, y)$$

$\mathcal{L}_{\text{bass}}$ = {$bass_1$, $bass_2$, ..., $bass_8$}

x	*y*
he's a bass in the choir .	$bass_3$
our bass is line-caught from the Atlantic .	$bass_4$

- S: (n) **bass** (the lowest part of the musical ra
- S: (n) **bass**, bass part (the lowest part in pol
- **S: (n) bass, basso (an adult male singer w**
- S: (n) sea bass, **bass** (the lean flesh of a salt Serranidae)
- S: (n) freshwater bass, **bass** (any of various with lean flesh (especially of the genus Micr
- S: (n) **bass**, bass voice, basso (the lowest ad
- S: (n) **bass** (the member with the lowest ran instruments)
- S: (n) **bass** (nontechnical name for any of nu freshwater spiny-finned fishes)

Applications of our Classification Framework

skip-gram model as a classifier:

$$\text{classify}_{\text{skipgram}}(x, \boldsymbol{\theta}) = \underset{y \in \mathcal{L}}{\text{argmax}} \ \boldsymbol{\theta}^{(\text{in},x)} \cdot \boldsymbol{\theta}^{(\text{out},y)}$$

$\mathcal{L}$ = *V* (the entire vocabulary)

x	*y*
agriculture	<s>
agriculture	is
agriculture	the

corpus (English Wikipedia):

agriculture is the traditional mainstay of the cambodian economy .

but benares has been destroyed by an earthquake .

…

Simplest kind of structured prediction: Sequence Labeling

Part-of-Speech Tagging

determiner	verb (past)	prep.	proper noun	proper noun	poss.	adj.	noun
Some	questioned	if	Tim	Cook	’s	first	product

modal	verb	det.	adjective	noun	prep.	proper noun	punc.
would	be	a	breakaway	hit	for	Apple	.

Formulating segmentation tasks as sequence labeling via B-I-O labeling:

Named Entity Recognition

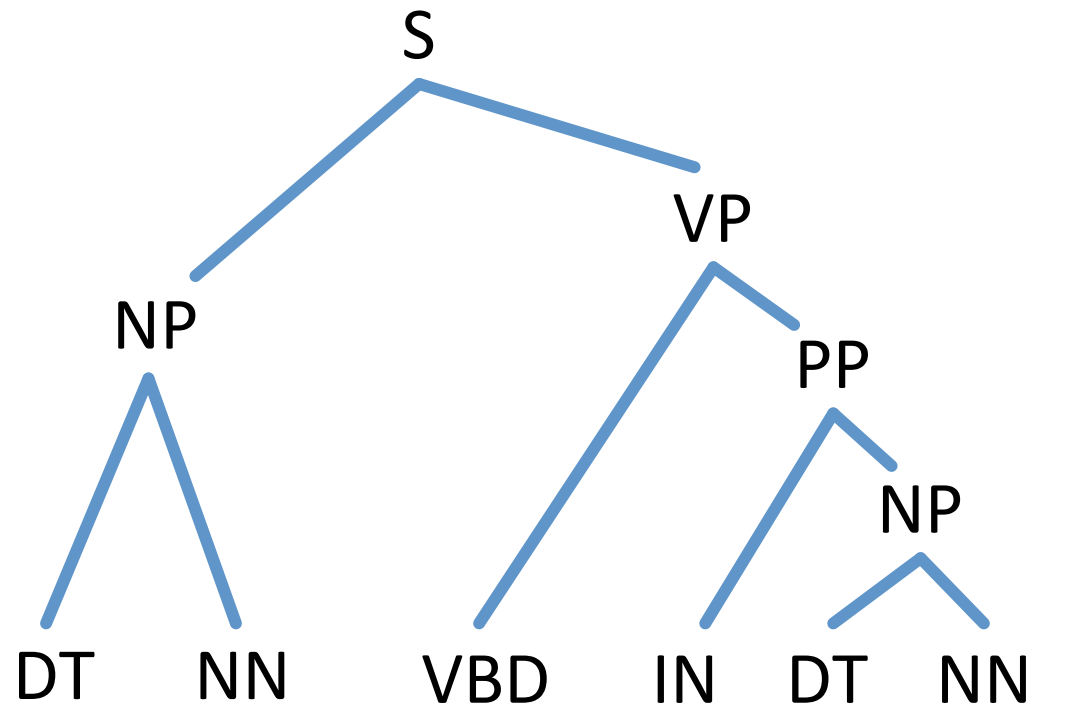
O	O	O	B-PERSON	I-PERSON	O	O	O
Some	questioned	if	Tim	Cook	’s	first	product

O	O	O	O	O	O	B-ORGANIZATION	O
would	be	a	breakaway	hit	for	Apple	.

B = “begin”
I = “inside”
O = “outside”

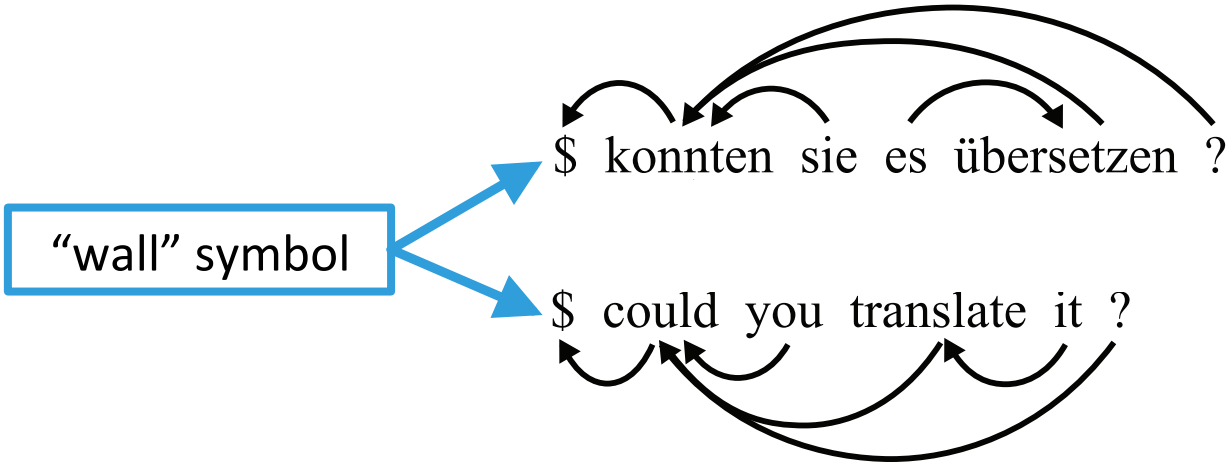
Constituent Parsing

(S (NP the man) (VP walked (PP to (NP the park))))



Key:
S = sentence
NP = noun phrase
VP = verb phrase
PP = prepositional phrase
DT = determiner
NN = noun
VBD = verb (past tense)
IN = preposition

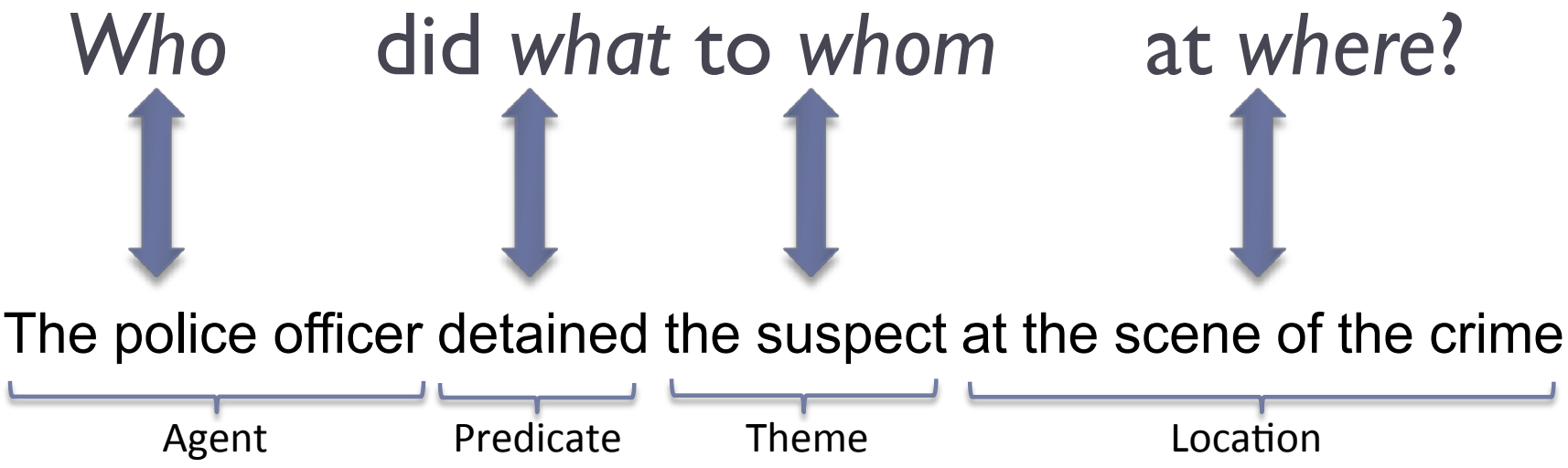
Dependency Parsing



Several Kinds of Semantic Parsing

- semantic role labeling (SRL)
- frame-semantic parsing
- "semantic parsing" (first-order logic)
- abstract meaning representation (AMR)
- dependency-based compositional semantics

Semantic Role Labeling



Semantic role labeling (SRL)

- The task of finding the semantic roles of each argument of each predicate in a sentence.
- FrameNet versus PropBank:

[You]	can't	[blame]	[the program]	[for being unable to identify it]
COGNIZER		TARGET	EVALUEE	REASON

[The San Francisco Examiner]	issued	[a special edition]	[yesterday]	
ARG0	TARGET	ARG1	ARGM-TMP	

Machine Translation

Applications of our Classifier Framework

task	input (x)	output (y)	output space ($\mathcal{L}$)	size of $\mathcal{L}$
text classification	a sentence	gold standard label for x	pre-defined, small label set (e.g., {positive, negative})	2-10
word sense disambiguation	instance of a particular word (e.g., *bass*) with its context	gold standard word sense of x	pre-defined sense inventory from WordNet for *bass*	2-30
learning skip-gram word embeddings	instance of a word in a corpus	a word in the context of x in a corpus	vocabulary	$\|V\|$
part-of-speech tagging	a sentence	gold standard part-of-speech tags for x	all possible part-of-speech tag sequences with same length as x	$\|P\|^{\|x\|}$

Applications of Classifier Framework (continued)

task	input (x)	output (y)	output space ($\mathcal{L}$)	size of $\mathcal{L}$
named entity recognition	a sentence	gold standard named entity labels for x (BIO tags)	all possible BIO label sequences with same length as x	$\|P\|^{\|x\|}$
constituent parsing	a sentence	gold standard constituent parse (labeled bracketing) of x	all possible labeled bracketings of x	exponential in length of x (Catalan number)
dependency parsing	a sentence	gold standard dependency parse (labeled directed spanning tree) of x	all possible labeled directed spanning trees of x	exponential in length of x
machine translation	a sentence	a translation of x	all possible translations of x	potentially infinite

Modeling

model families

- linear models
  - lots of freedom in defining features, though feature engineering required for best performance
  - learning uses optimization of a loss function
  - one can (try to) interpret learned feature weights
- stochastic/generative models
  - linear models with simple “features” (counts of events)
  - learning is easy: count & normalize (but smoothing needed)
  - easy to generate samples
- neural networks
  - can usually get away with less feature engineering
  - learning uses optimization of a loss function
  - hard to interpret (though we try!), but often works best

special case of linear models: stochastic/generative models

model	tasks	context expansion
n-gram language models	language modeling (for MT, ASR, etc.)	increase *n*
hidden Markov models	part-of-speech tagging, named entity recognition, word clustering	increase order of HMM (e.g., bigram HMM → trigram HMM)
probabilistic context-free grammars	constituent parsing	increase size of rules, e.g., flattening, parent annotation, etc.

