TTIC 31210:
Advanced Natural Language Processing

Kevin Gimpel

Spring 2019

Lecture 11:
Finish Learning in Structured Prediction; Probabilistic Modeling and Latent Variables

Roadmap

- intro (1 lecture)
- deep learning for NLP (5 lectures)
- **structured prediction (4.5 lectures)**
  - introducing/formalizing structured prediction, categories of structures
  - inference: dynamic programming, greedy algorithms, beam search
  - inference with non-local features
  - **learning in structured prediction**
- generative models, latent variables, unsupervised learning, variational autoencoders (1.5 lectures)
- Bayesian methods in NLP (2 lectures)
- Bayesian nonparametrics in NLP (2 lectures)
- review & other topics (1 lecture)

Assignments

- Assignment 1 grades were sent out today (sorry for the multiple emails)
- any questions about Assignment 3?
- after the break, we'll go over Assignment 2 solutions briefly

Inference: Summary

- exact DP algorithms if parts are small
- beam search
- coarse-to-fine
- gradient descent for inference
- inference networks
- linear programming / ILP

Learning in Structured Prediction

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

learning: choose $\boldsymbol{\theta}$

- most loss functions used in structured prediction have the same form as those used in multi-class classification
- part that changes: now structured inference is required for computing gradients
- we can use any inference strategy we discussed in the context of learning
- there are also new inference problems that arise for certain loss functions

Cost Functions

- **cost function**: how different are these two structures?

$$\mathrm{cost} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$$

- typically used to compare predicted structure to gold standard
- should reflect evaluation metric for task

Cost Functions

- typical cost for multi-class classification:

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

- how about for sequences? $\text{cost} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$
  - "Hamming cost": $$\text{cost}(\boldsymbol{y}, \boldsymbol{y}') = \sum_{t=1}^{|\boldsymbol{y}|} \mathbb{I}[y_t \neq y'_t]$$
  - "0-1 cost": $$\text{cost}(\boldsymbol{y}, \boldsymbol{y}') = \mathbb{I}[\boldsymbol{y} \neq \boldsymbol{y}']$$

Empirical Risk Minimization

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}} \sum_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{D}} \text{cost}(\boldsymbol{y}, \text{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$$

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

- this is intractable so we typically minimize a **surrogate loss function** instead

Loss Functions for Structured Prediction

$\boldsymbol{x}, \boldsymbol{y}$ form a structured input/output pair in the training data

name	loss	where used
cost ("0-1")	$\mathrm{cost}(\boldsymbol{y}, \mathrm{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)

Loss Functions for Structured Prediction

$\boldsymbol{x}, \boldsymbol{y}$ form a structured input/output pair in the training data

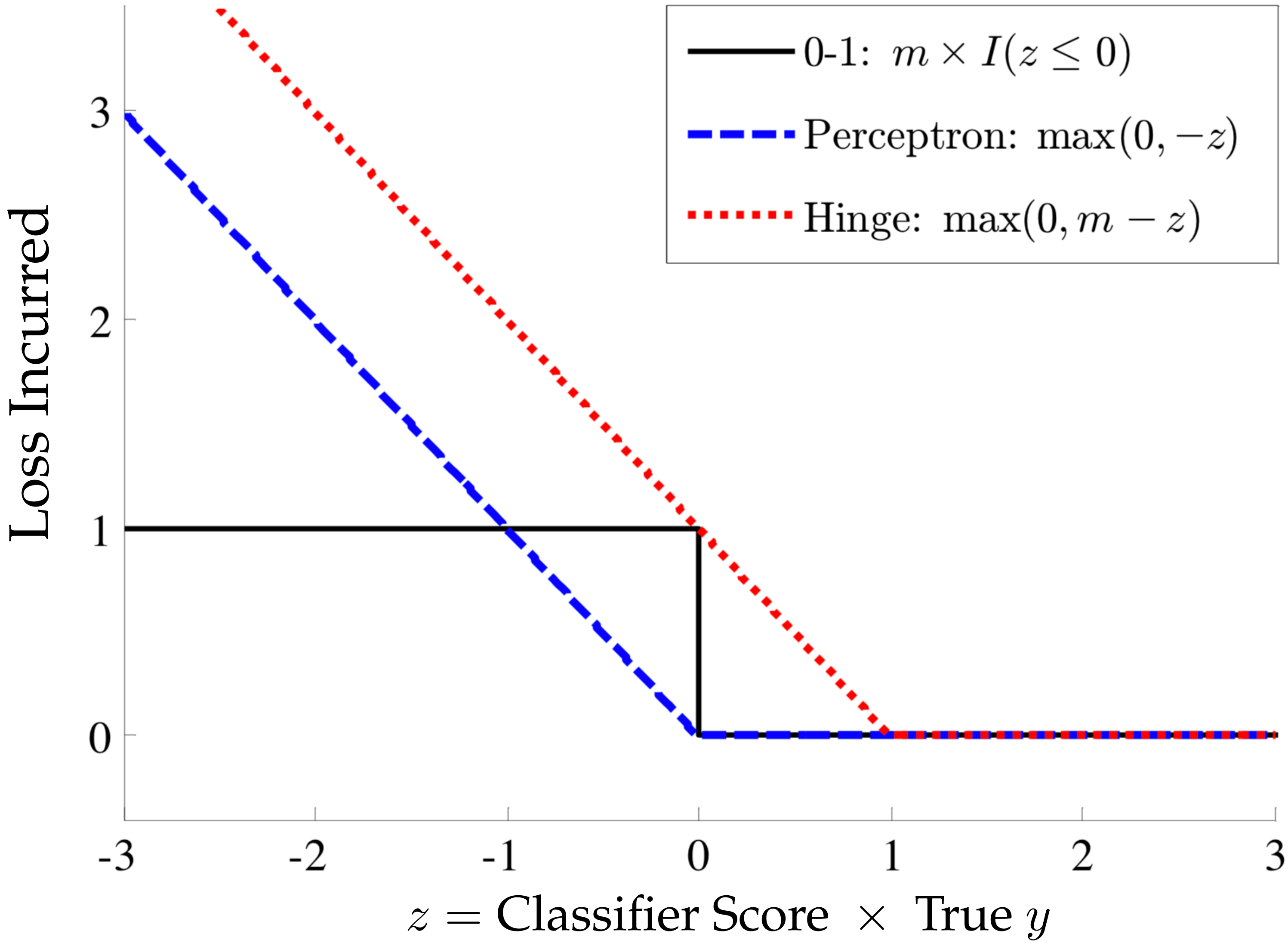
name	loss	where used
cost ("0-1")	$\text{cost}(\boldsymbol{y}, \text{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)
percep-tron	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$	structured perceptron (Collins, 2002)

Loss Functions for Structured Prediction

$\boldsymbol{x}, \boldsymbol{y}$ form a structured input/output pair in the training data

name	loss	where used
cost ("0-1")	$\mathrm{cost}(\boldsymbol{y}, \mathrm{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)
percep-tron	$-\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \mathrm{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$	structured perceptron (Collins, 2002)
hinge	$-\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} (\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \mathrm{cost}(\boldsymbol{y}, \boldsymbol{y}'))$	structured SVMs (Taskar et al., *inter alia*)

