Vector Embeddings & Training LR Models

Instructor: Lara J. Martin (she/they)

TA: Omkar Kulkarni (he)

https://laramartin.net/NLP-class/

Slides modified from Dr. Frank Ferraro

Learning Objectives

Calculate the distance between vector embeddings

Recognize popular vector embeddings

Define an objective for LR modeling

Visualize the learning process for maxent models

Distinguish between discriminatively- and generatively-trained maxent models

Review:
Dense Embeddings: Key Ideas

Vector embeddings can be used for phrases, paragraphs, or even whole documents!

1. Acquire basic contextual statistics (often counts) for each word type v

2. Extract a real-valued vector e_v for each word v from those statistics

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

3. Use the vectors to represent each word in later tasks

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

Review: Three Common Kinds of Embedding Models

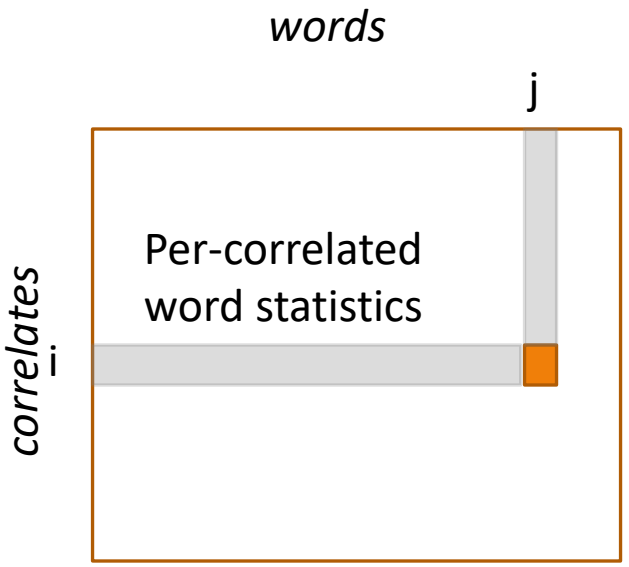
1. Co-occurrence matrices
2. Matrix Factorization: Singular value decomposition/Latent Semantic Analysis, Topic Models
3. Neural-network-inspired models (skip-grams, CBOW)

Review: 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)
- …



Assumption: Two words are similar if their vectors are similar

Review: 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)