- all use MLE + smoothing (though probably different kinds of smoothing)
- all assign probability to sentences (some assign probability jointly to pairs of <sentence, something else>)
- all have the same trade-off of increasing "context" (feature size) and needing more data / better smoothing

Feature Engineering for Text Classification

$$\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) = \sum_i \theta_i f_i(\boldsymbol{x}, y)$$

- Two features:

$$f_1(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$
$$f_2(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

where $\mathbb{I}[S] = 1$ if S is true, 0 otherwise

Higher-Order Binary Feature Templates

unigram binary template:

$$f^{\mathrm{u,b}}(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ label}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{word}]$$

bigram binary template:

$$f^{\mathrm{b,b}}(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ label}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{“word1 word2”}]$$

trigram binary features

…

2-transformation (1-layer) network

$$\boldsymbol{z}^{(1)} = g\left(W^{(0)}\boldsymbol{x} + \boldsymbol{b}^{(0)}\right)$$

$$\boldsymbol{s} = g\left(W^{(1)}\boldsymbol{z}^{(1)} + \boldsymbol{b}^{(1)}\right)$$

vector of label scores

- we'll call this a "2-transformation" neural network, or a "1-layer" neural network
- input vector is $\boldsymbol{x}$
- score vector is $\boldsymbol{s}$
- one hidden vector $\boldsymbol{z}^{(1)}$ ("hidden layer")

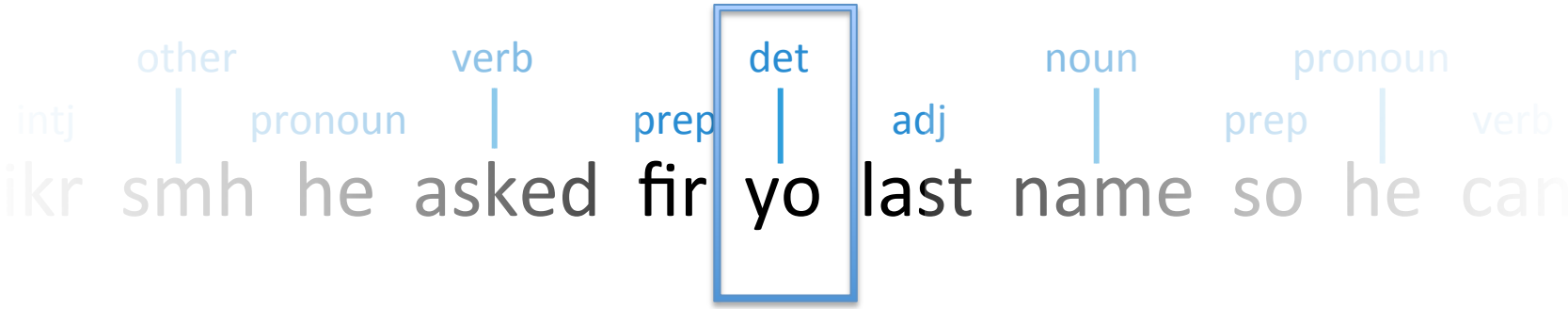
1-layer neural network for sentiment classification

$$\boldsymbol{z}^{(1)} = g\left(W^{(0)}\boldsymbol{x} + \boldsymbol{b}^{(0)}\right)$$

$$\boldsymbol{s} = g\left(W^{(1)}\boldsymbol{z}^{(1)} + \boldsymbol{b}^{(1)}\right)$$

$$\boldsymbol{s} = \begin{bmatrix} \text{score}(\boldsymbol{x}, \text{positive}, \boldsymbol{\theta}) \\ \text{score}(\boldsymbol{x}, \text{negative}, \boldsymbol{\theta}) \end{bmatrix}$$

Neural Networks for Twitter Part-of-Speech Tagging



- let's use the center word + two words to the right:

$$\boldsymbol{x} = [\underbrace{0.4 \ \ldots \ 0.9}_{\text{vector for } yo} \ \underbrace{0.2 \ \ldots \ 0.7}_{\text{vector for } last} \ \underbrace{0.3 \ \ldots \ 0.6}_{\text{vector for } name}]^\top$$

- if *name* is to the right of *yo*, then *yo* is probably a form of *your*
- but our ***x*** above uses separate dimensions for each position!
  - i.e., *name* is two words to the right
  - what if *name* is one word to the right?

Convolution

$\boldsymbol{c}$ = "feature map", has an entry for each word position in context window / sentence

$$\boldsymbol{x} = [\underbrace{0.4 \ \ldots \ 0.9}_{\text{vector for } yo} \ \underbrace{0.2 \ \ldots \ 0.7}_{\text{vector for } last} \ \underbrace{0.3 \ \ldots \ 0.6}_{\text{vector for } name}]^\top$$

$$c_1 = \boldsymbol{w} \cdot \boldsymbol{x}_{1:d}$$

$$c_2 = \boldsymbol{w} \cdot \boldsymbol{x}_{d+1:2d}$$

$$c_3 = \boldsymbol{w} \cdot \boldsymbol{x}_{2d+1:3d}$$

Pooling

$\boldsymbol{c}$ = "feature map", has an entry for each word position in context window / sentence

how do we convert this into a fixed-length vector?
use **pooling**:

- max-pooling: returns maximum value in $\boldsymbol{c}$
- average pooling: returns average of values in $\boldsymbol{c}$

vector for *yo* vector for *last* vector for *name*

$$c_1 = \boldsymbol{w} \cdot \boldsymbol{x}_{1:d}$$

$$c_2 = \boldsymbol{w} \cdot \boldsymbol{x}_{d+1:2d}$$

$$c_3 = \boldsymbol{w} \cdot \boldsymbol{x}_{2d+1:3d}$$

Pooling

$\boldsymbol{c}$ = "feature map", has an entry for each word position in context window / sentence

how do we convert this into a fixed-length vector?
use **pooling**:

max-pooling: returns maximum value in $\boldsymbol{c}$
average pooling: returns average of values in $\boldsymbol{c}$

vector for *yo* vector for *last* vector for *name*

$$c_1 = \boldsymbol{w} \cdot \boldsymbol{x}_{1:d}$$

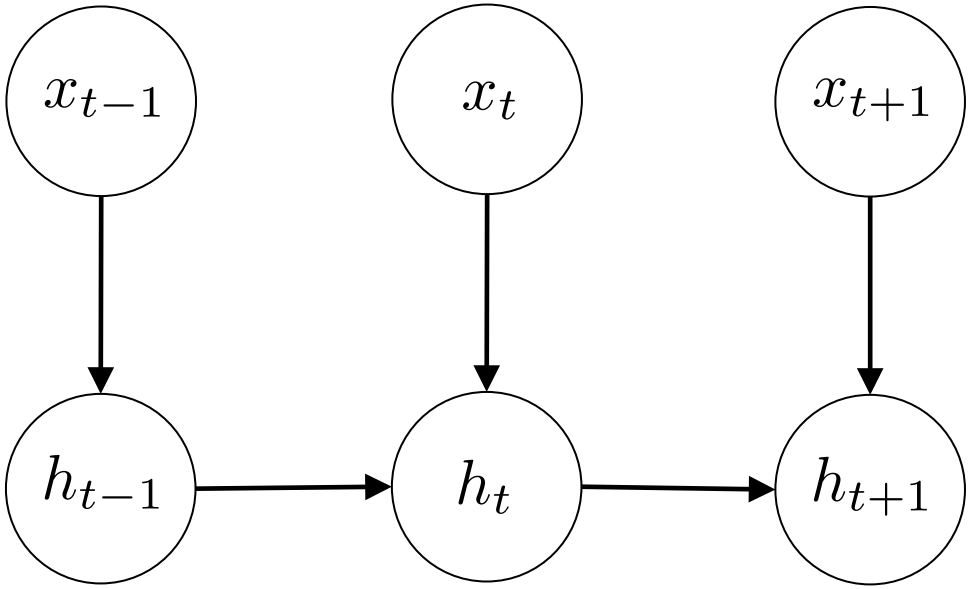
then, this single filter $\boldsymbol{w}$ produces a single feature value (the output of some kind of pooling).
in practice, we use many filters of many different lengths (e.g., *n*-grams rather than words).

Convolutional Neural Networks

- convolutional neural networks (**convnets** or **CNNs**) use filters that are "convolved with" (matched against all positions of) the input
- think of convolution as "perform the same operation everywhere on the input in some systematic order"
- "convolutional layer" = set of filters that are convolved with the input vector (whether ***x*** or hidden vector)
- could be followed by more convolutional layers, or by a type of pooling
- often used in NLP to convert a sentence into a feature vector

Recurrent Neural Networks

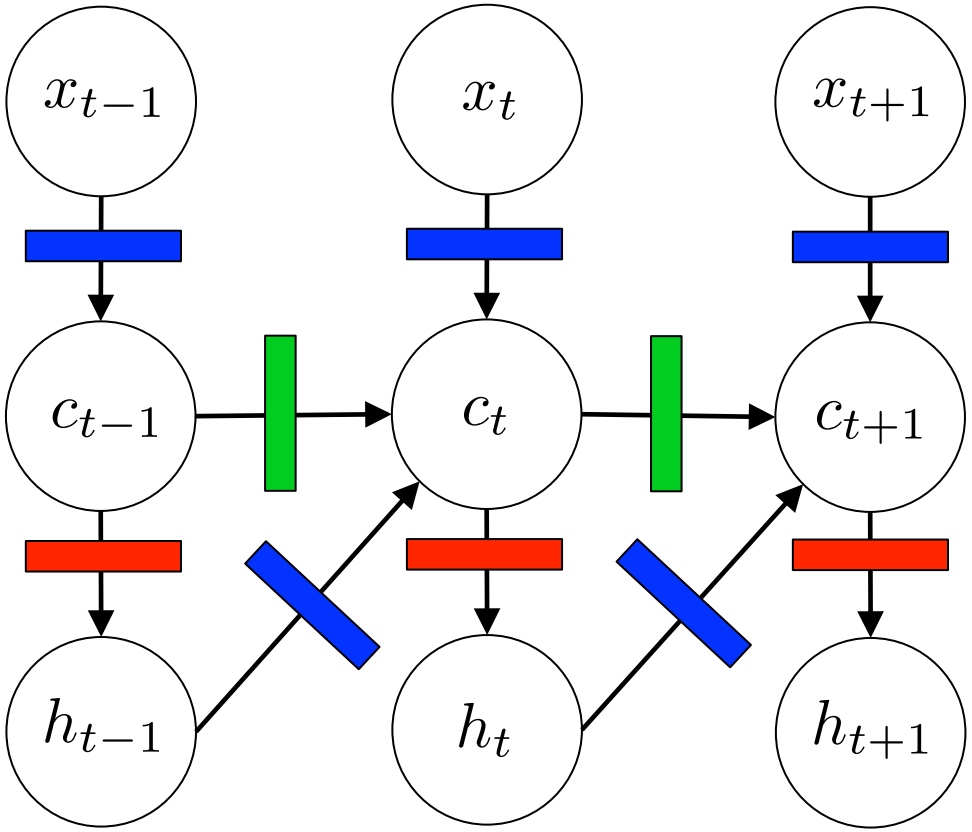
$$h_t = \tanh\left(W^{(xh)}x_t + W^{(hh)}h_{t-1} + b^{(h)}\right)$$



"hidden vector"