Visualizing Losses for Binary Classification

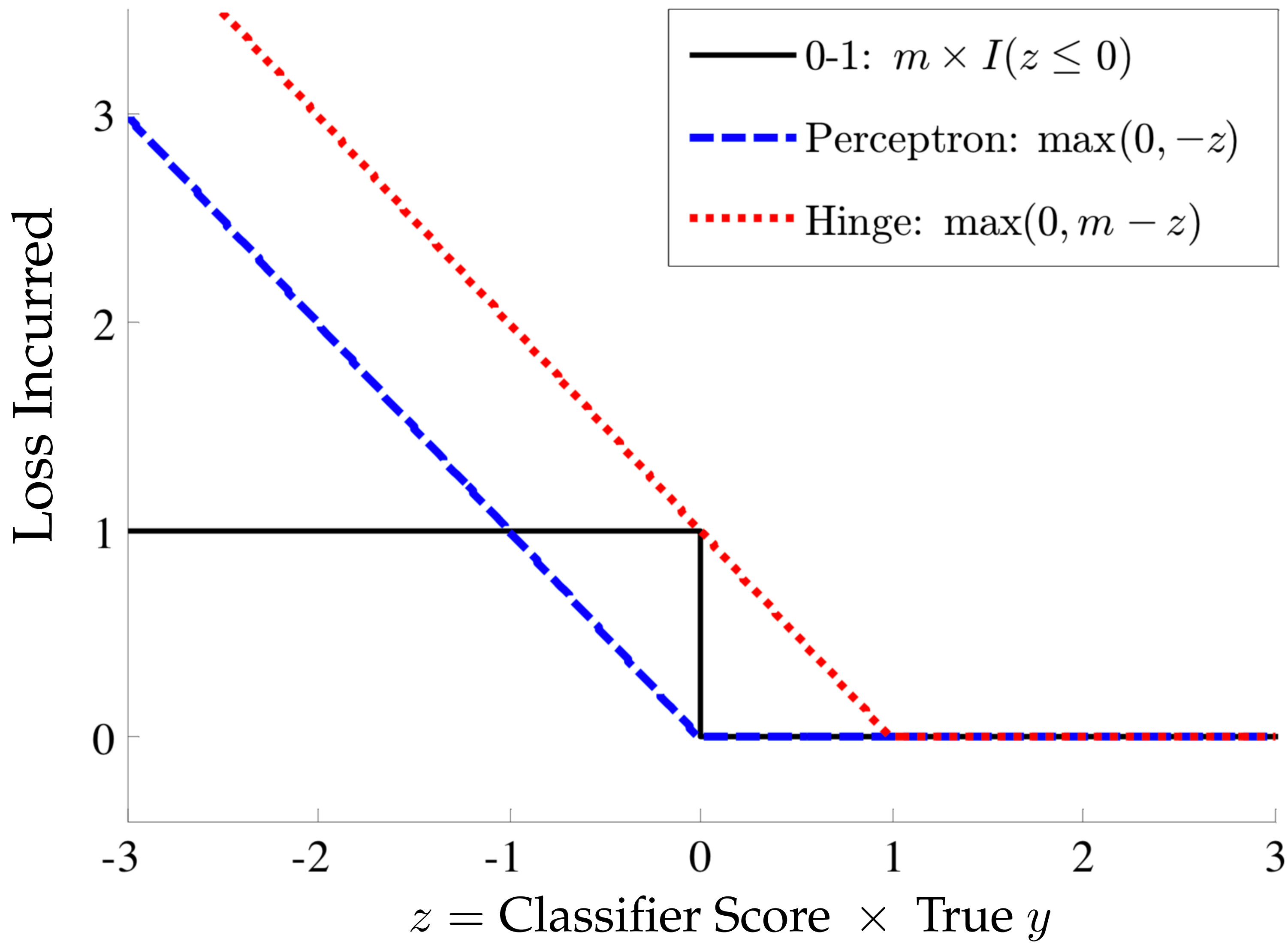


m = cost multiplier

this is for binary classification, so true *y* is either -1 or 1

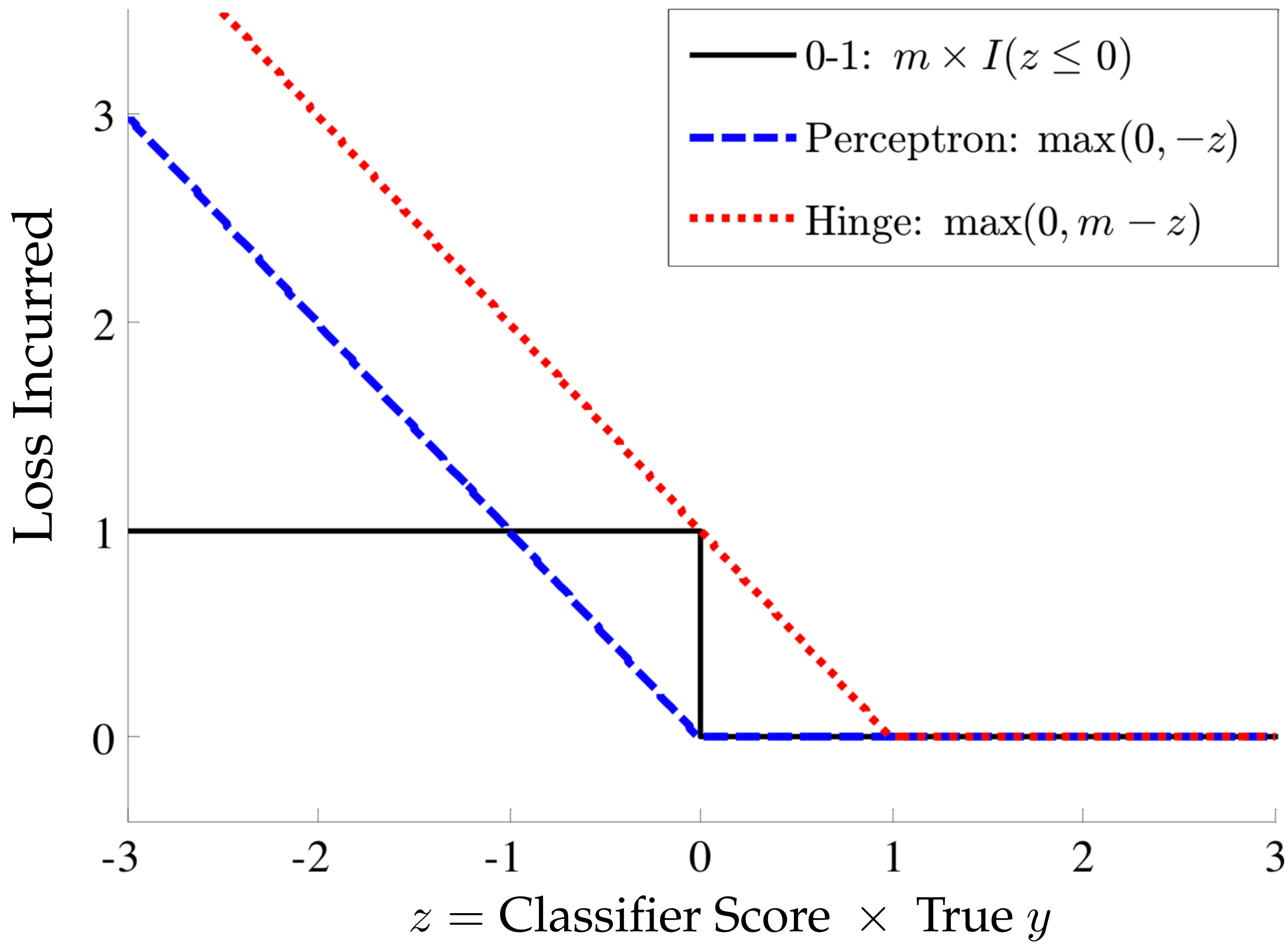
the larger the classifier score is, the more confident the classifier is

Visualizing Losses for Binary Classification



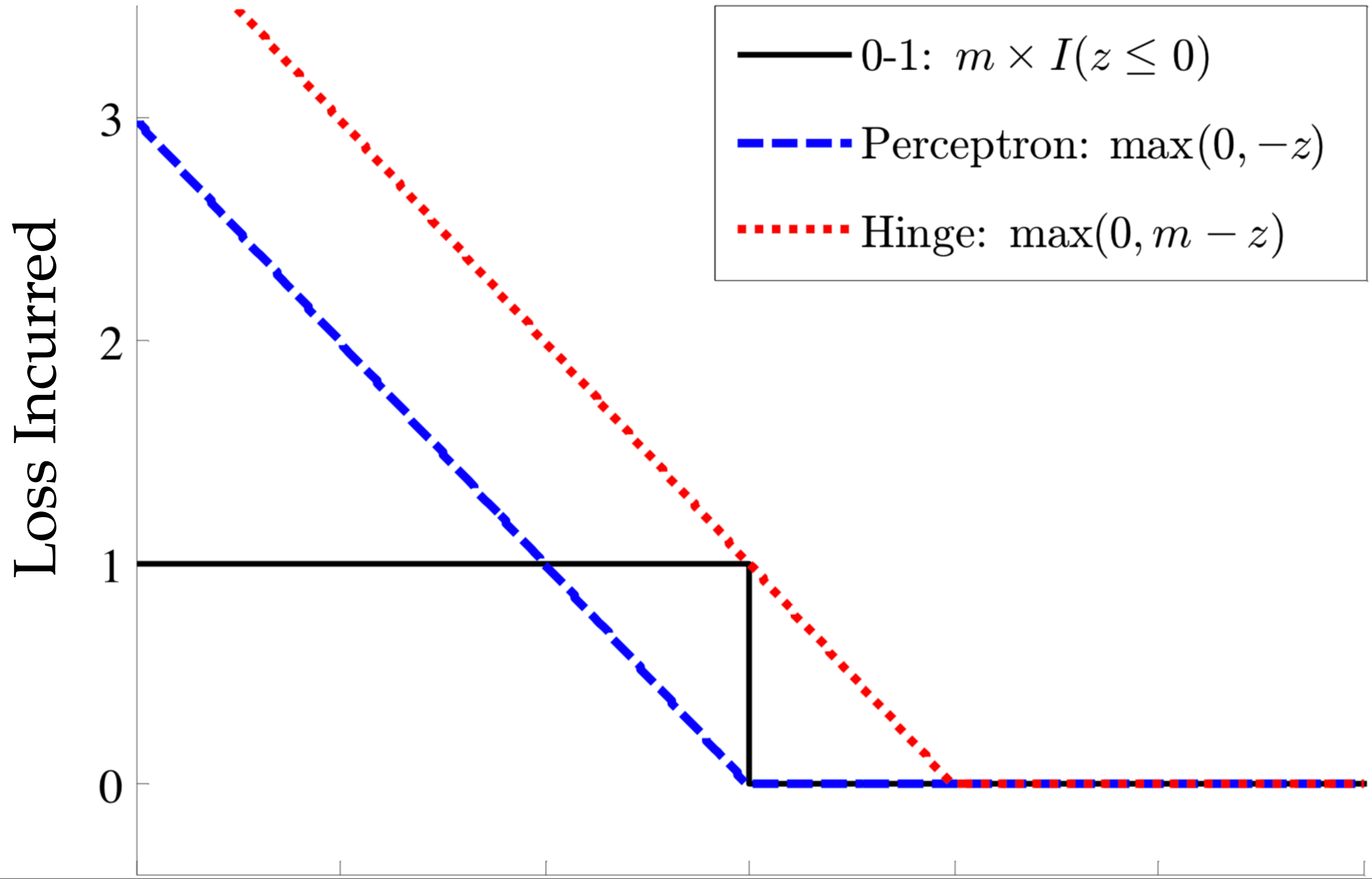
with 0-1 loss: if classifier is correct, what is the loss?

Visualizing Losses for Binary Classification



with 0-1 loss: if classifier is correct, what is the loss? 0

Visualizing Losses for Binary Classification

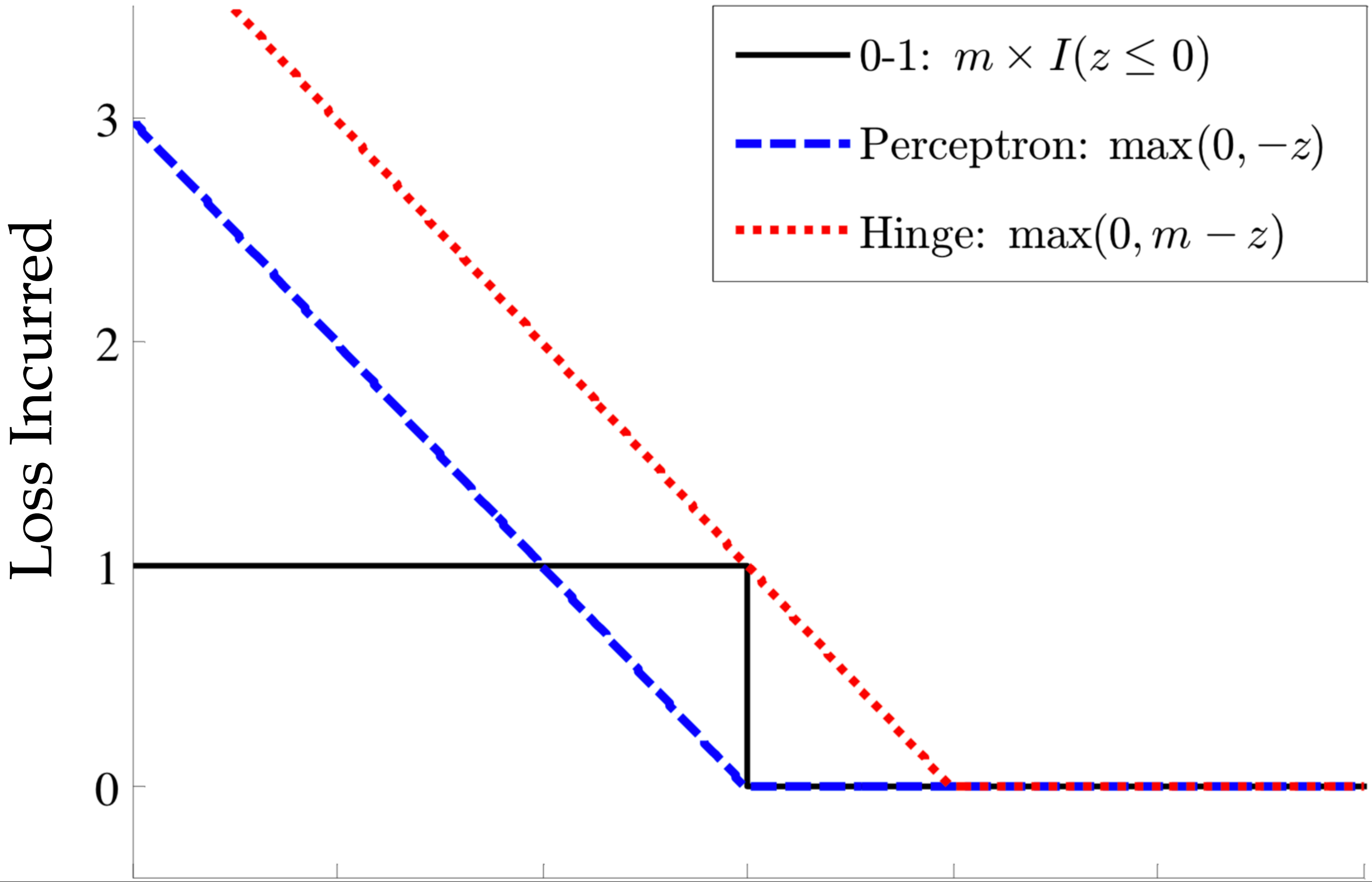


with perceptron loss:

if classifier is correct, what is the loss?

if classifier is incorrect, what is the loss?