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

probability that word x occurs

probability that word y occurs

Review: Word2Vec

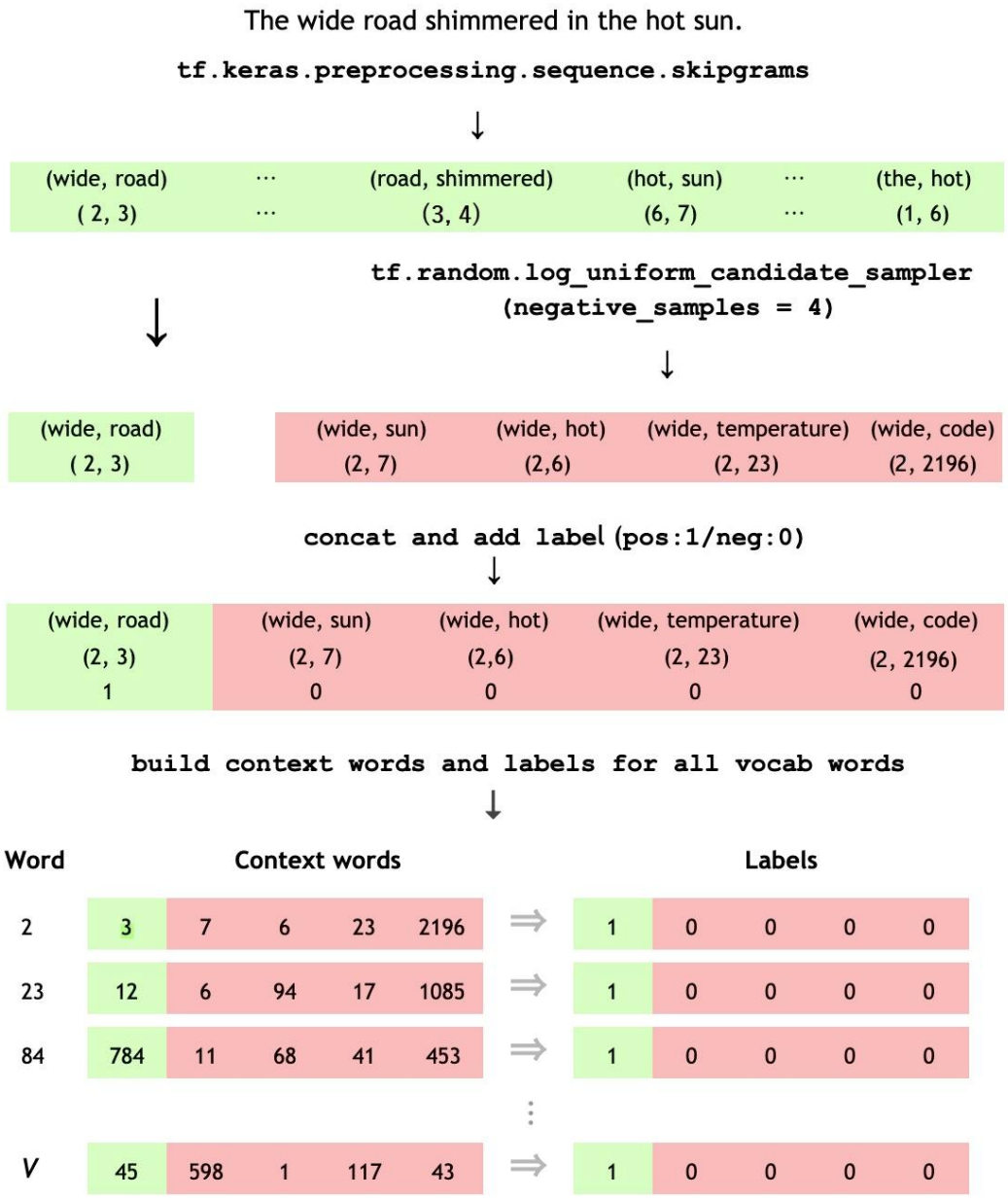
context (↓)-word (→) count matrix

	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

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

Example (Tensorflow)



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

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:

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

FastText:

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

FastText Details

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

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

decompose

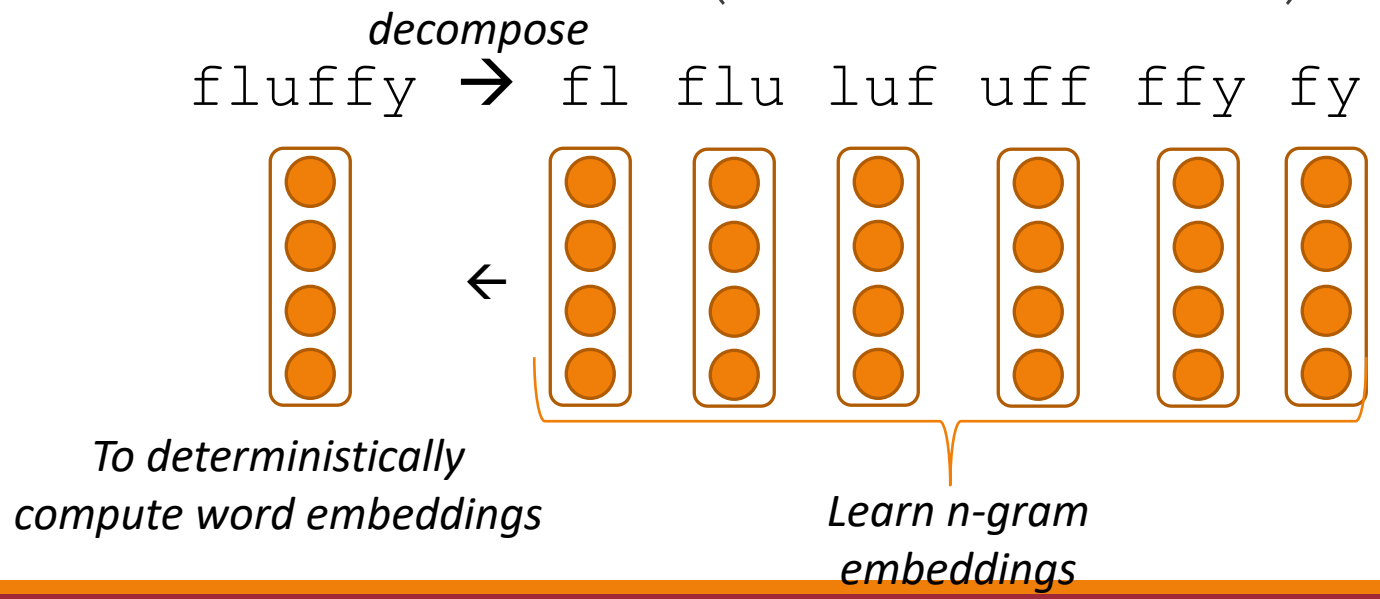
`fluffy` → `fl flu luf uff ffy fy`

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

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




123
SESAME STREET

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):