Long Short-Term Memory (LSTM) Recurrent Neural Networks

$$c_t = f_t c_{t-1} + i_t \tanh\left(W^{(xc)} x_t + W^{(hc)} h_{t-1} + b^{(c)}\right)$$

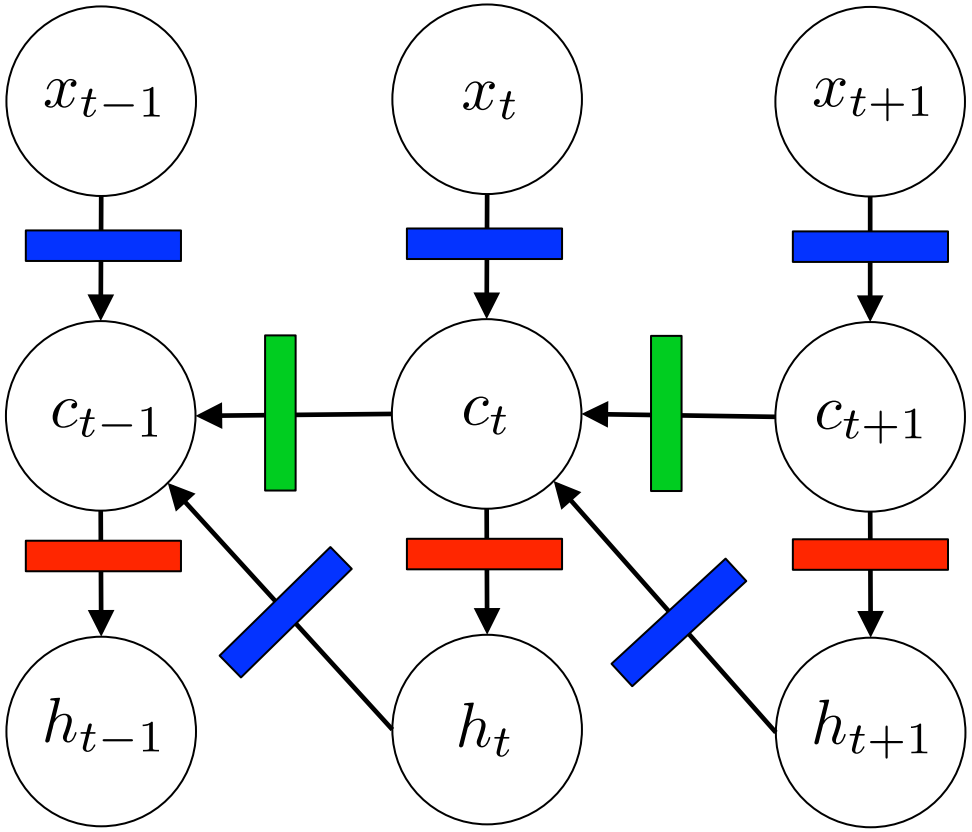


$$h_t = o_t \tanh(c_t)$$

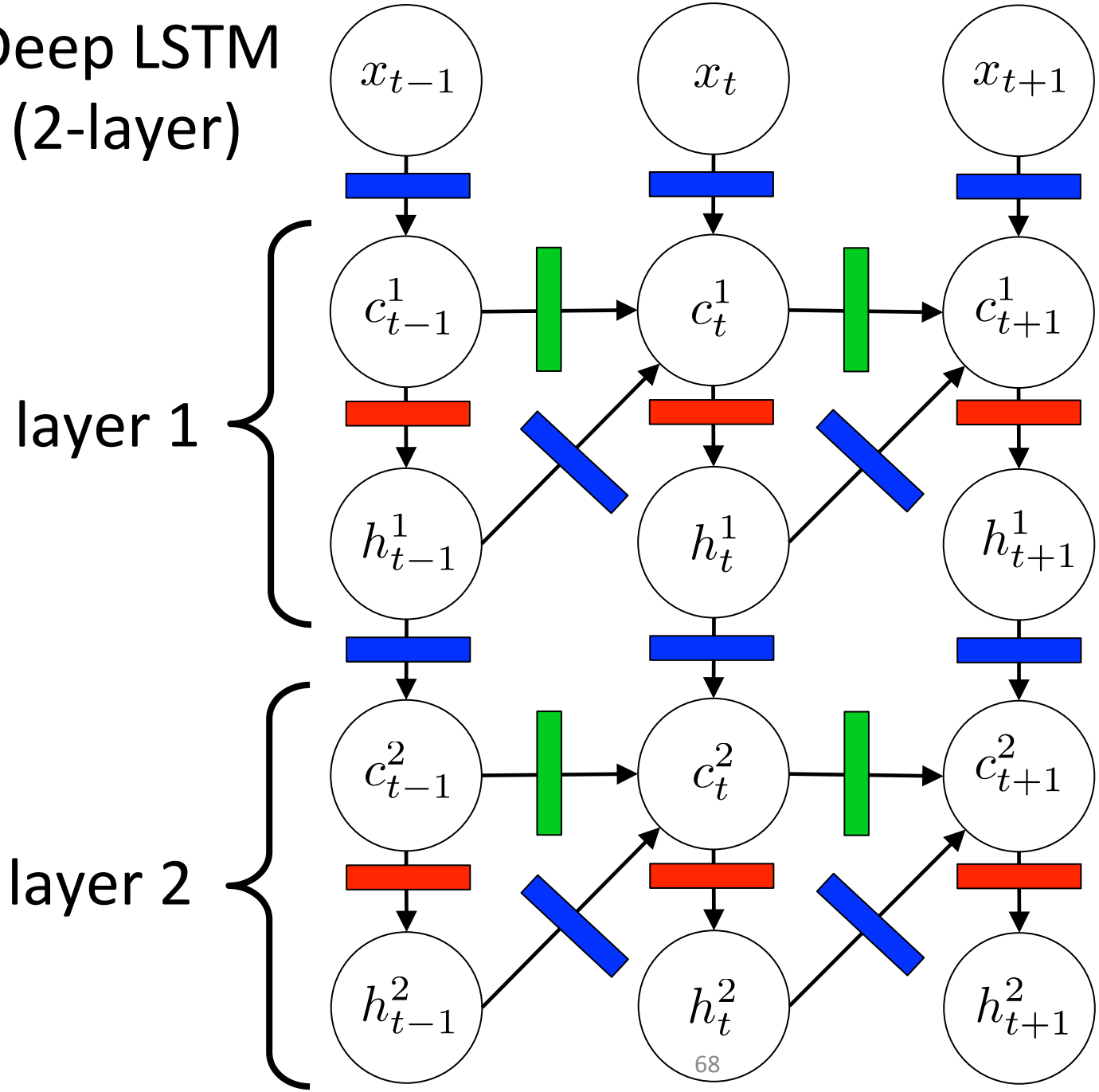
Backward & Bidirectional LSTMs

bidirectional:

if shallow, just use forward and backward LSTMs in parallel, concatenate final two hidden vectors, feed to softmax

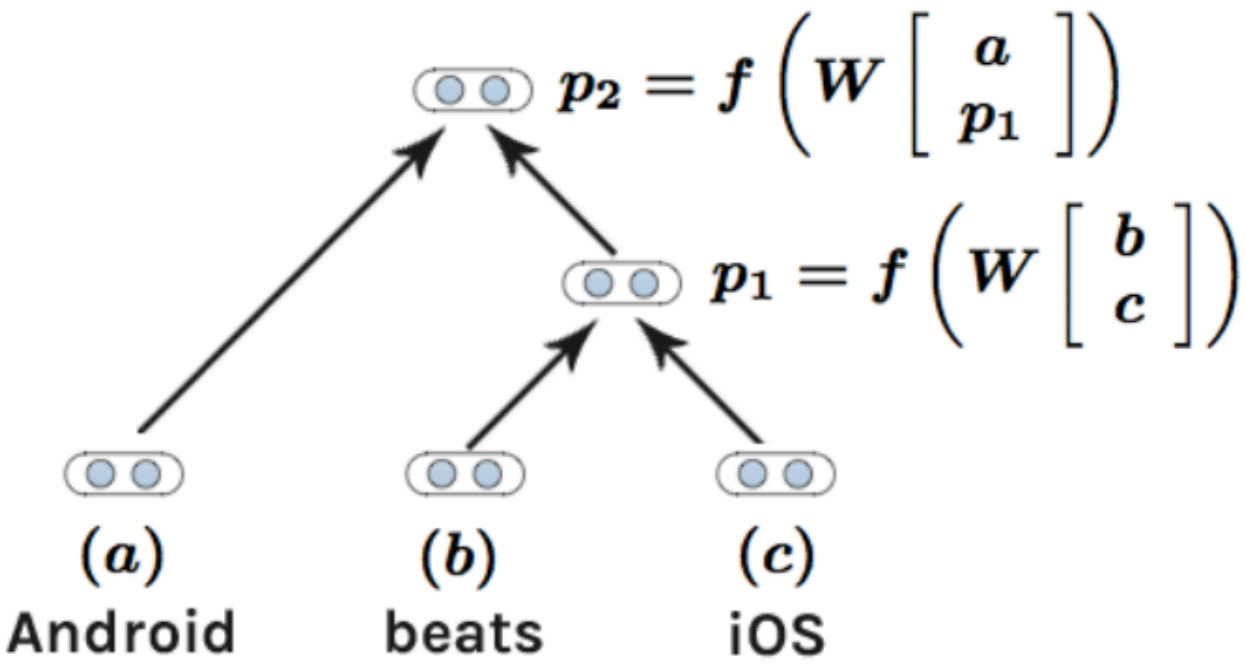


Deep LSTM (2-layer)



Recursive Neural Networks for NLP

- first, run a constituent parser on the sentence
- convert the constituent tree to a binary tree (each rewrite has exactly two children)
- construct vector for sentence recursively at each rewrite ("split point"):



Learning

Cost Functions

- cost function: scores output against a gold standard

$$\text{cost} : \mathcal{L} \times \mathcal{L} \rightarrow \mathbb{R}_{\geq 0}$$

- should reflect the evaluation metric for your task

- usual conventions: $\text{cost}(y, y) = 0$ $\text{cost}(y, y') = \text{cost}(y', y)$

- for classification, what cost should we use?

$$\text{cost}(y, y') = \mathbb{I}[y \neq y']$$

Empirical Risk Minimization
(Vapnik et al.)

- replace expectation with sum over examples:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; \mathbb{E}_{P(\boldsymbol{x},y)} \left[\operatorname{cost}(y, \operatorname{classify}(\boldsymbol{x}, \boldsymbol{\theta})) \right]$$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{T}|} \operatorname{cost}(y^{(i)}, \operatorname{classify}(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}))$$

Empirical Risk Minimization
(Vapnik et al.)