Visualizing Losses for Binary Classification

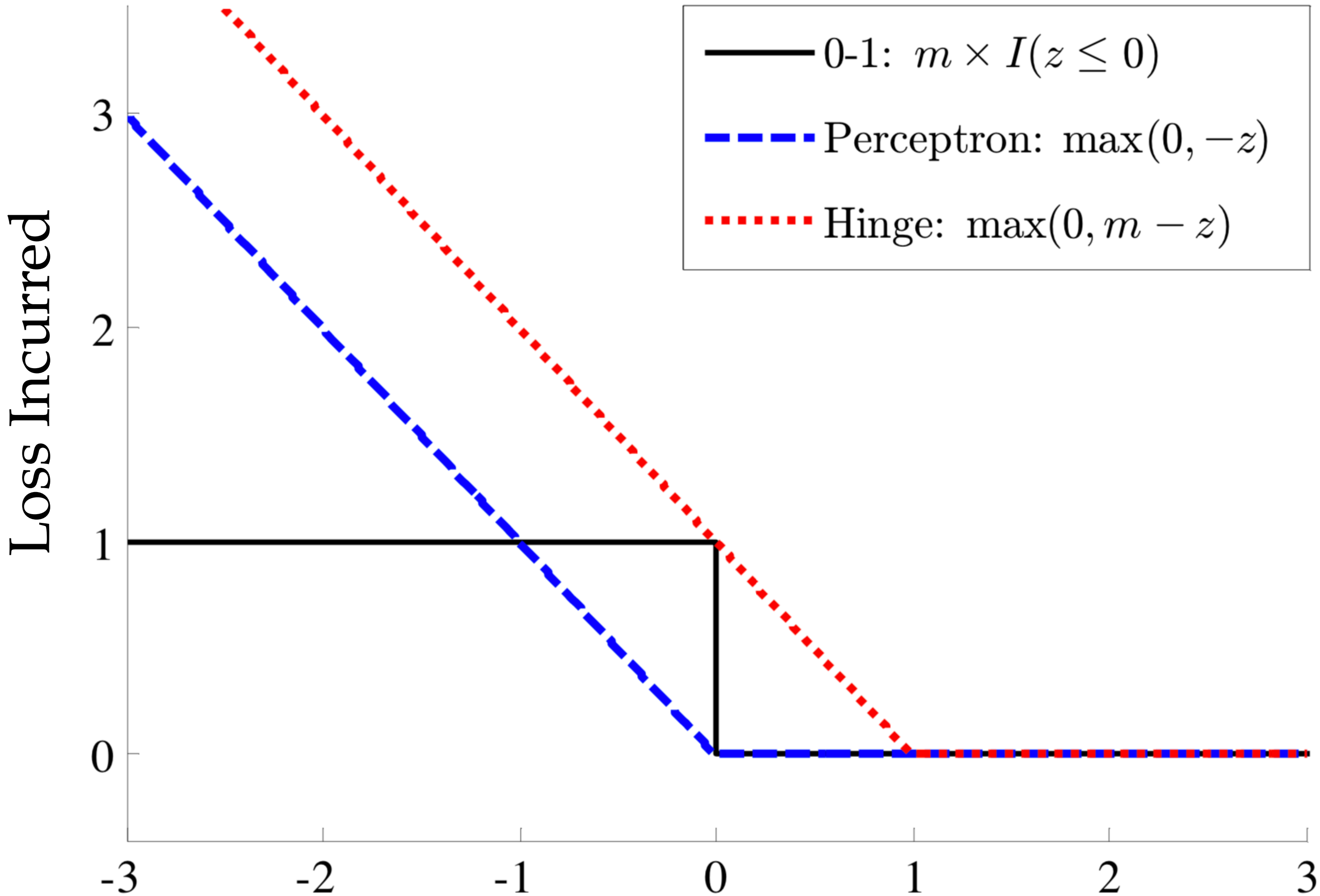


with perceptron loss:

if classifier is correct, what is the loss? 0

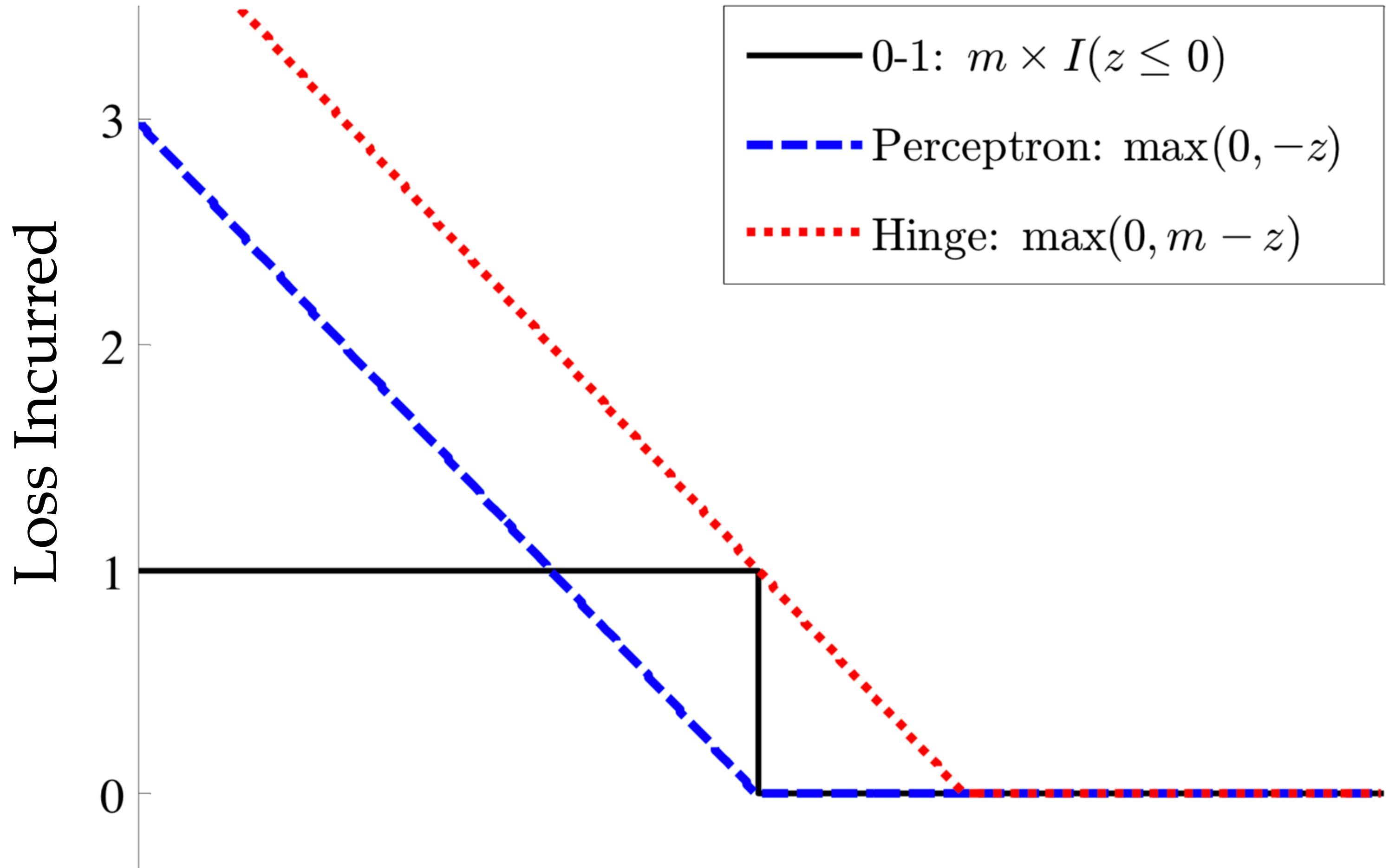
if classifier is incorrect, what is the loss? classifier score

Visualizing Losses for Binary Classification



with hinge loss:
if classifier is correct, what is the loss?
if classifier is incorrect, what is the loss?

Visualizing Losses for Binary Classification



with hinge loss:

if classifier is correct, what is the loss? 0 if classifier score bigger than *m,* else *m* minus classifier score

if classifier is incorrect, what is the loss? classifier score + *m*

Loss Functions for Structured Prediction

$\boldsymbol{x}, \boldsymbol{y}$ form a structured input/output pair in the training data

name	loss	where used
cost ("0-1")	$\text{cost}(\boldsymbol{y}, \text{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)
percep-tron	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$	structured perceptron (Collins, 2002)
hinge	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} (\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}'))$	structured SVMs (Taskar et al., *inter alia*)
log	$-\log p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})$	CRFs (Lafferty et al., 2001)

Loss Functions for Structured Prediction

$\boldsymbol{x}, \boldsymbol{y}$ form a structured input/output pair in the training data

name	loss	where used
cost ("0-1")	$\mathrm{cost}(\boldsymbol{y}, \mathrm{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)
percep-tron	$-\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \ \mathrm{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$	structured perceptron (Collins, 2002)
hinge	$-\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \ (\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \mathrm{cost}(\boldsymbol{y}, \boldsymbol{y}'))$	structured SVMs (Taskar et al., *inter alia*)
log	$-\log p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = -\log \frac{\exp\{\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta})\}}{\sum_{\boldsymbol{y}'} \exp\{\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})\}}$	CRFs (Lafferty et al., 2001)

$$= -\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \exp\{\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})\}$$

note: this uses the usual softmax transformation to convert scores to probabilities, but the summation is over all output structures

New Inference Problem: Summing

$$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \exp\left\{\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})\right\}$$

- to compute gradients of this loss, we need to sum over all structured outputs
- for certain structures, we can do this efficiently using DP algorithms

Viterbi Algorithm for HMMs

$$V(1, y) = p_{\boldsymbol{\eta}}(x_1 \mid y)\, p_{\boldsymbol{\tau}}(y \mid \text{<s>})$$
$$V(m, y) = \max_{y' \in \mathcal{L}} \left(p_{\boldsymbol{\eta}}(x_m \mid y)\, p_{\boldsymbol{\tau}}(y \mid y')\, V(m-1, y') \right)$$

Viterbi efficiently iterates over **all** label sequences in polynomial time

can we repurpose this algorithm to efficiently sum over all label sequences?