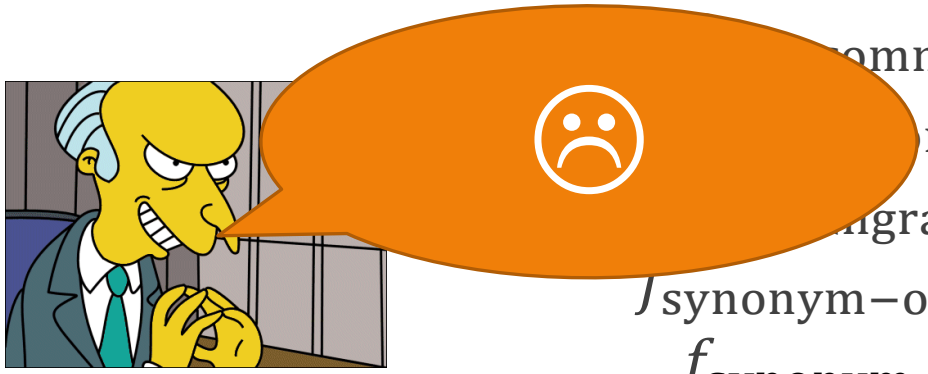
- ELMo: "Deep contextualized word representations" Peters et al. (2018; NAACL)
- https://allennlp.org/elmo
- BERT: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" Devlin et al. (2019; NAACL)
  - https://github.com/google-research/bert

Evaluating Vector Embeddings

Case Study: Maxent Plagiarism Detector (Feature Example)

Given two documents x_1, x_2, predict y = 1 (plagiarized) or y = 0 (not plagiarized)

Intuition: documents are more likely to be plagiarized if they have words in common



$f_{\text{ommon-word,Plag.}}(x_1, x_2) = ???$

$f_{\text{rd v>,Plag.}}(x_1, x_2) = ???$

$f_{\text{ngram Z>,Plag.}}(x_1, x_2) = ???$

$f_{\text{synonym-of-<word v>,Plag.}}(x_1, x_2) = ???$

$f_{\text{synonym-of-<ngram Z>,Plag.}}(x_1, x_2) =$

get_similarity_with_embeddings()

Image: http://3.bp.blogspot.com/_MWJDzvrnu7Y/TTcXpCWZerI/AAAAAAAAWY/zxu1GwrIBz0/s1600/excellent-mr-burns.gif

Evaluating Similarity

Extrinsic (task-based, end-to-end) Evaluation:

- Question Answering
- Spell Checking
- Essay grading

Common Evaluation: Correlation between similarity ratings

Input: list of N word pairs $\{(x_1, y_1), \ldots, (x_N, y_N)\}$

- Each word pair (x_i, y_i) has a human-provided similarity score h_i

Use your embeddings to compute an embedding similarity score $s_i = \text{sim}(x_i, y_i)$