- replace expectation with sum over examples:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ \mathbb{E}_{P(\boldsymbol{x},y)}\left[\operatorname{cost}(y, \operatorname{classify}(\boldsymbol{x}, \boldsymbol{\theta}))\right]$$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{T}|} \operatorname{cost}(y^{(i)}, \operatorname{classify}(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}))$$

problem: NP-hard even for binary classification with linear models

Empirical Risk Minimization with Surrogate Loss Functions

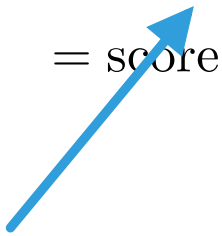
- given training data: $\mathcal{T} = \{\langle \boldsymbol{x}^{(i)}, y^{(i)} \rangle\}_{i=1}^{|\mathcal{T}|}$
  where each $y^{(i)} \in \mathcal{L}$ is a label
- we want to solve the following:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{T}|} \text{loss}(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

many possible loss functions to consider optimizing

Loss Functions

name	loss	where used
cost ("0-1")	$\text{cost}(y, \text{classify}(\boldsymbol{x}, \boldsymbol{\theta}))$	intractable, but underlies "direct error minimization"
perceptron	$-\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} \text{score}(\boldsymbol{x}, y', \boldsymbol{\theta})$	perceptron algorithm (Rosenblatt, 1958)
hinge	$-\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} (\text{score}(\boldsymbol{x}, y', \boldsymbol{\theta}) + \text{cost}(y, y'))$	support vector machines, other large-margin algorithms
log	$-\log p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x})$ $= \text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \log \sum_{y' \in \mathcal{L}} \exp\{\text{score}(\boldsymbol{x}, y', \boldsymbol{\theta})\}$	logistic regression, conditional random fields, maximum entropy models



$$p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x}) = \frac{\exp\{\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta})\}}{\sum_{y' \in \mathcal{L}} \exp\{\text{score}(\boldsymbol{x}, y', \boldsymbol{\theta})\}}$$

(Sub)gradients of Losses for Linear Models

name	entry *j* of (sub)gradient of loss for linear model
cost ("0-1")	not subdifferentiable in general
perceptron	$-f_j(\boldsymbol{x}, y) + f_j(\boldsymbol{x}, \hat{y}),\ \text{where } \hat{y} = \text{classify}(\boldsymbol{x}, \boldsymbol{\theta})$
hinge	$-f_j(\boldsymbol{x}, y) + f_j(\boldsymbol{x}, \tilde{y}),\ \text{where } \tilde{y} = \text{costClassify}(\boldsymbol{x}, y, \boldsymbol{\theta})$
log	

$$\text{classify}(\boldsymbol{x}, \boldsymbol{\theta}) = \underset{y' \in \mathcal{L}}{\text{argmax}}\ \text{score}(\boldsymbol{x}, y', \boldsymbol{\theta})$$

$$\text{costClassify}(\boldsymbol{x}, y, \boldsymbol{\theta}) = \underset{y' \in \mathcal{L}}{\text{argmax}}\ \text{score}(\boldsymbol{x}, y', \boldsymbol{\theta}) + \text{cost}(y, y')$$

(Sub)gradients of Losses for Linear Models

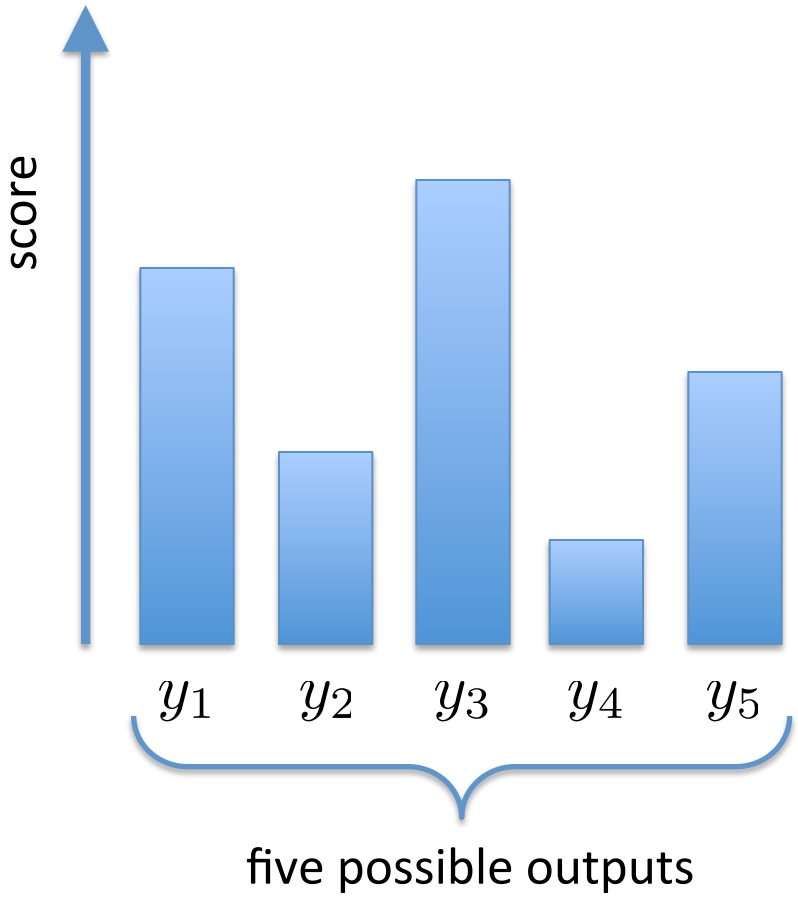
name	entry *j* of (sub)gradient of loss for linear model
cost ("0-1")	not subdifferentiable in general
perceptron	$-f_j(\boldsymbol{x}, y) + f_j(\boldsymbol{x}, \hat{y}),\ \text{where } \hat{y} = \text{classify}(\boldsymbol{x}, \boldsymbol{\theta})$
hinge	$-f_j(\boldsymbol{x}, y) + f_j(\boldsymbol{x}, \tilde{y}),\ \text{where } \tilde{y} = \text{costClassify}(\boldsymbol{x}, y, \boldsymbol{\theta})$
log	$-f_j(\boldsymbol{x}, y) + \mathbb{E}_{p_{\boldsymbol{\theta}}(\cdot\|\boldsymbol{x})}[f_j(\boldsymbol{x}, \cdot)]$

expectation of feature value with respect to distribution over *y* (where distribution is defined by theta)

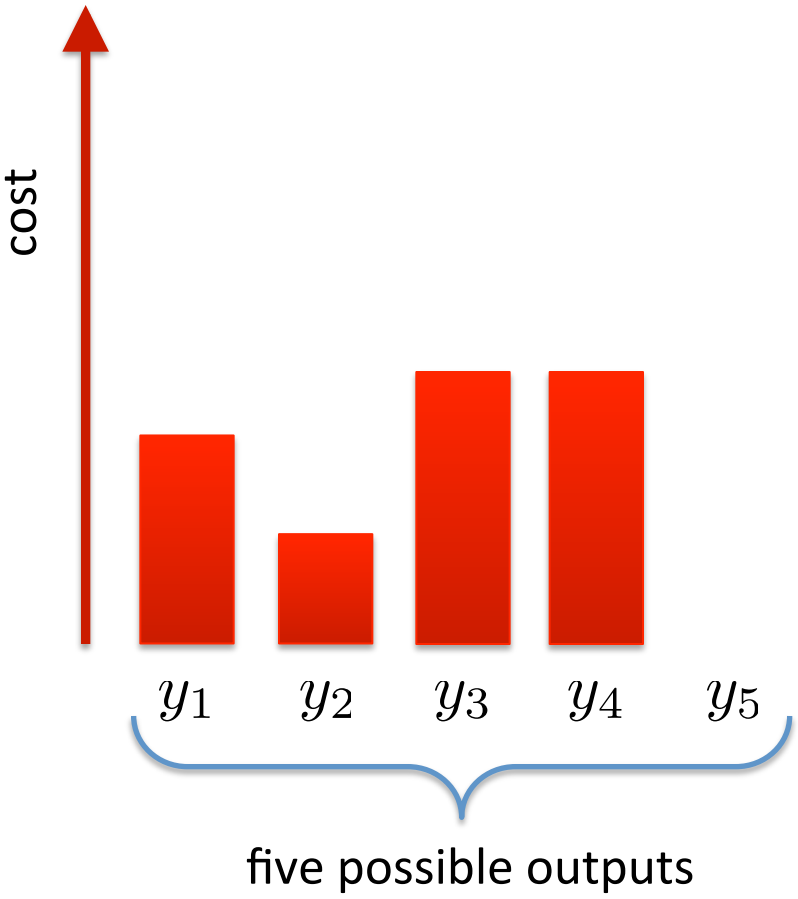
alternative notation:

$$-f_j(\boldsymbol{x}, y) + \mathbb{E}_{y' \sim p_{\boldsymbol{\theta}}(Y|\boldsymbol{x})}[f_j(\boldsymbol{x}, y')]$$