Viterbi → "Forward" Algorithm

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

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

just change max to sum!

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

$$F(m, y) = \sum_{y' \in \mathcal{L}} p_{\boldsymbol{\eta}}(x_m \mid y)\, p_{\boldsymbol{\tau}}(y \mid y')\, F(m-1, y')$$

Forward-Backward Algorithm?

- we used to derive a second algorithm ("backward" or "outside") and use both algorithms to compute expected counts/posteriors/gradients
- now, we just implement the forward DP algorithm with memoization directly using computation graphs, then use autodifferentiation
- this is not new, though for many years it was not mainstream

see Eisner (2016): *Inside-Outside and Forward-Backward Algorithms Are Just Backprop*

Summing Over Structures

- when parts are small, we can repurpose max DP algorithms for summing
- how about when parts are big or DP is too slow?
- "approximate summing" is much trickier than approximate argmax
- depending on what you want to do in the summation, you may have to be careful about bias (e.g., if you're estimating expectations)

Loss Functions for Structured Prediction

$\boldsymbol{x}, \boldsymbol{y}$ form a structured input/output pair in the training data

name	loss	where used
cost ("0-1")	$\mathrm{cost}(\boldsymbol{y}, \mathrm{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)
percep-tron	$-\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \mathrm{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$	structured perceptron (Collins, 2002)
hinge	$-\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} (\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \mathrm{cost}(\boldsymbol{y}, \boldsymbol{y}'))$	structured SVMs (Taskar et al., *inter alia*)
log	$-\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \exp\{\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})\}$	CRFs (Lafferty et al., 2001)
softmax margin	$-\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \exp\{\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \mathrm{cost}(\boldsymbol{y}, \boldsymbol{y}')\}$	Povey et al. (2008), Gimpel & Smith (2010)

Relationships Among Losses

$$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$$

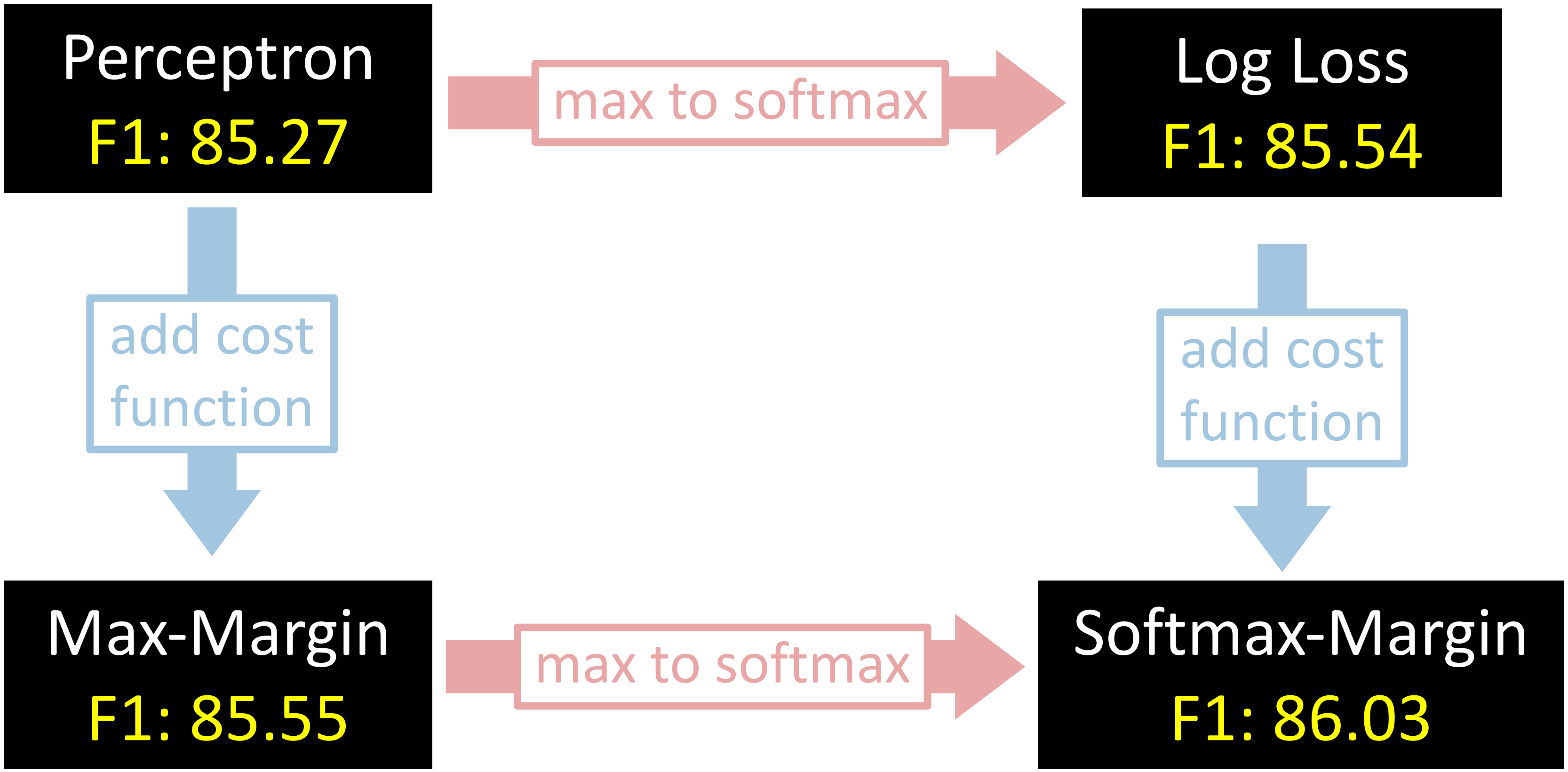
$$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \exp\{\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})\}$$

$$\begin{tikzcd}
\text{perceptron loss} \arrow[r, "\text{max to softmax}"] \arrow[d, "\text{add cost function}"] & \text{log loss} \arrow[d, "\text{add cost function}"] \\
\text{max-margin} \arrow[r, "\text{max to softmax}"] & \text{softmax-margin}
\end{tikzcd}$$

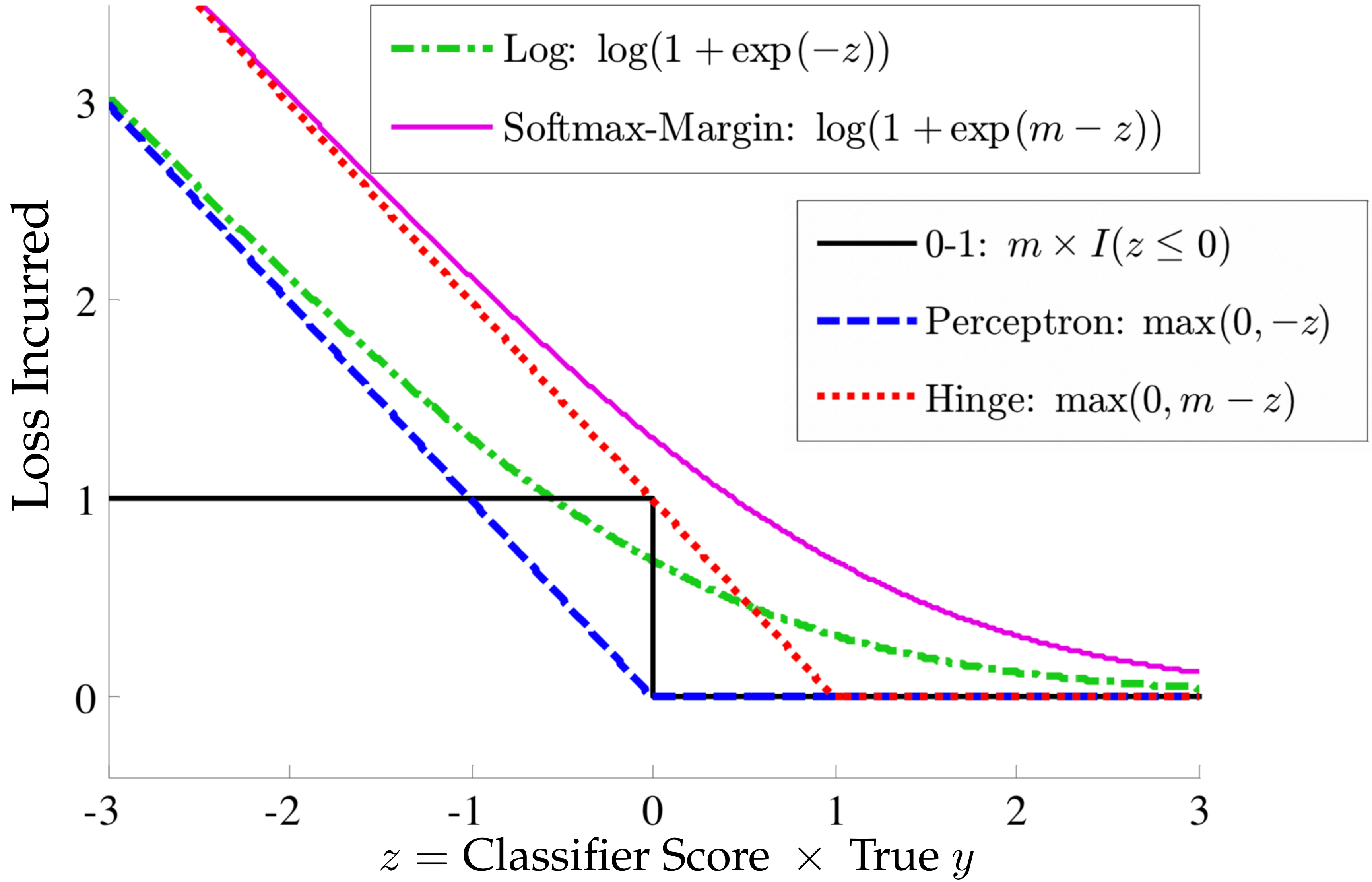
$$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \left(\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}')\right)$$

$$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \exp\{\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}')\}$$