Compute the correlation between human and computed similarities

$$\rho = \text{Corr}((h_1, \ldots, h_N), (s_1, \ldots, s_N))$$

Wordsim353: 353 noun pairs rated 0-10

Cosine: Measuring Similarity

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

Dot product or inner product from linear algebra

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

- 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

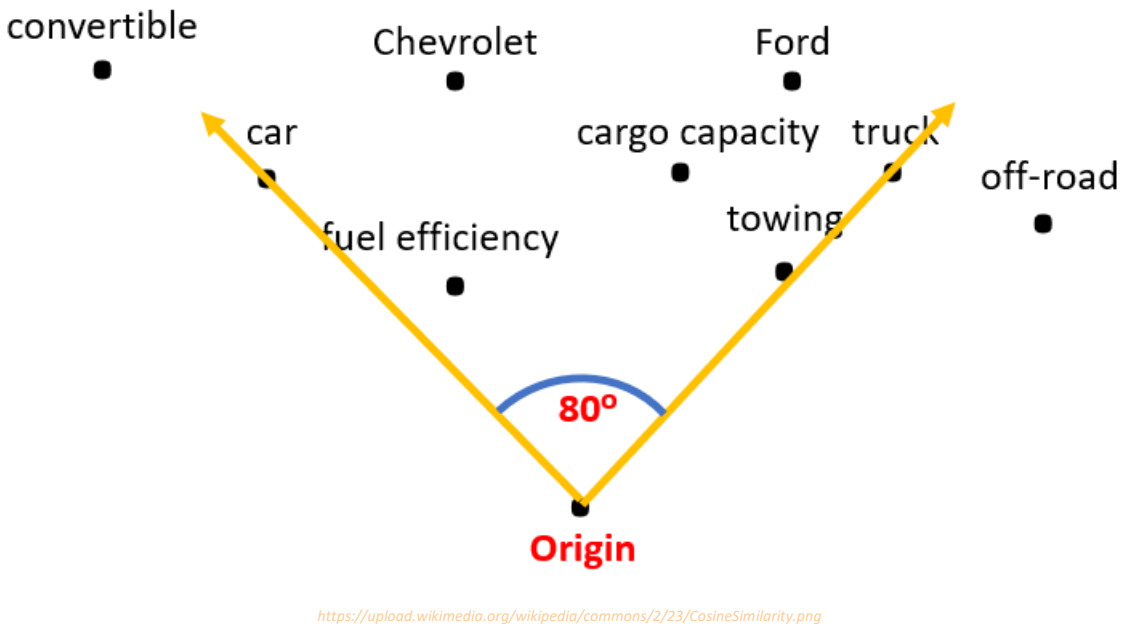
Divide the dot product by the length of the two vectors

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

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

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

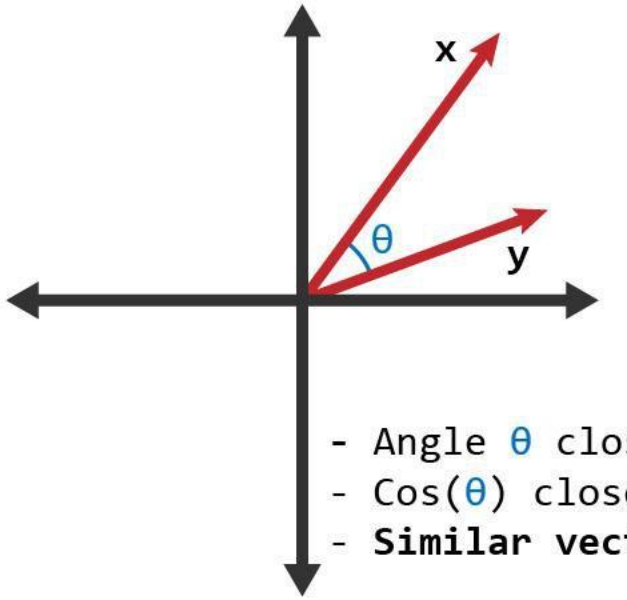
	Dim. 1	Dim. 2	Dim. 3
apricot	2	0	0
digital	0	1	2
information	1	6	1

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

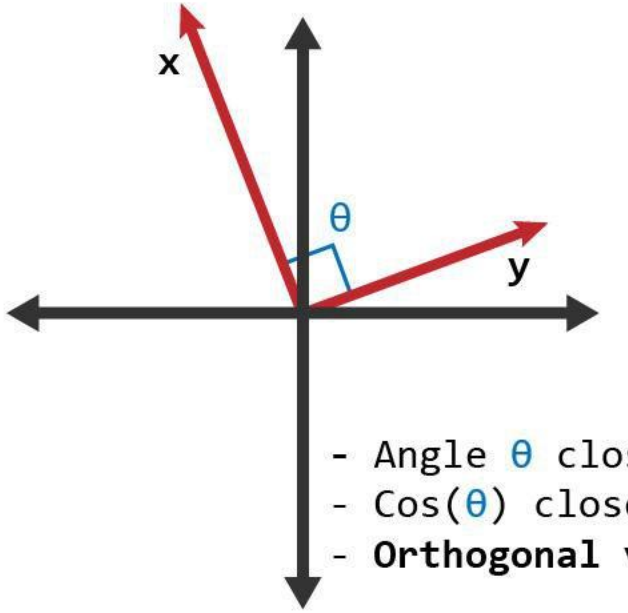
cosine(digital,information) = $\frac{0+6+2}{\sqrt{0+1+4}\sqrt{1+36+1}} = 0.5804$