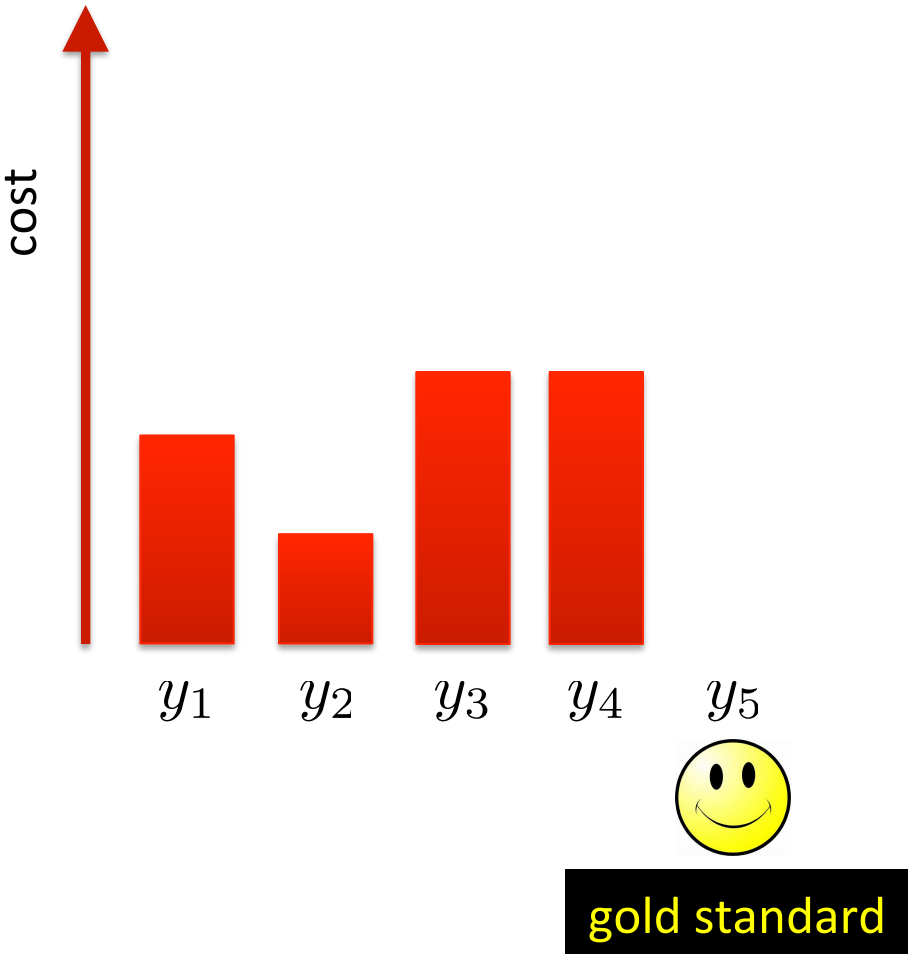
Visualization



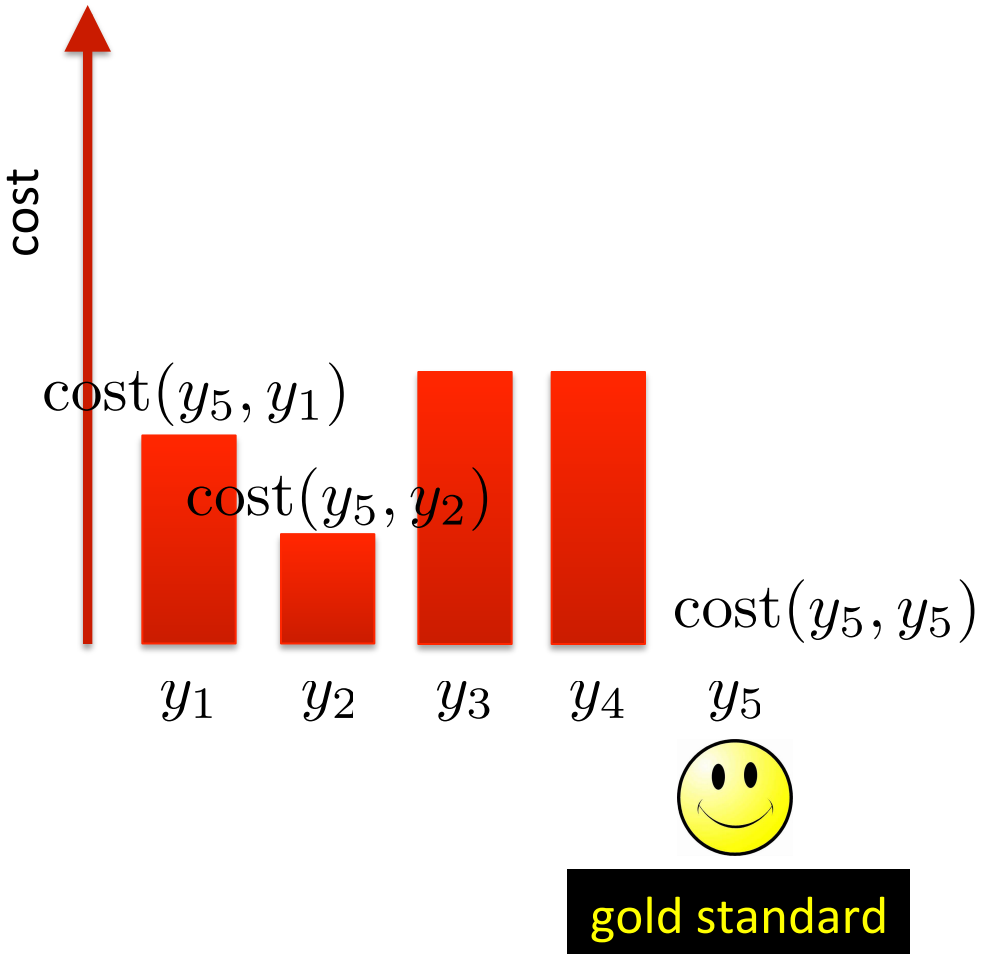
Visualization



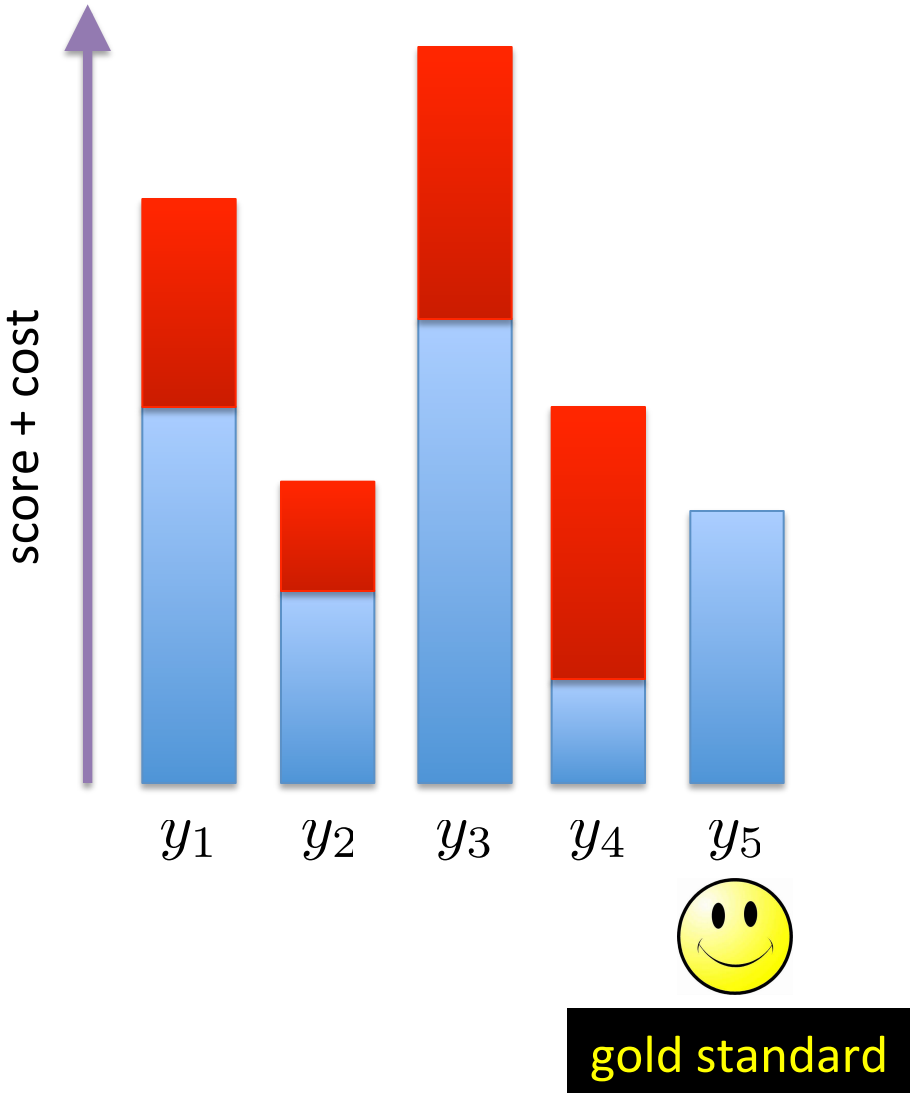
Visualization



Visualization



Visualization

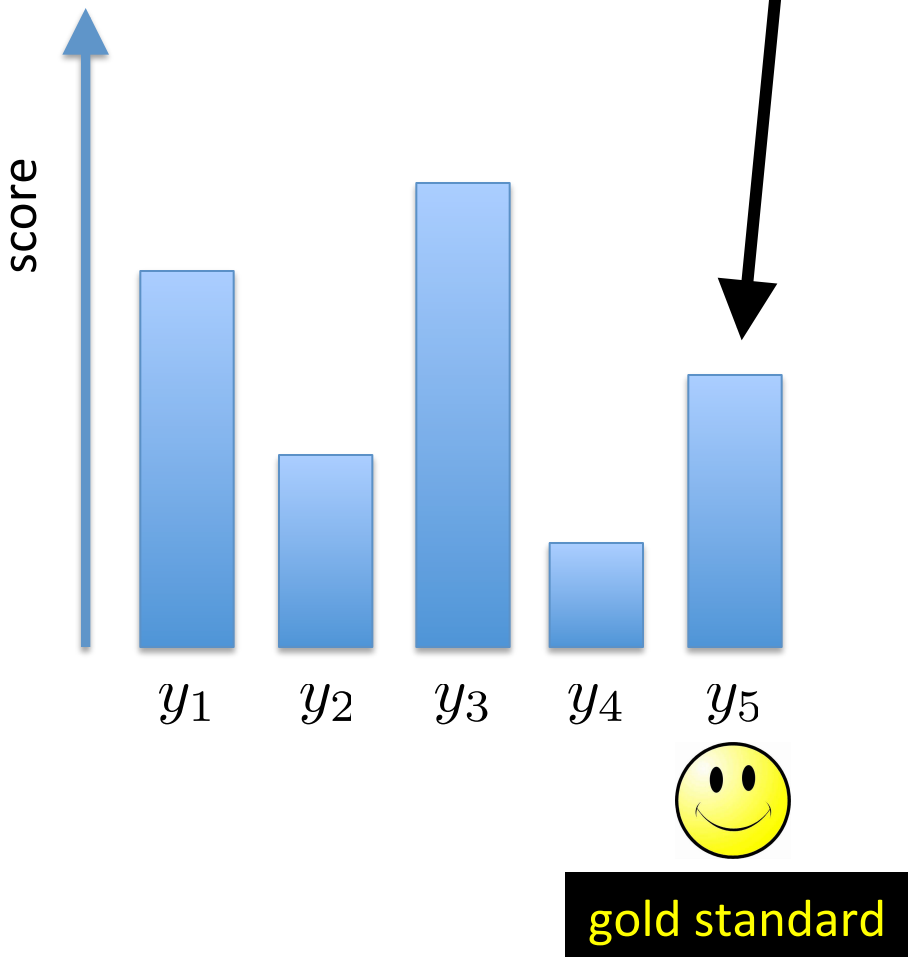


perceptron loss:

$$\text{loss}_{\text{perc}}(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} \text{score}(\boldsymbol{x}, y', \boldsymbol{\theta})$$

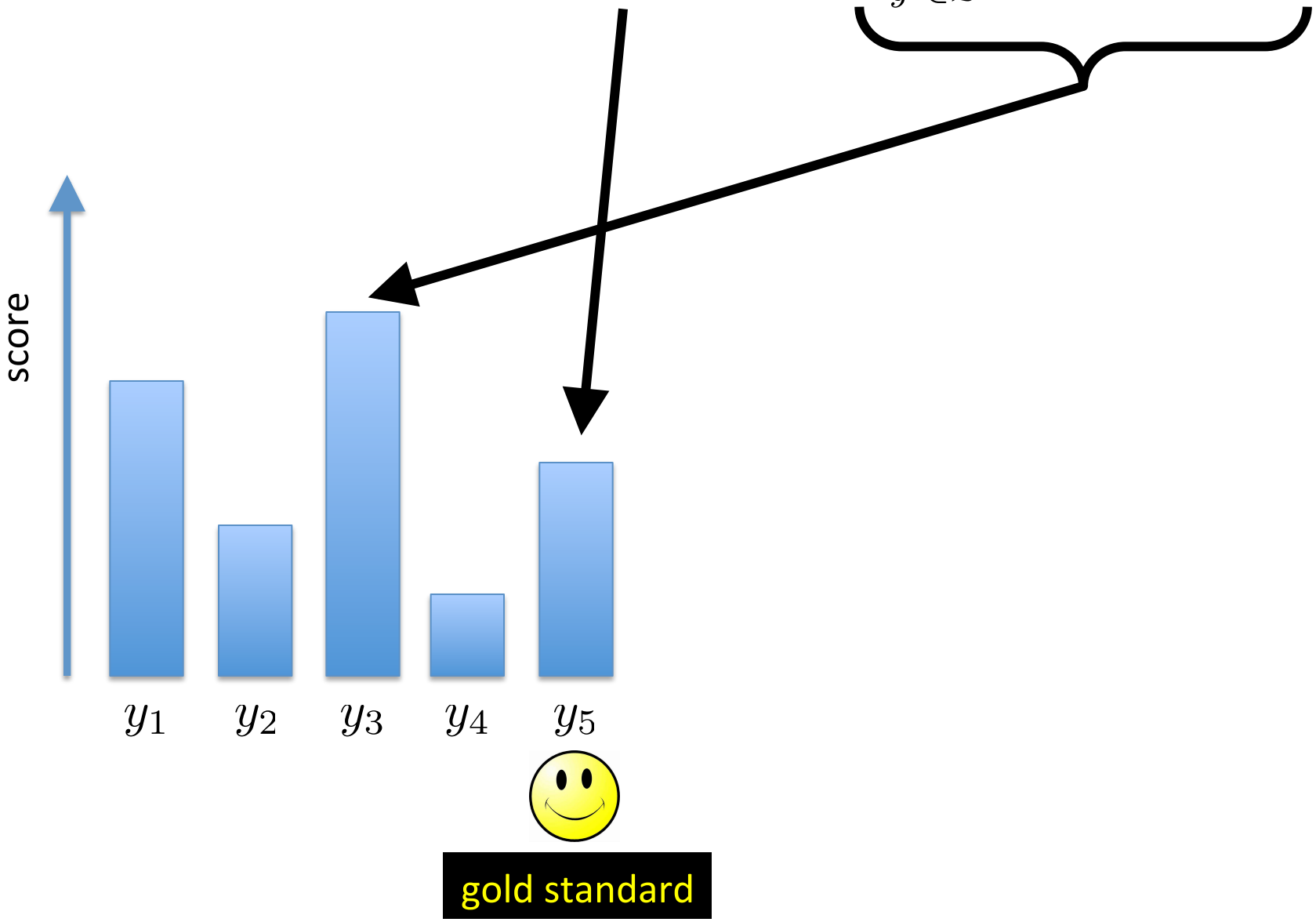
perceptron loss:

$$\text{loss}_{\text{perc}}(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} \text{score}(\boldsymbol{x}, y', \boldsymbol{\theta})$$