Results: Named Entity Recognition



Visualizing Losses for Binary Classification



m = cost multiplier

this is for binary classification, so true *y* is either -1 or 1

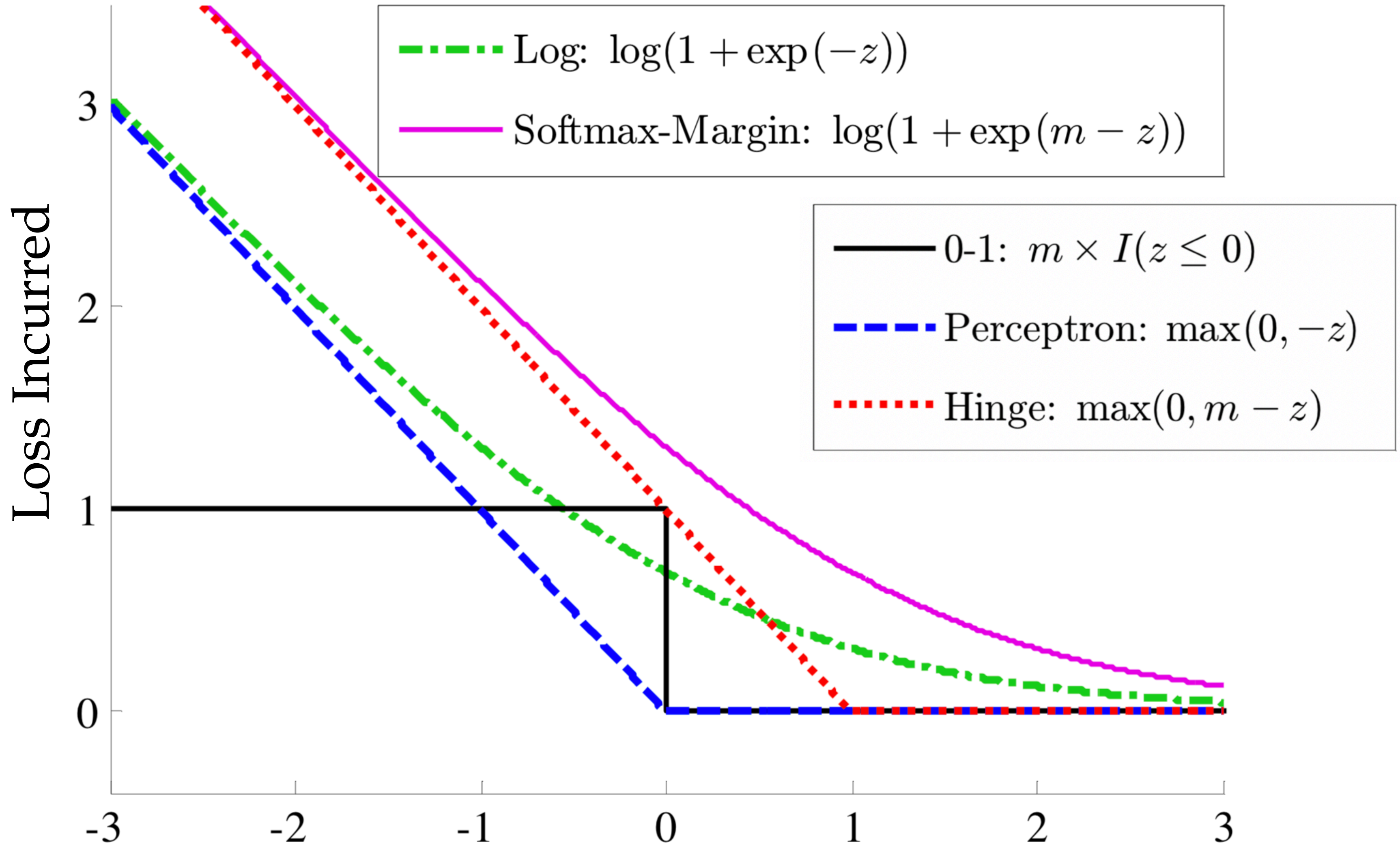
the larger the classifier score is, the more confident the classifier is

New Inference Problem: Cost-Augmented Summing

$$-\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \exp\left\{\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \mathrm{cost}(\boldsymbol{y}, \boldsymbol{y}')\right\}$$

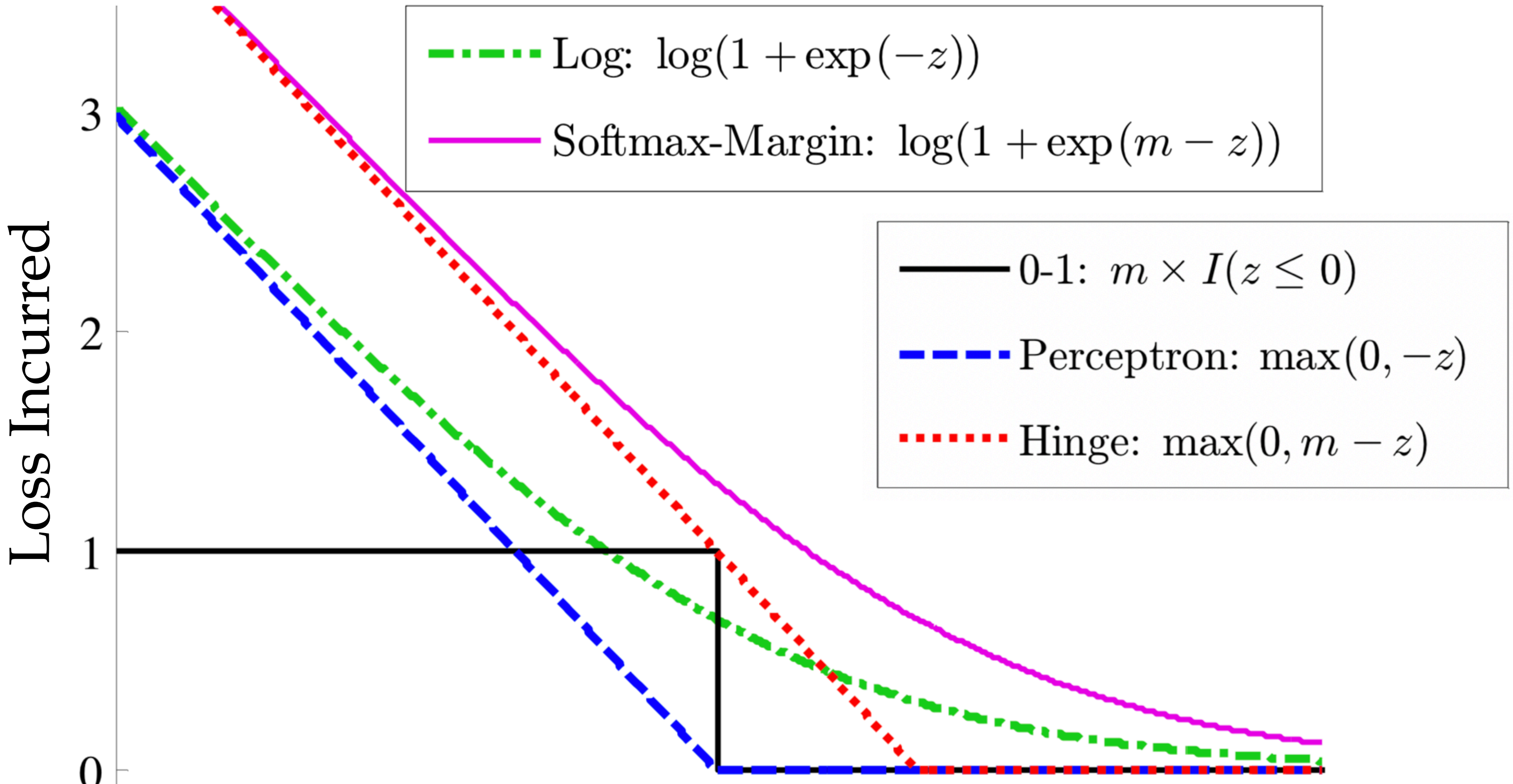
- if cost function decomposes additively like the score function (i.e., if cost and score functions use same parts), we can use same algorithms as for log loss

Visualizing Losses for Binary Classification



- these 4 surrogate loss functions are convex
- good for optimization, but any potential problems?

Visualizing Losses for Binary Classification



- these 4 surrogate loss functions are convex
- good for optimization, but:
  - loose approximations to 0-1 loss
  - may be sensitive to outliers

Non-Convex Surrogate Losses

- risk (also called Bayes risk)

Risk

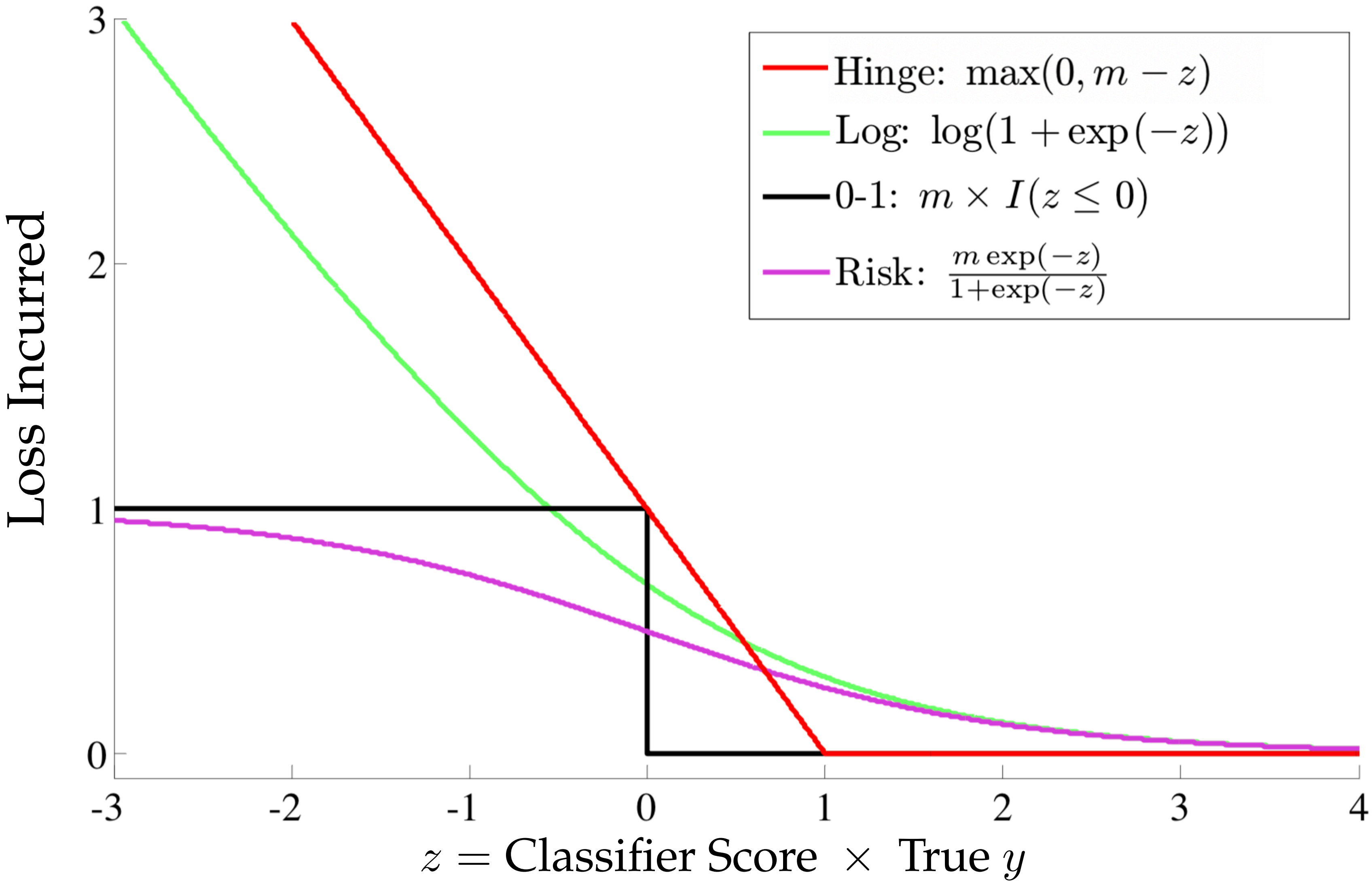
- expectation under the model of the cost function
- has been used for speech & machine translation (Kaiser et al., 2000; Smith & Eisner, 2006)