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

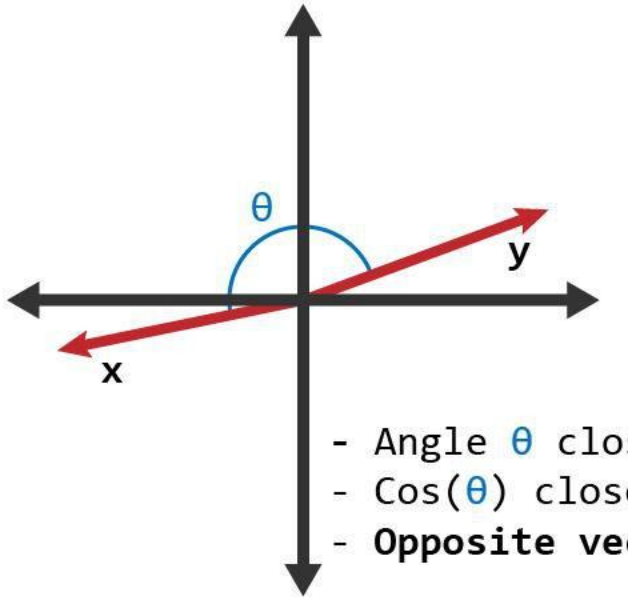
Cosine Similarity Range



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



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



- Angle θ close to 180
- Cos(θ) close to -1
- **Opposite vectors**

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

Other Similarity Measures

$$\text{sim}_{\text{cosine}}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

$$\text{sim}_{\text{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} \max(v_i,w_i)}$$

$$\text{sim}_{\text{Dice}}(\vec{v},\vec{w}) = \frac{2\times\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} (v_i+w_i)}$$

$$\text{sim}_{\text{JS}}(\vec{v}||\vec{w}) = D(\vec{v}|\frac{\vec{v}+\vec{w}}{2}) + D(\vec{w}|\frac{\vec{v}+\vec{w}}{2})$$

Back to LR Classifiers

HOW ARE THEY TRAINED?