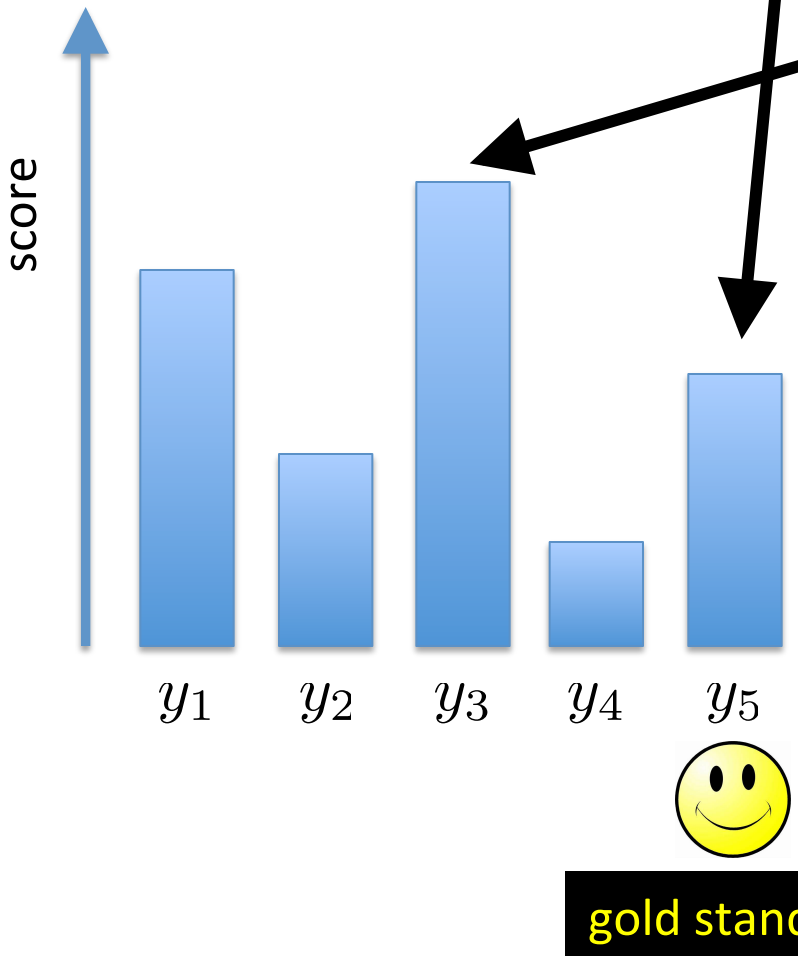
perceptron loss:

$$\text{loss}_{\text{perc}}(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} \text{score}(\boldsymbol{x}, y', \boldsymbol{\theta})$$



perceptron loss:

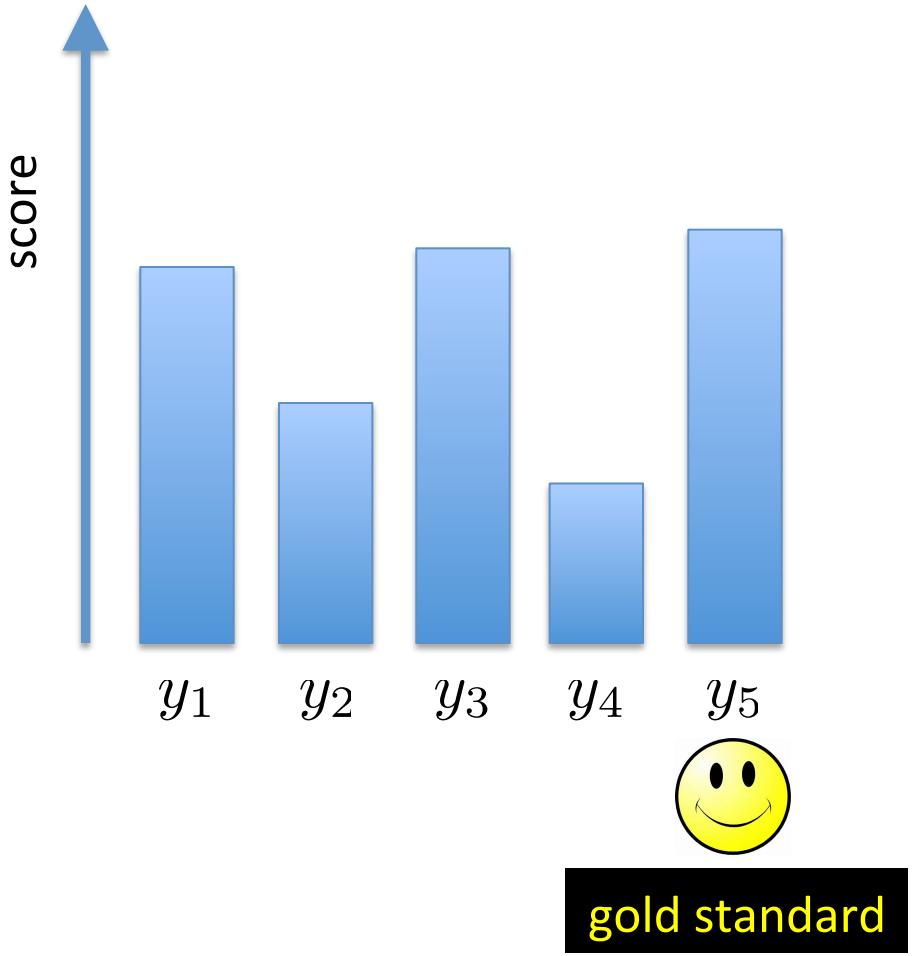
$$\text{loss}_{\text{perc}}(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} \text{score}(\boldsymbol{x}, y', \boldsymbol{\theta})$$



effect of learning?

perceptron loss:

$$\text{loss}_{\text{perc}}(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} \text{score}(\boldsymbol{x}, y', \boldsymbol{\theta})$$



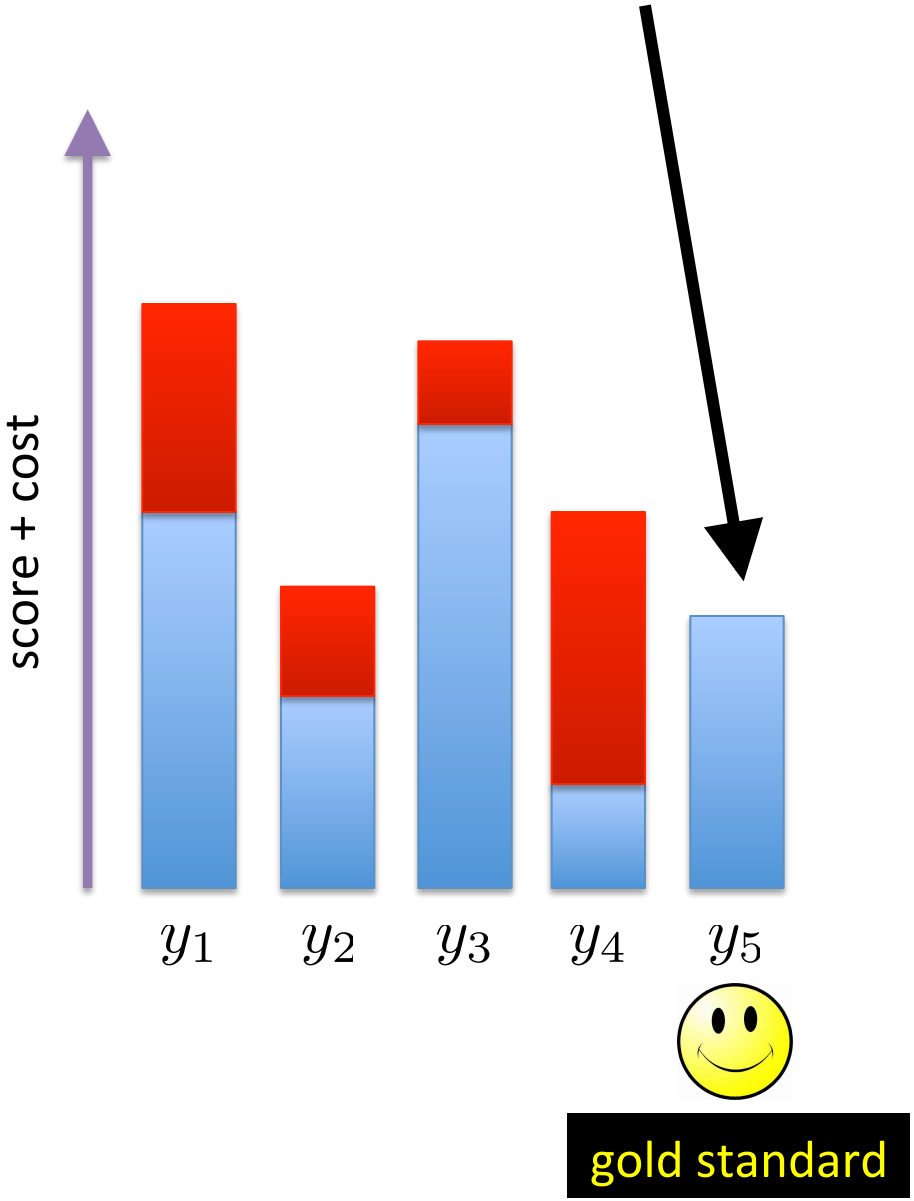
effect of learning: gold standard will have highest score

hinge loss:

$$\text{loss}_{\text{hinge}}(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} \left(\text{score}(\boldsymbol{x}, y', \boldsymbol{\theta}) + \text{cost}(y, y')\right)$$