$$\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{y}' \mid \boldsymbol{x})}\left[\text{cost}(\boldsymbol{y}, \boldsymbol{y}')\right]$$

where model probability is produced
using a softmax over structured outputs,
just like with log loss:

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

Visualizing Risk for Binary Classification



risk is non-convex and tracks 0-1 loss more closely than the convex losses

Risk

$$\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{y}'|\boldsymbol{x})}\left[\text{cost}(\boldsymbol{y},\boldsymbol{y}')\right] = \sum_{\boldsymbol{y}'} \text{cost}(\boldsymbol{y},\boldsymbol{y}') \frac{\exp\{\text{score}(\boldsymbol{x},\boldsymbol{y}',\boldsymbol{\theta})\}}{\sum_{\boldsymbol{y}''}\exp\{\text{score}(\boldsymbol{x},\boldsymbol{y}'',\boldsymbol{\theta})\}}$$

- how can we compute gradients for optimizing this?
- for log loss, we only needed a summing algorithm
- for risk, we need to compute expectations of products (intuitively: need to track pairs of parts)
- there are DP algorithms that can be used to efficiently compute gradients for risk (and related quantities)

Li & Eisner (2009): *First- and second-order expectation semirings with applications to minimum-risk training on translation forests*

Xiong et al. (2009): *Minimum tag error for discriminative training of conditional random fields*

Minimum Risk Training for Machine Translation

- used for MT by Smith & Eisner (2006) and neural MT by Shen et al. (2016)
- found effective in large-scale comparison by Edunov et al. (2018)
- most use a cost function related to BLEU score
- approximate sums using *n*-best lists:

$$\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{y}'|\boldsymbol{x})}[\mathrm{cost}(\boldsymbol{y}, \boldsymbol{y}')] = \sum_{\boldsymbol{y}'} \mathrm{cost}(\boldsymbol{y}, \boldsymbol{y}') \frac{\exp\{\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})\}}{\sum_{\boldsymbol{y}''} \exp\{\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}'', \boldsymbol{\theta})\}}$$

sums approximated
using *n*-best lists

Loss Functions for Structured Prediction

name	loss	where used
cost ("0-1")	$\text{cost}(\boldsymbol{y}, \text{predict}(\boldsymbol{x}, \boldsymbol{\theta}))$	MERT (Och, 2003)
percep-tron	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \ \text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$	structured perceptron (Collins, 2002)
hinge	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \ (\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}'))$	structured SVMs (Taskar et al., *inter alia*)
log	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \ \exp\{\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})\}$	CRFs (Lafferty et al., 2001)
softmax margin	$-\text{score}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) + \log \sum_{\boldsymbol{y}'} \ \exp\{\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}')\}$	Povey et al. (2008), Gimpel & Smith (2010)
risk	$\sum_{\boldsymbol{y}'} \text{cost}(\boldsymbol{y}, \boldsymbol{y}') \dfrac{\exp\{\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})\}}{\sum_{\boldsymbol{y}''} \exp\{\text{score}(\boldsymbol{x}, \boldsymbol{y}'', \boldsymbol{\theta})\}}$	Kaiser et al., (2000); Smith & Eisner (2006)

Non-Convex Surrogate Losses

- risk
- ramp

Ramp Loss

- what is this loss doing?

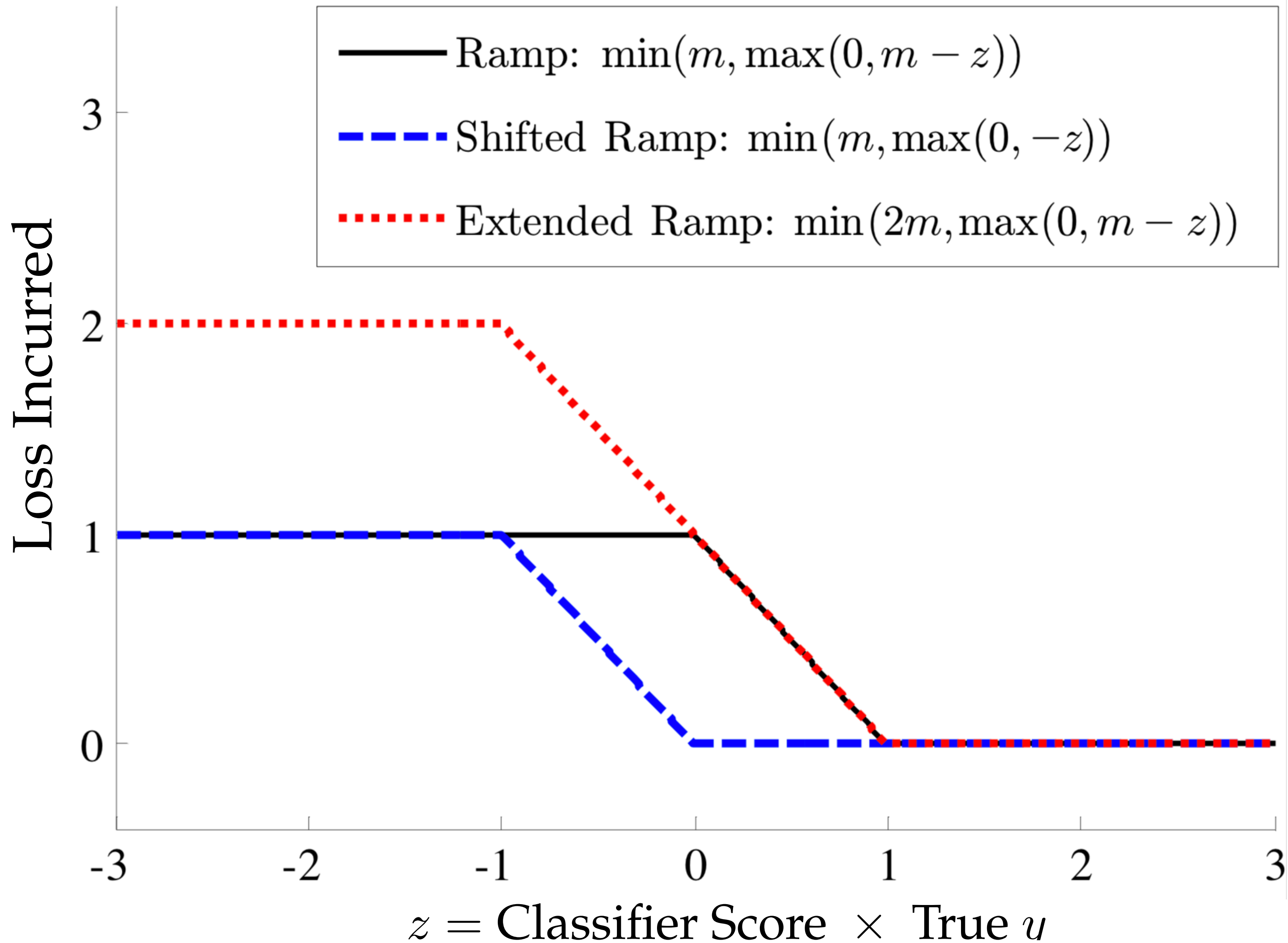
$$-\max_{\boldsymbol{y}''} \text{score}(\boldsymbol{x}, \boldsymbol{y}'', \boldsymbol{\theta}) + \max_{\boldsymbol{y}'} \left(\text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta}) + \text{cost}(\boldsymbol{y}, \boldsymbol{y}')\right)$$

Do et al. (2008): *Tighter bounds for structured estimation*

- second form of ramp loss:

$$-\max_{\boldsymbol{y}''} \left(\text{score}(\boldsymbol{x}, \boldsymbol{y}'', \boldsymbol{\theta}) - \text{cost}(\boldsymbol{y}, \boldsymbol{y}'')\right) + \max_{\boldsymbol{y}'} \text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$$

Ramp Losses for Binary Classification



m = cost multiplier

this is for binary classification, so true *y* is either -1 or 1

the larger the classifier score is, the more confident the classifier is

New Inference Problem: Cost-Diminished Inference

$$-\max_{\boldsymbol{y}''} \left(\text{score}(\boldsymbol{x}, \boldsymbol{y}'', \boldsymbol{\theta}) - \text{cost}(\boldsymbol{y}, \boldsymbol{y}'')\right) + \max_{\boldsymbol{y}'} \text{score}(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{\theta})$$

- can use same algorithms as cost-augmented inference

Learning in Structured Prediction: Summary

- losses have same form as in multi-class classification
- two things change:
  - structured inference is required during learning, and it can take various forms
  - cost function usually defined to decompose across parts of structured output
    - in multi-class classification, cost is much simpler