Outline

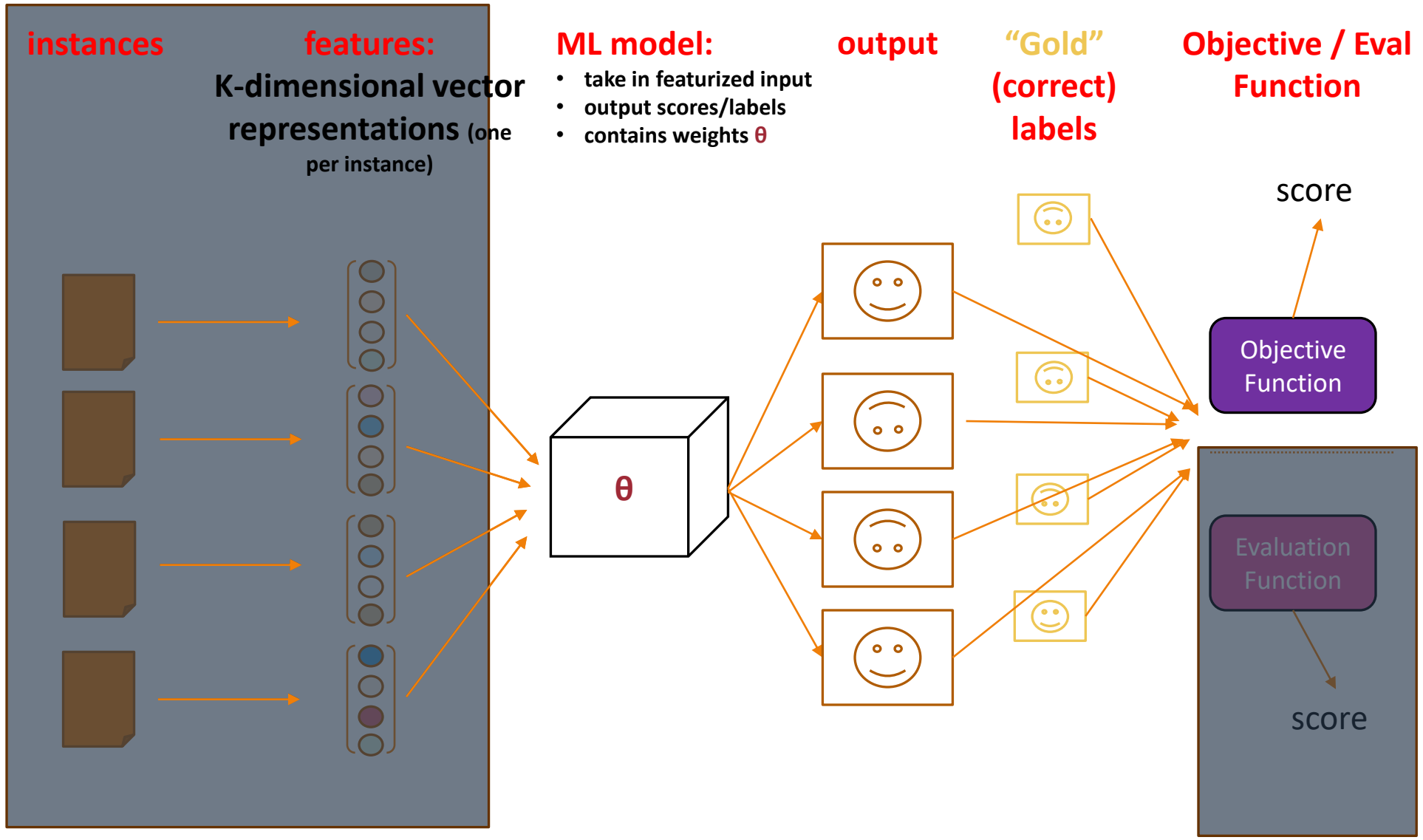
Maximum Entropy classifiers

- Defining the model: Discriminatively
- Defining the objective
- Learning: Optimizing the objective
- Defining the model: Generatively

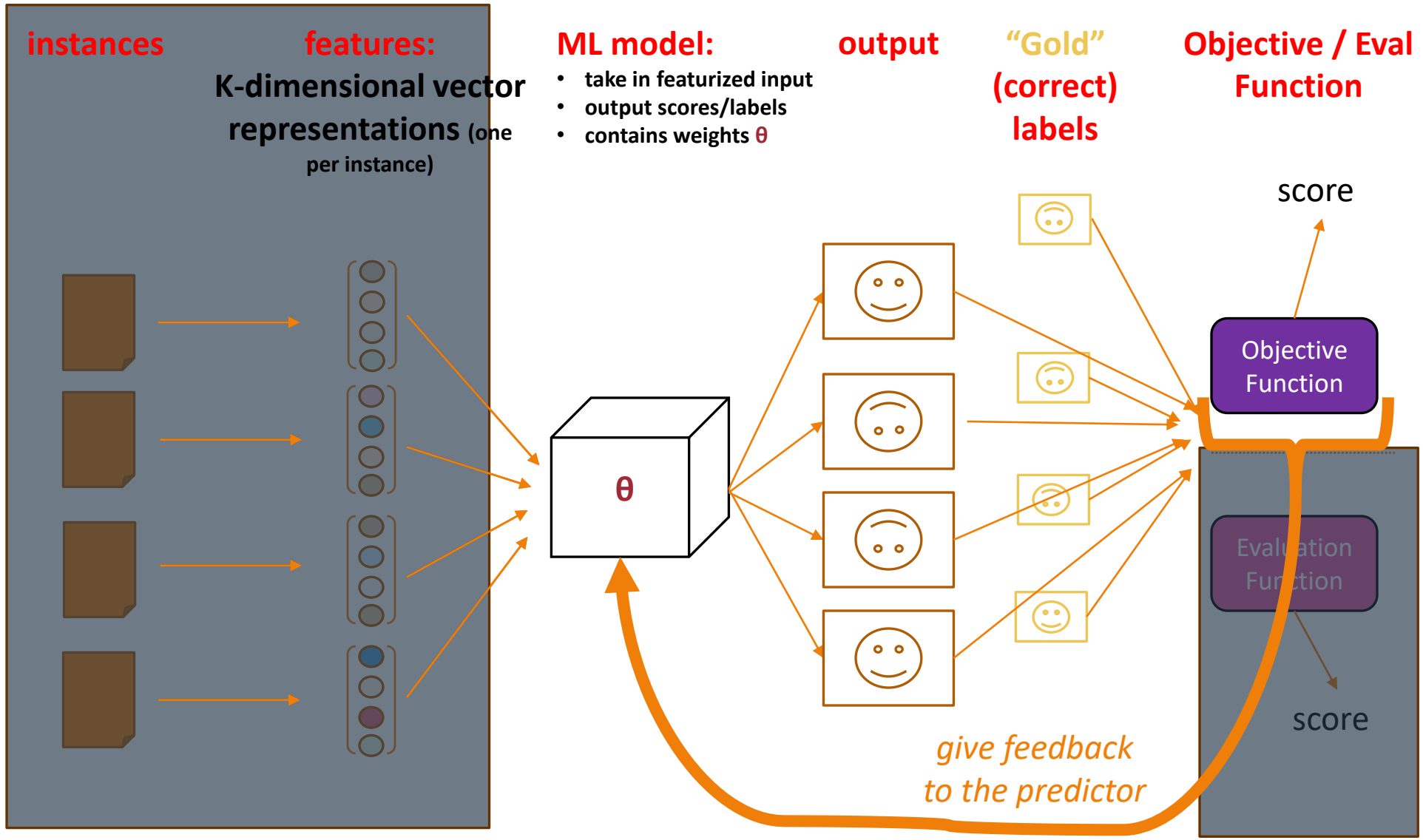
$$p_\theta(y \mid x)$$ probabilistic model