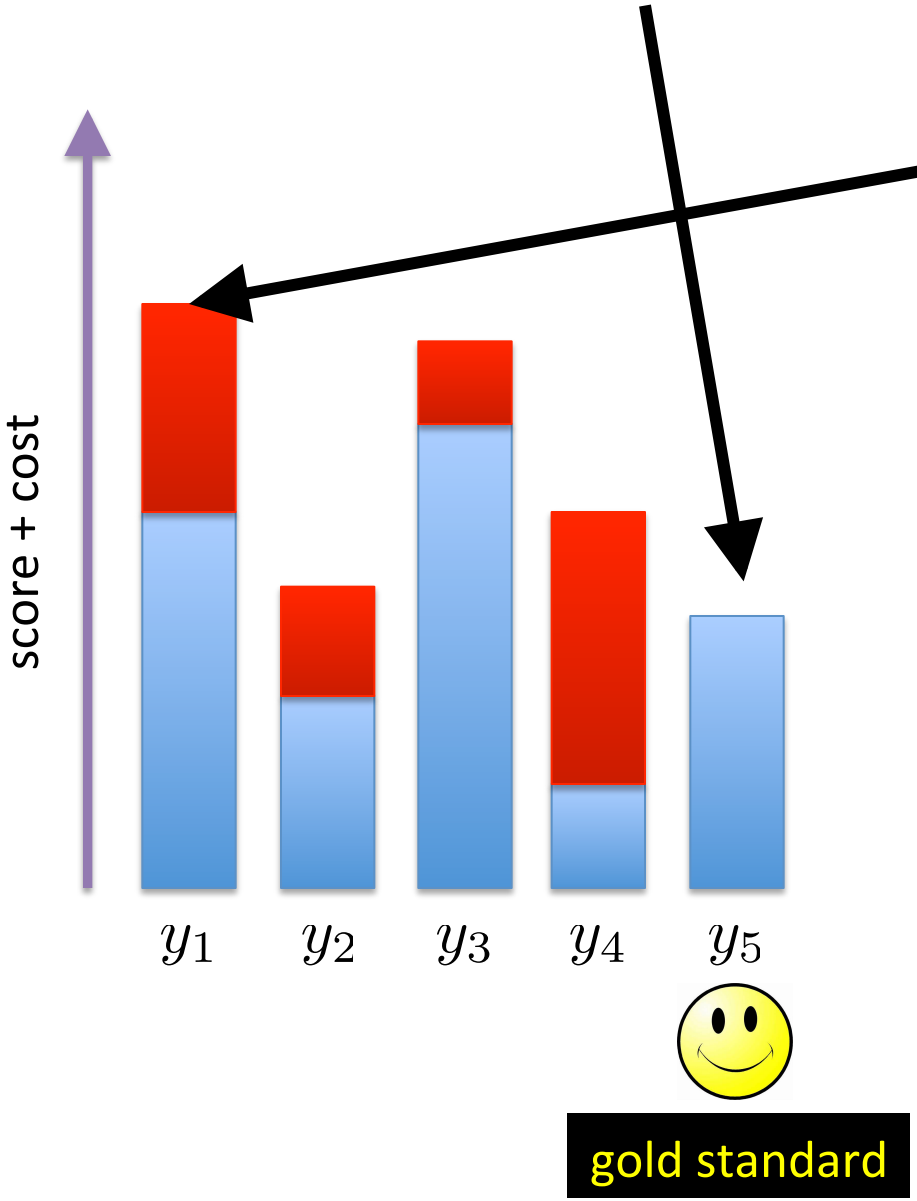
hinge loss:

$$\text{loss}_{\text{hinge}}(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \underbrace{\max_{y' \in \mathcal{L}} (\text{score}(\boldsymbol{x}, y', \boldsymbol{\theta}) + \text{cost}(y, y'))}$$



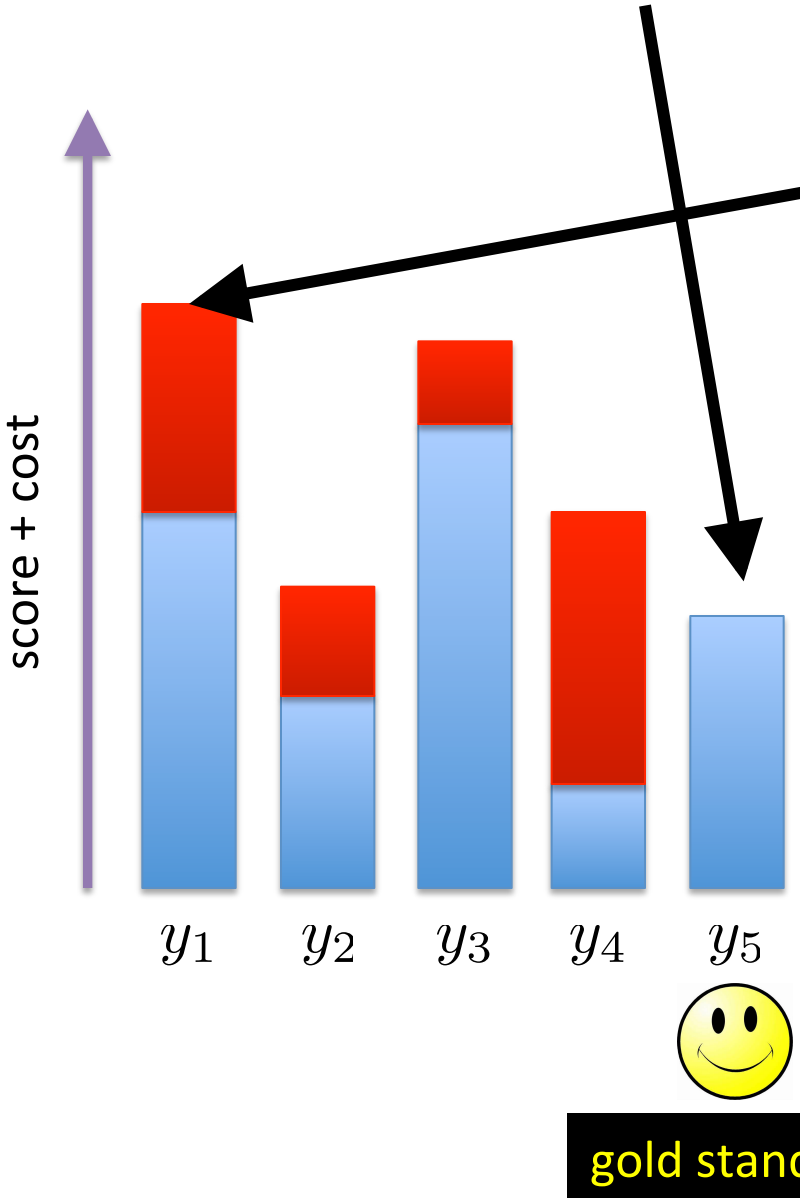
hinge loss:

$$\text{loss}_{\text{hinge}}(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} \left(\text{score}(\boldsymbol{x}, y', \boldsymbol{\theta}) + \text{cost}(y, y')\right)$$



hinge loss:

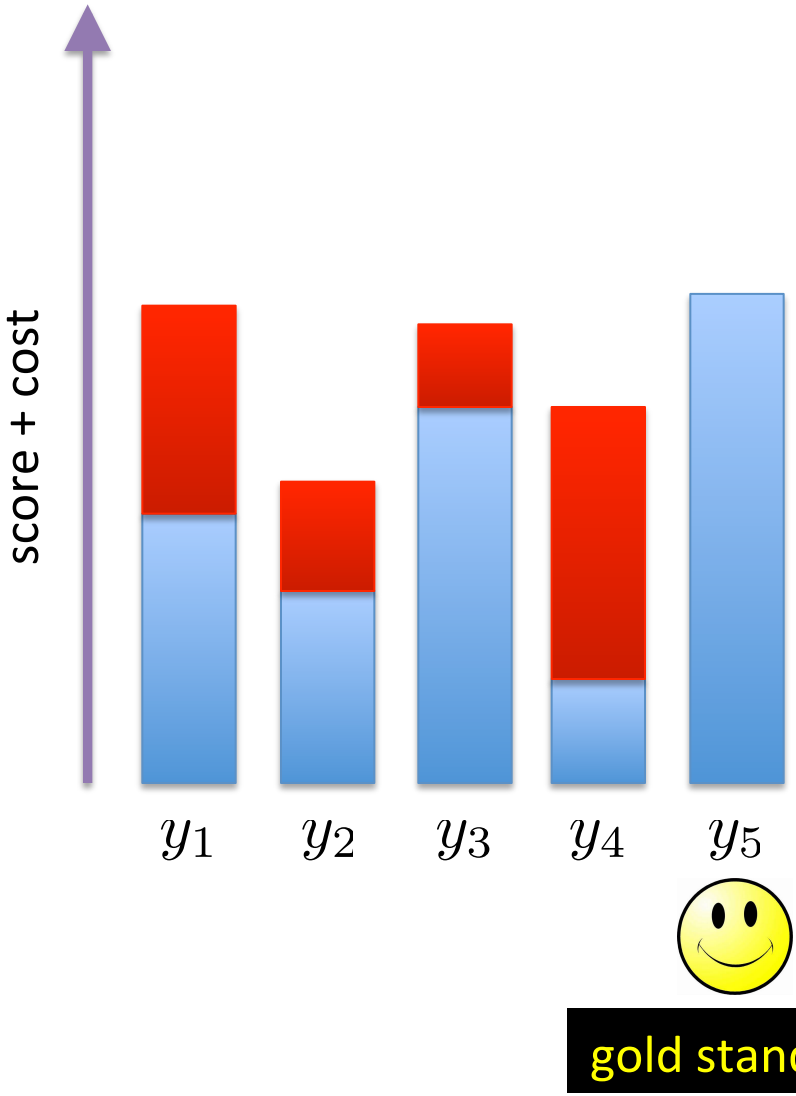
$$\text{loss}_{\text{hinge}}(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} \left(\text{score}(\boldsymbol{x}, y', \boldsymbol{\theta}) + \text{cost}(y, y')\right)$$



effect of learning?

hinge loss:

$$\text{loss}_{\text{hinge}}(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} \left(\text{score}(\boldsymbol{x}, y', \boldsymbol{\theta}) + \text{cost}(y, y')\right)$$



effect of learning: score of gold standard will be higher than score+cost of all others

Regularized Empirical Risk Minimization

- given training data: $\mathcal{T} = \{\langle \boldsymbol{x}^{(i)}, y^{(i)} \rangle\}_{i=1}^{|\mathcal{T}|}$

  where each $y^{(i)} \in \mathcal{L}$ is a label

- we want to solve the following:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{T}|} \operatorname{loss}(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta})$$

regularization strength

regularization term

Regularization Terms

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{T}|} \operatorname{loss}(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta})$$

- most common: penalize large parameter values
- intuition: large parameters might be instances of overfitting
- examples:

L_2 regularization: $R_{\mathrm{L2}}(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_2^2 = \sum_i \theta_i^2$

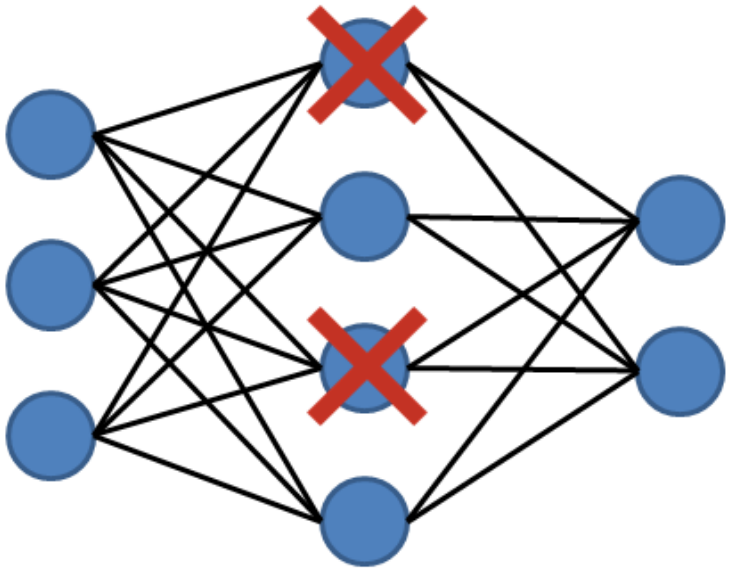
(also called Tikhonov regularization or ridge regression)

L_1 regularization: $R_{\mathrm{L1}}(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_1 = \sum_i |\theta_i|$

(also called basis pursuit or LASSO)

Dropout

- popular regularization method for neural networks
- randomly “drop out” (set to zero) some of the vector entries in the layers