Learning in Structured Prediction: Summary

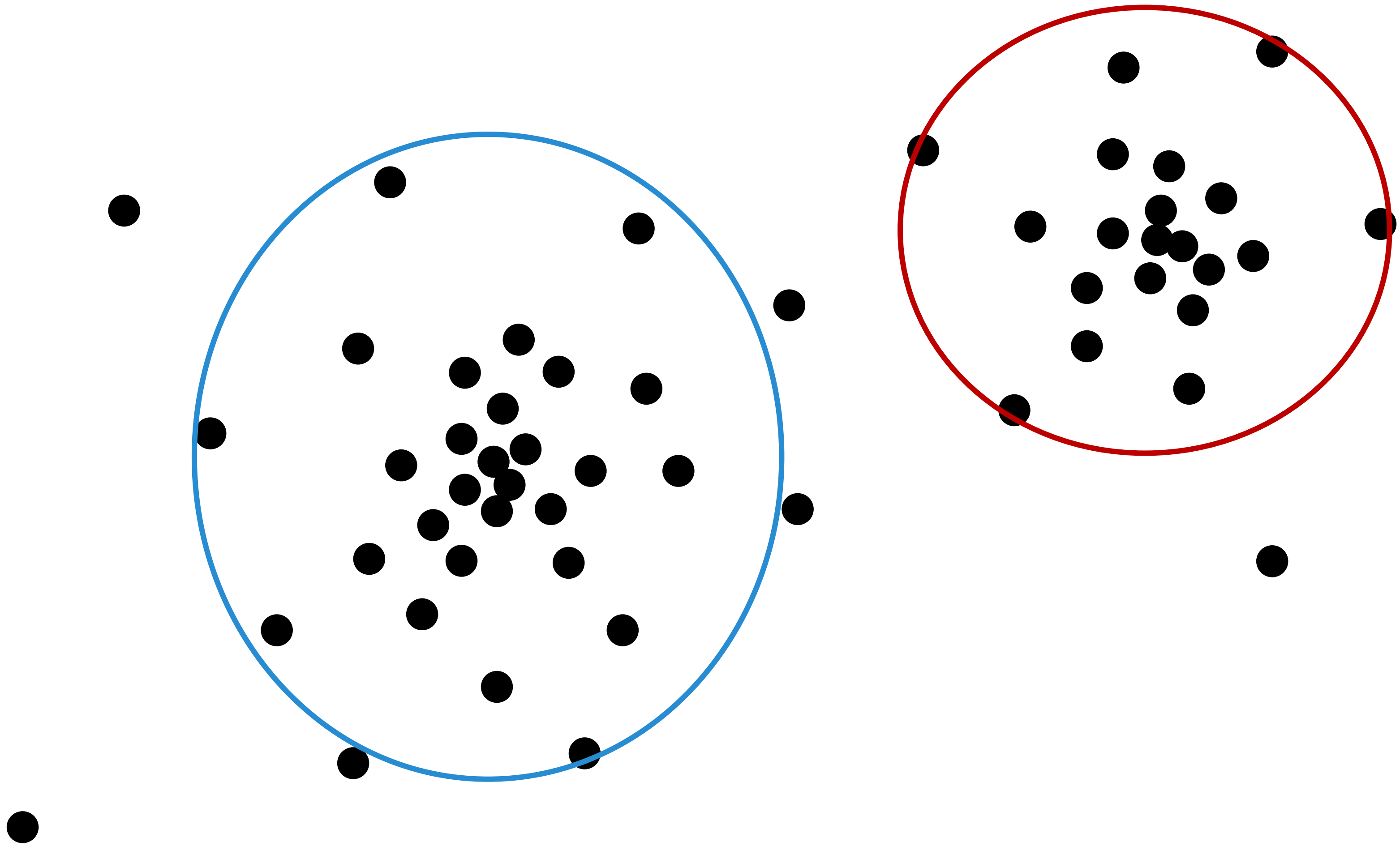
- computational bottleneck in learning is inference
- various forms of inference:
  - perceptron: argmax inference
  - hinge: cost-augmented inference
  - log: summation over structures
  - softmax-margin: cost-augmented summation
  - risk: expectations of products
  - ramp: cost-augmented/cost-diminished inference
- argmax inference is easier/faster than summing inference, so perceptron/hinge losses are commonly used in structured prediction

Roadmap

- intro (1 lecture)
- deep learning for NLP (5 lectures)
- structured prediction (4.5 lectures)
- **generative models, latent variables, unsupervised learning, variational autoencoders (1.5 lectures)**
- Bayesian methods in NLP (2 lectures)
- Bayesian nonparametrics in NLP (2 lectures)
- review & other topics (1 lecture)

- NLP historically has had a lot of probabilistic modeling with latent variables
- sometimes supervised, sometimes unsupervised
- unsupervised learning in NLP often takes the form: “consider the unseen output as a latent variable”

Prototypical Latent-Variable Model: Clustering



Latent-Variable Modeling

- why would we want to introduce latent variables in our models?
- we may want to assume there is some unseen ("latent" or "hidden"), underlying structure in the data-generating process
- this latent structure can help us in defining the generative model of the data
- e.g., clustering

"Brown Clustering"

Class-Based *n*-gram Models of Natural Language

Peter F. Brown*
Peter V. deSouza*
Robert L. Mercer*
Vincent J. Della Pietra*
Jenifer C. Lai*

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays

June March July April January December October November September August

people guys folks fellows CEOs chaps doubters commies unfortunates blokes

down backwards ashore sideways southward northward overboard aloft downwards adrift

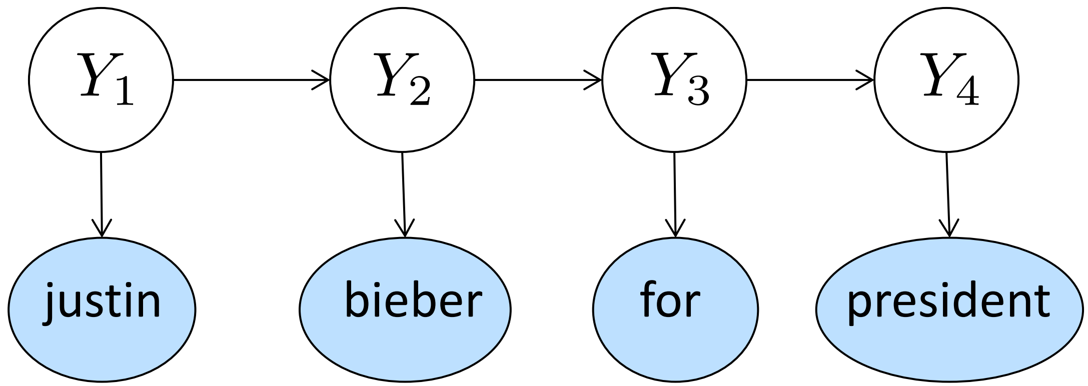
water gas coal liquid acid sand carbon steam shale iron

great big vast sudden mere sheer gigantic lifelong scant colossal

Computational Linguistics, 1992

HMMs for Word Clustering

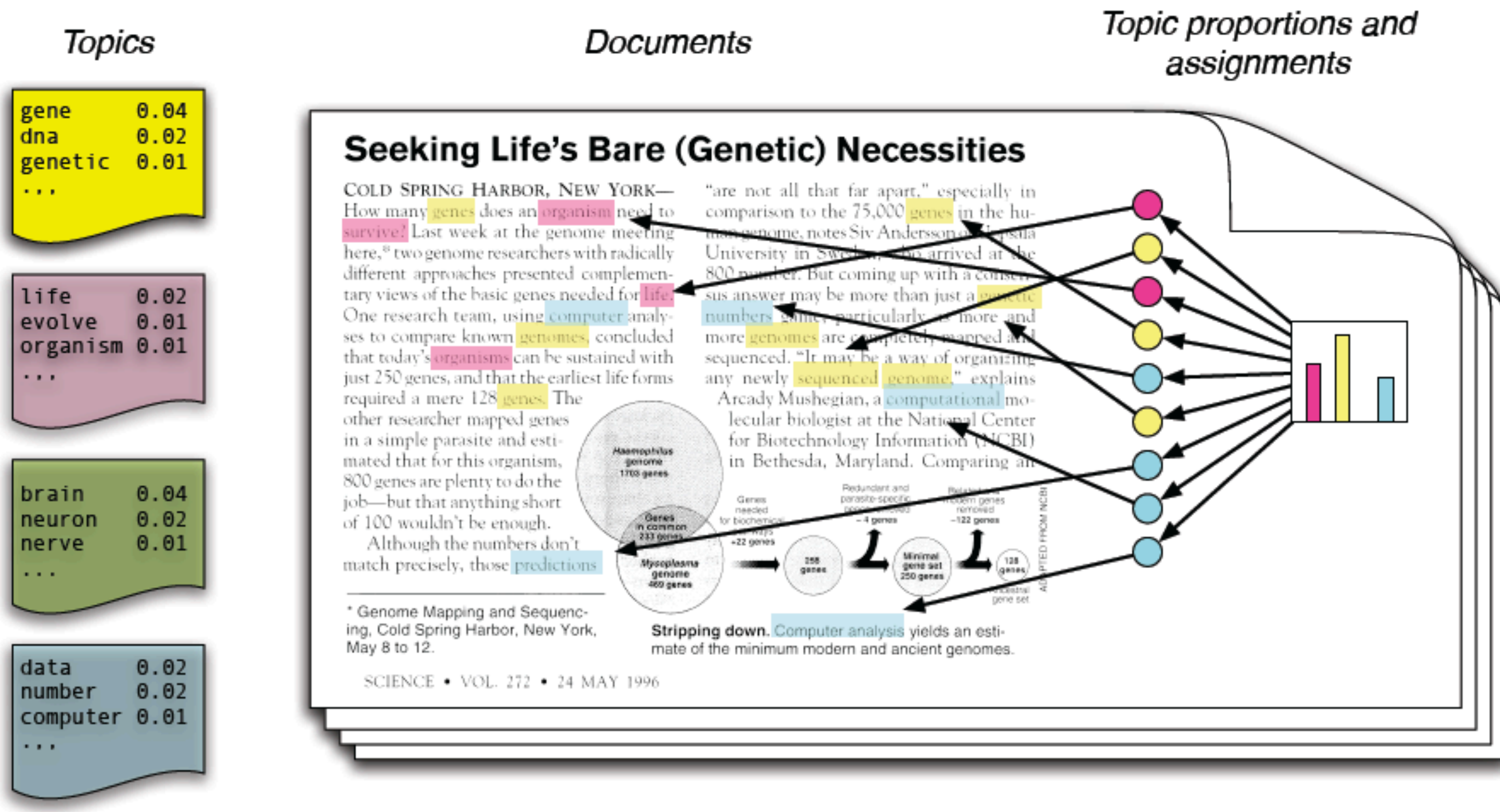
(Brown et al., 1992)



each $y_i \in \mathcal{L}$ is a cluster ID

so, label space is $\mathcal{L} = \{1, 2, ..., 100\}$

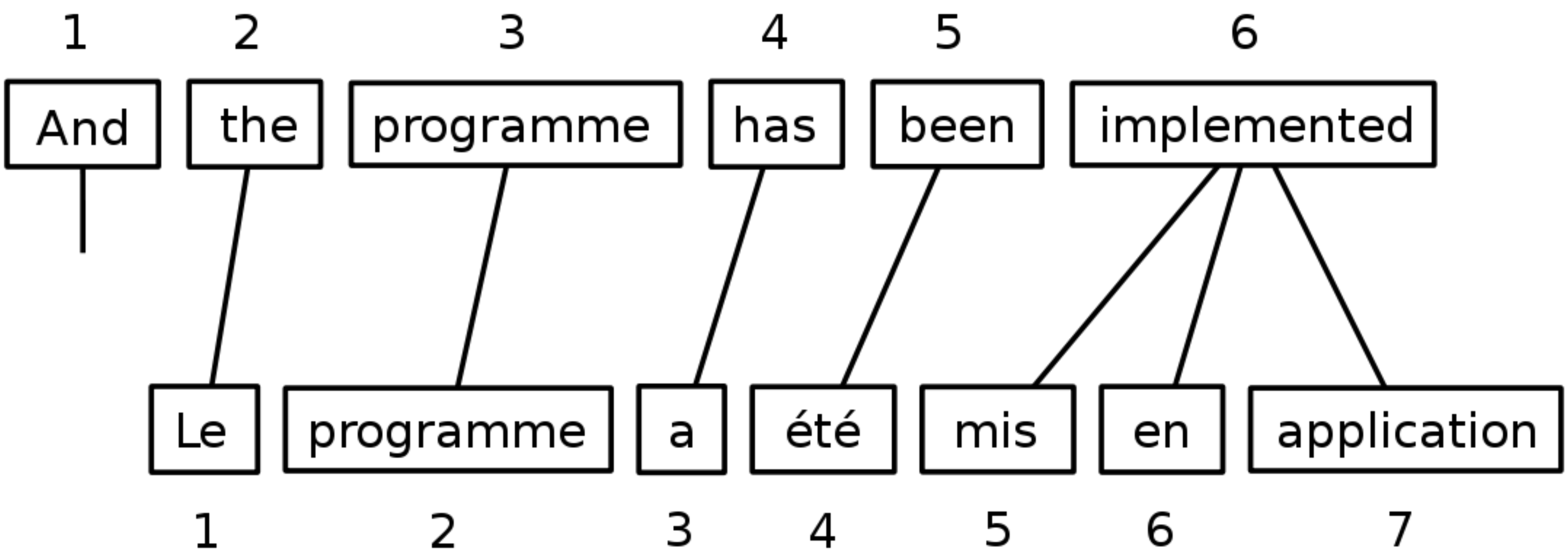
Topic Modeling



Blei et al. (2003)

Word Alignment

parallel sentences are observed, word alignments are latent variables:



Brown et al. (1990)

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	.

Unsupervised Part-of-Speech Tagging

1	2	4	3	3	4	5	2
Some	questioned	if	Tim	Cook	’s	first	product

2	4	1	5	2	4	2	1
would	be	a	breakaway	hit	for	Apple	.

sentences are observed, part-of-speech tags are treated as latent variables

Unsupervised Part-of-Speech Tagging

1	2	4	3	3	4	5	2
Some	questioned	if	Tim	Cook	’s	first	product

2	4	1	5	2	4	2	1
would	be	a	breakaway	hit	for	Apple	.

1-to-1 accuracy:

1 ➔ determiner
2 ➔ verb
3 ➔ noun
…

Unsupervised Part-of-Speech Tagging

1	2	4	3	3	4	5	2
Some	questioned	if	Tim	Cook	’s	first	product

2	4	1	5	2	4	2	1
would	be	a	breakaway	hit	for	Apple	.

1-to-1 accuracy:

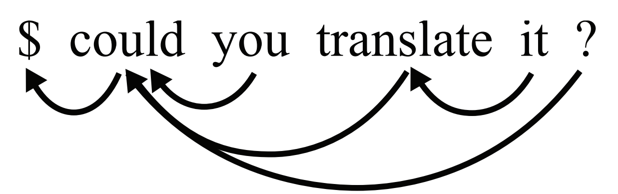
1 ➔ determiner
2 ➔ verb
3 ➔ noun
…

many-to-1 accuracy:

1 ➔ determiner
2 ➔ noun
3 ➔ noun
…

Unsupervised Dependency Parsing

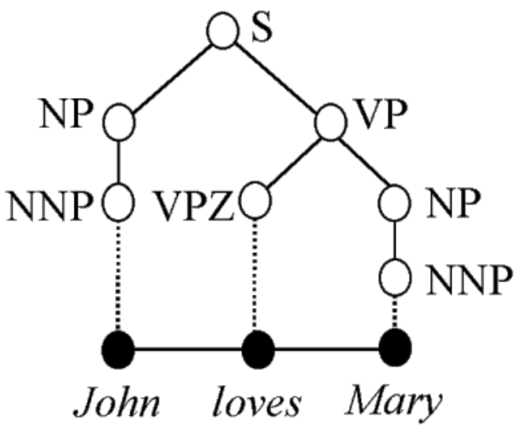
sentences are observed, dependency parse trees are treated as latent variables:



Klein & Manning
Smith & Eisner

Latent Syntactic Categories for Parsing

- split Penn Treebank syntactic categories into finer subcategories



NNP

NNP-0	Jr.	Goldman	INC.
NNP-1	Bush	Noriega	Peters
NNP-2	J.	E.	L.
NNP-3	York	Francisco	Street
NNP-4	Inc	Exchange	Co
NNP-5	Inc.	Corp.	Co.

RB

RB-0	recently	previously	still
RB-1	here	back	now
RB-2	very	highly	relatively
RB-3	so	too	as
RB-4	also	now	still
RB-5	however	Now	However

Petrov (2009)

Morphological Segmentation

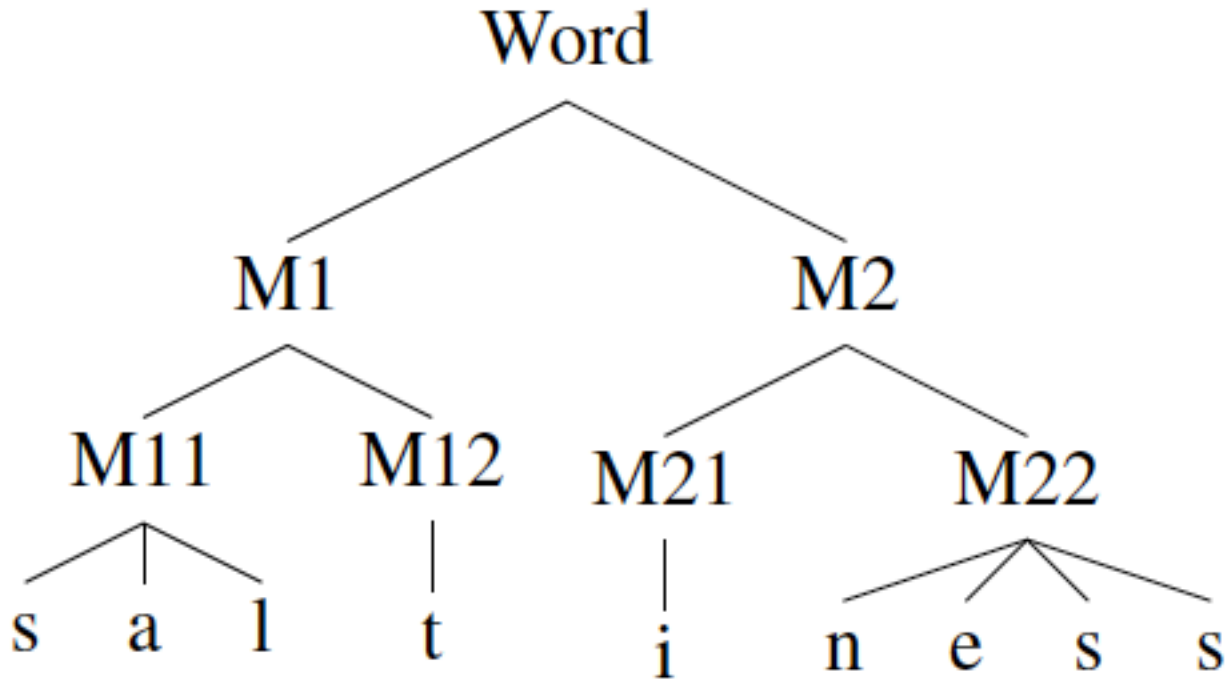


Figure 1: The parse tree generated by the metagrammar of depth 2 for the word *saltiness*.

Sirts & Goldwater (2013)

Morphological Segmentations, POS, and Syntactic Trees

Snyder & Barzilay

Generative Stories

- we hypothesize latent variables through which data are generated
- define “generative story” that describes how latent variables are generated, then how data is generated using latent variables

Expectation Maximization (EM)

$$\max_\theta \prod_i \sum_z p(x^{(i)}, z \mid \theta)$$

- EM is an algorithmic template that finds a local maximum of the marginal likelihood of the observed data

Expectation Maximization (EM)

$$\max_{\theta} \prod_i \sum_z p(x^{(i)}, z \mid \theta)$$

- working instead with the log-likelihood:

$$\sum_i \log \sum_z p(x^{(i)}, z \mid \theta)$$

$$= \sum_i \log \sum_z q_i(z) \frac{p(x^{(i)}, z \mid \theta)}{q_i(z)}$$

- where q_i is some distribution over values for z

Expectation Maximization (EM)

$$\max_\theta \prod_i \sum_z p(x^{(i)}, z \mid \theta)$$

- working instead with the log-likelihood:

$$\sum_i \log \sum_z p(x^{(i)}, z \mid \theta)$$

$$= \sum_i \log \sum_z q_i(z) \frac{p(x^{(i)}, z \mid \theta)}{q_i(z)}$$

via Jensen's inequality $$\geq \sum_i \sum_z q_i(z) \log \frac{p(x^{(i)}, z \mid \theta)}{q_i(z)}$$

Expectation Maximization (EM)

$$\max_{\theta} \prod_i \sum_z p(x^{(i)}, z \mid \theta)$$

- maximize lower bound of the log-likelihood:

$$\sum_i \sum_z q_i(z) \log \frac{p(x^{(i)}, z \mid \theta)}{q_i(z)}$$

- alternate between optimizing wrt *q* and theta

EM

- "E" step:
  - compute posteriors over latent variables:

$$\text{for each } i,\ q_i(z) = p(z \mid x^{(i)}, \theta)$$

EM

- "E" step:
  - compute posteriors over latent variables:

$$\text{for each } i,\ q_i(z) = p(z \mid x^{(i)}, \theta)$$

- "M" step:
  - update parameters given posteriors:

$$\theta = \operatorname*{argmax}_{\theta'} \sum_i \sum_z q_i(z) \log \frac{p(x^{(i)}, z \mid \theta')}{q_i(z)}$$

EM for Structured Prediction

- to compute posteriors, we need to sum over all output structures

EM Today

- today we don’t always need to do the alternating steps of EM
- just like summing inference for structured prediction, we can implement the summing algorithm using computation graphs, then use autodifferentiation
- parameterize categorical distributions using a “softmax parameterization” (i.e., do optimization in the logits, not probabilities)