Inference

Exponentially-Large Search Problems

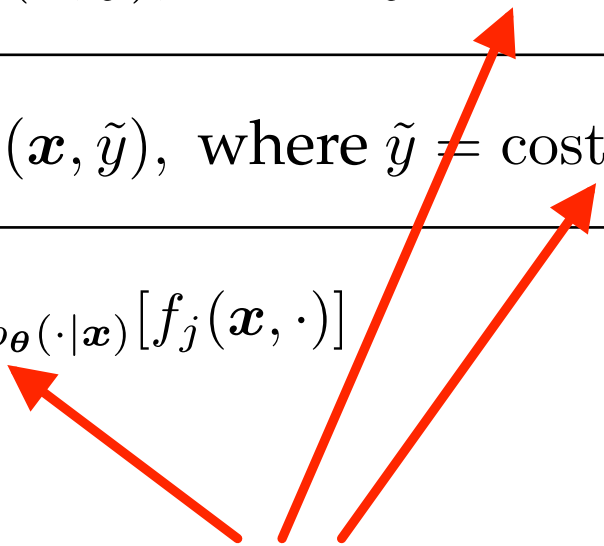
inference: solve argmax

$$\text{classify}(x, \boldsymbol{\theta}) = \operatorname*{argmax}_{y} \ \text{score}(x, y, \boldsymbol{\theta})$$

- when output is a sequence or tree, this argmax requires iterating over an exponentially-large set

Learning requires solving exponentially-hard problems too!

loss	entry *j* of (sub)gradient of loss for linear model
perceptron	$-f_j(\boldsymbol{x}, y) + f_j(\boldsymbol{x}, \hat{y}),\ \text{where } \hat{y} = \text{classify}(\boldsymbol{x}, \boldsymbol{\theta})$
hinge	$-f_j(\boldsymbol{x}, y) + f_j(\boldsymbol{x}, \tilde{y}),\ \text{where } \tilde{y} = \text{costClassify}(\boldsymbol{x}, y, \boldsymbol{\theta})$
log	$-f_j(\boldsymbol{x}, y) + \mathbb{E}_{p_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x})}[f_j(\boldsymbol{x}, \cdot)]$



computing each of these terms requires iterating through every possible output

Dynamic Programming (DP)

- what is dynamic programming?
  - a family of algorithms that break problems into smaller pieces and reuse solutions for those pieces
  - only applicable when the problem has certain properties (**optimal substructure** and **overlapping sub-problems**)

- in this class, we use DP to iterate over exponentially-large output spaces in polynomial time
- we focus on a particular type of DP algorithm: **memoization**

Implementing DP algorithms

- even if your goal is to compute a sum or a max, focus first on **counting mode** (count the number of unique outputs for an input)
- memoization = recursion + saving/reusing solutions
  - start by defining recursive equations
  - "memoize" by creating a table to store all intermediate results from recursive equations, use them when requested

Inference in HMMs

$$\text{classify}(\boldsymbol{x}, \boldsymbol{\theta}) = \underset{\boldsymbol{y}}{\text{argmax}}\ p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) = \underset{\boldsymbol{y}}{\text{argmax}} \prod_{i=1}^{|\boldsymbol{x}|} p_{\boldsymbol{\tau}}(y_i \mid y_{i-1}) p_{\boldsymbol{\eta}}(x_i \mid y_i)$$

- since the output is a sequence, this argmax requires iterating over an exponentially-large set
- last week we talked about using dynamic programming (DP) to solve these problems
- for HMMs (and other sequence models), the for solving this is called the **Viterbi algorithm**

Viterbi Algorithm

- recursive equations + memoization:

base case:
returns probability of sequence starting with label *y* for first word

$$V(1, y) = p_{\boldsymbol{\eta}}(x_1 \mid y)\, p_{\boldsymbol{\tau}}(y \mid \texttt{<s>})$$

$$V(m, y) = \max_{y' \in \mathcal{L}} \left(p_{\boldsymbol{\eta}}(x_m \mid y)\, p_{\boldsymbol{\tau}}(y \mid y')\, V(m-1, y') \right)$$

recursive case:
computes probability of max-probability label sequence that ends with label *y* at position *m*

final value is in: $V(|\boldsymbol{x}| + 1, \texttt{</s>})$

Viterbi Algorithm

- space and time complexity?
- can be read off from the recursive equations:

space complexity:
size of memoization table, which is # of unique indices of recursive equations

length of sentence * **number of labels**

$$V(m, y) = \max_{y' \in \mathcal{L}} \left(p_{\boldsymbol{\eta}}(x_m \mid y)\, p_{\boldsymbol{\tau}}(y \mid y')\, V(m-1, y') \right)$$

so, space complexity is O(|*x*| |*L*|)

Viterbi Algorithm

- space and **time** complexity?
- can be read off from the recursive equations:

time complexity:
size of memoization table * complexity of computing each entry

length of sentence * **number of labels** * **each entry requires iterating through the labels**

$$V(m, y) = \max_{y' \in \mathcal{L}} \left(p_{\boldsymbol{\eta}}(x_m \mid y)\, p_{\boldsymbol{\tau}}(y \mid y')\, V(m-1, y') \right)$$

so, time complexity is O(|*x*| |*L*| |*L*|) = O(|*x*| |*L*|^2)

Feature Locality

- **feature locality**: how "big" are your features?
- when designing efficient inference algorithms (whether w/ DP or other methods), we need to be mindful of this
- features can be arbitrarily big in terms of the input, but not in terms of the *output*!
- the features in HMMs are small in both the input and output sequences (only two pieces at a time)

Defining Features

- This is a large part of NLP
- Last 20 years: **feature engineering**
- Last 2 years: **representation learning**

Defining Features

- This is a large part of NLP
- Last 20 years: **feature engineering**
- Last 2 years: **representation learning**

- In this course, we'll do both
- Learning representations doesn't mean that we don't have to look at the data or the output!
- There's still plenty of engineering required in representation learning

Feature Engineering

- Often decried as “costly, hand-crafted, expensive, domain-specific”, etc.
- But in practice, simple features typically give the bulk of the performance

- Let’s get concrete: how should we define features for text classification?

Feature Engineering for Text Classification

$$\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) = \sum_i \theta_i f_i(\boldsymbol{x}, y)$$

Feature Engineering for Text Classification

$$\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) = \sum_i \theta_i f_i(\boldsymbol{x}, y)$$

$\boldsymbol{x}$ is now a vector because it is a sequence of words

Feature Engineering for Text Classification

$$\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) = \sum_i \theta_i f_i(\boldsymbol{x}, y)$$

$\boldsymbol{x}$ is now a vector because it is a sequence of words

let's consider sentiment analysis:
$y \in \{\text{positive}, \text{negative}\}$

Feature Engineering for Text Classification

$$\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) = \sum_i \theta_i f_i(\boldsymbol{x}, y)$$

$\boldsymbol{x}$ is now a vector because it is a sequence of words

let's consider sentiment analysis:
$y \in \{\text{positive}, \text{negative}\}$

so, here is our sentiment classifier that uses a linear model:

$$\text{classify}^{\text{linear}}_{\text{senti}}(\boldsymbol{x}, \boldsymbol{\theta}) = \underset{y \in \{\text{positive},\text{negative}\}}{\text{argmax}} \sum_i \theta_i f_i(\boldsymbol{x}, y)$$

Feature Engineering for Text Classification

$$\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) = \sum_i \theta_i f_i(\boldsymbol{x}, y)$$

- Two features:

$$f_1(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$
$$f_2(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

where $\mathbb{I}[S] = 1$ if S is true, 0 otherwise

Feature Engineering for Text Classification

$$\text{score}(\boldsymbol{x}, y, \boldsymbol{\theta}) = \sum_i \theta_i f_i(\boldsymbol{x}, y)$$

- Two features:

$$f_1(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$
$$f_2(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

where $\mathbb{I}[S] = 1$ if S is true, 0 otherwise

- What should the weights be?

$$\theta_1 > \theta_2? \qquad \theta_1 = \theta_2? \qquad \theta_1 < \theta_2?$$

Inference for Text Classification

$$\text{classify}^{\text{linear}}_{\text{senti}}(\boldsymbol{x}, \boldsymbol{\theta}) = \underset{y \in \{\text{positive}, \text{negative}\}}{\text{argmax}} \sum_i \theta_i f_i(\boldsymbol{x}, y)$$

inference: solve argmax

- trivial (loop over labels)

Text Classification

$$\text{classify}^{\text{linear}}_{\text{senti}}(\boldsymbol{x}, \boldsymbol{\theta}) = \underset{y \in \{\text{positive,negative}\}}{\text{argmax}} \sum_i \theta_i f_i(\boldsymbol{x}, y)$$

Learning for Text Classification

$$\text{classify}^{\text{linear}}_{\text{senti}}(\boldsymbol{x}, \boldsymbol{\theta}) = \underset{y \in \{\text{positive,negative}\}}{\text{argmax}} \sum_i \theta_i f_i(\boldsymbol{x}, y)$$

learning: choose $\boldsymbol{\theta}$

- There are many ways to choose $\boldsymbol{\theta}$

Experimental Practice

- in the beginning, we just had data

Experimental Practice

- in the beginning, we just had data
- first innovation: split into train and test
  - motivation: simulate conditions of applying system in practice

Experimental Practice

- in the beginning, we just had data
- first innovation: split into train and test
  - motivation: simulate conditions of applying system in practice
- but, there's a problem with this…

Experimental Practice

- in the beginning, we just had data
- first innovation: split into train and test
  - motivation: simulate conditions of applying system in practice
- but, there's a problem with this…
  - we need to explore and evaluate methodological choices
  - after multiple evaluations on test, it is no longer a simulation of real-world conditions

Experimental Practice

- we need to explore/evaluate methodological choices
- what should we do?
  - some use cross validation on train, but this is slow and doesn't quite simulate real-world settings (why?)

Experimental Practice

- we need to explore/evaluate methodological choices
- what should we do?
  - some use cross validation on train, but this is slow and doesn't quite simulate real-world settings (why?)
- second innovation: divide data into train, test, and a third set called development (dev) or validation (val)
  - use dev/val to evaluate choices
  - then, when ready to write the paper, evaluate the best model on test

Experimental Practice

- we need to explore/evaluate methodological choices
- what should we do?
  - some use cross validation on train, but this is slow and doesn't quite simulate real-world settings (why?)
- second innovation: divide data into train, test, and a third set called development (dev) or validation (val)
  - use dev/val to evaluate choices
  - then, when ready to write the paper, evaluate the best model on test
- are we done yet? no! there's still a problem:

Experimental Practice

- we need to explore/evaluate methodological choices
- what should we do?
  - some use cross validation on train, but this is slow and doesn't quite simulate real-world settings (why?)
- second innovation: divide data into train, test, and a third set called development (dev) or validation (val)
  - use dev/val to evaluate choices
  - then, when ready to write the paper, evaluate the best model on test
- are we done yet? no! there's still a problem:
  - overfitting to dev/val

Experimental Practice

- best practice: split data into train, development (dev), development test (devtest), and test
  - train model on train, tune hyperparameter values on dev, do preliminary testing on devtest, do final testing on test a single time when writing the paper
  - Even better to have even more test sets! test1, test2, etc.

- experimental credibility is a huge component of doing useful research
- when you publish a result, it had better be replicable without tuning anything on test

Don’t Cheat!

SECTIONS The New York Times SUBSCRIBE LOG IN

TECHNOLOGY

Computer Scientists Are Astir After Baidu Team Is Barred From A.I. Competition

By JOHN MARKOFF JUNE 3, 2015

Email

Share

Tweet

Save

SAN FRANCISCO — A group of researchers at the Chinese web services company Baidu have been barred from participating in an international competition for artificial intelligence technology after organizers discovered that the Baidu scientists broke the contest’s rules.