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

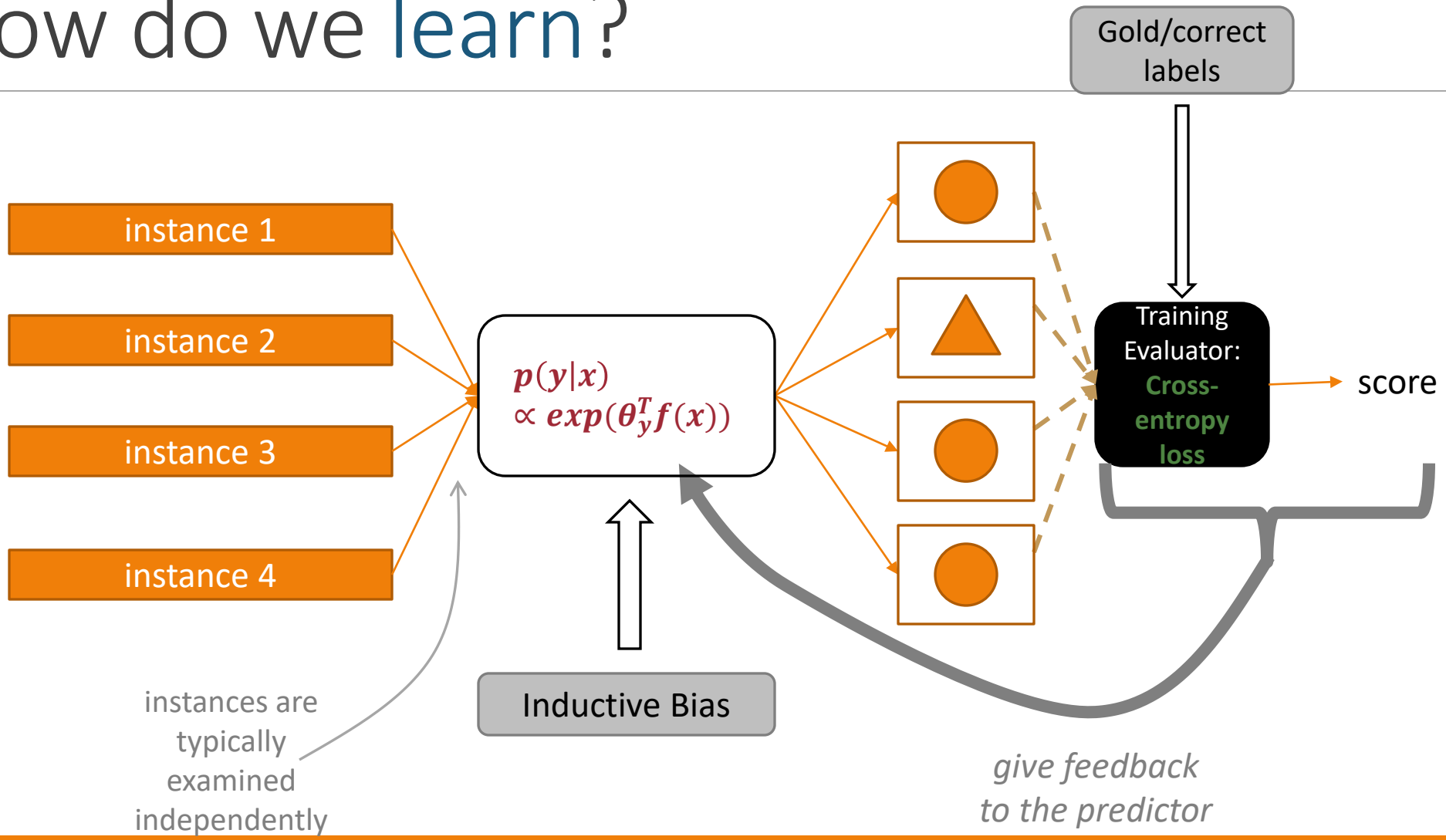
Defining the Objective



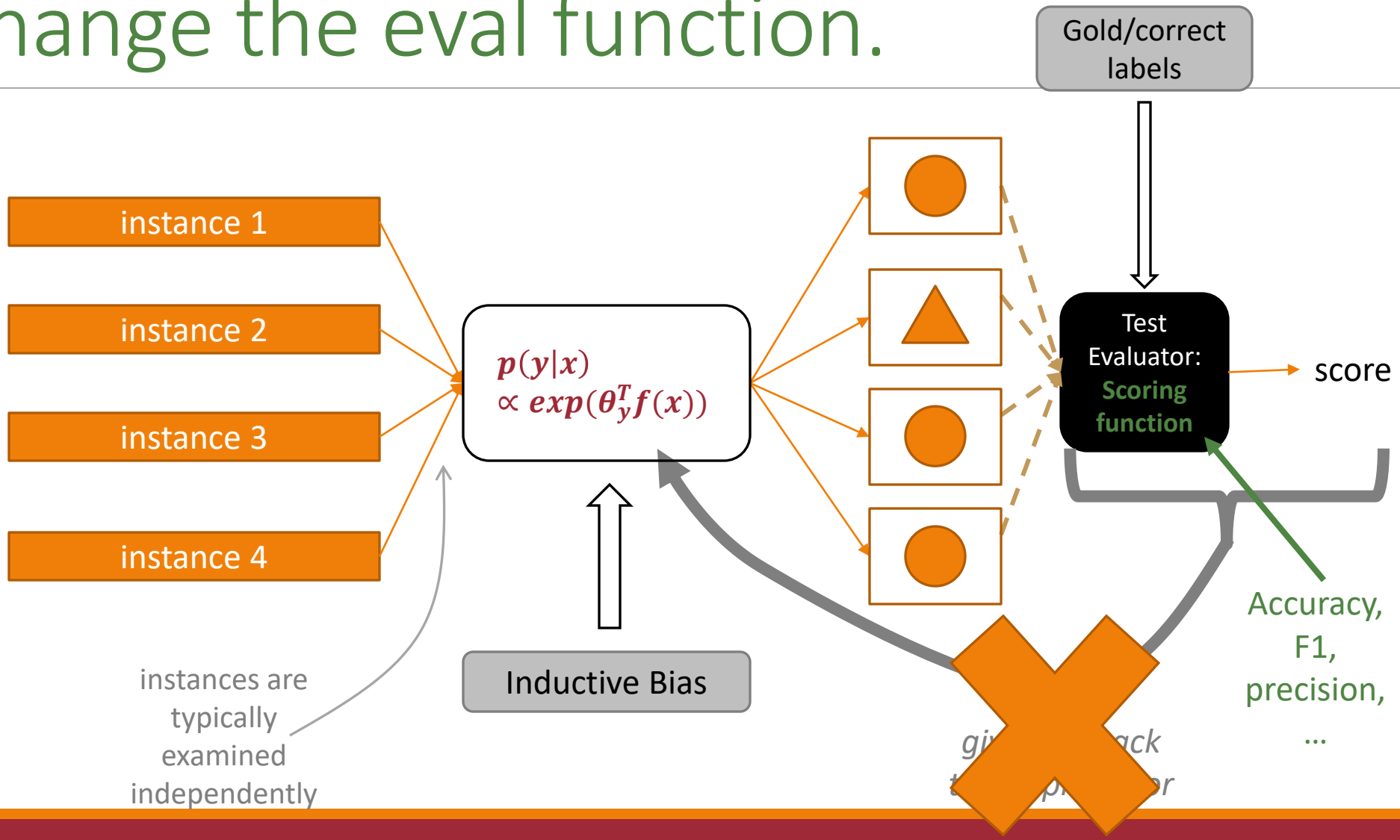
Defining the Objective



How do we learn?



How do we evaluate (or use)? Change the eval function.



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?

Our goal probability

Our maxent equation

$$\prod_i p_\theta(y_i|x_i) \propto \prod_i \exp(\theta_{y_i}^T f(x_i))$$

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?

$$\prod_i p_\theta(y_i|x_i)$$

Maximize Log-Likelihood

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

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

Inverse of exp
log(exp(x)) = x

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

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)

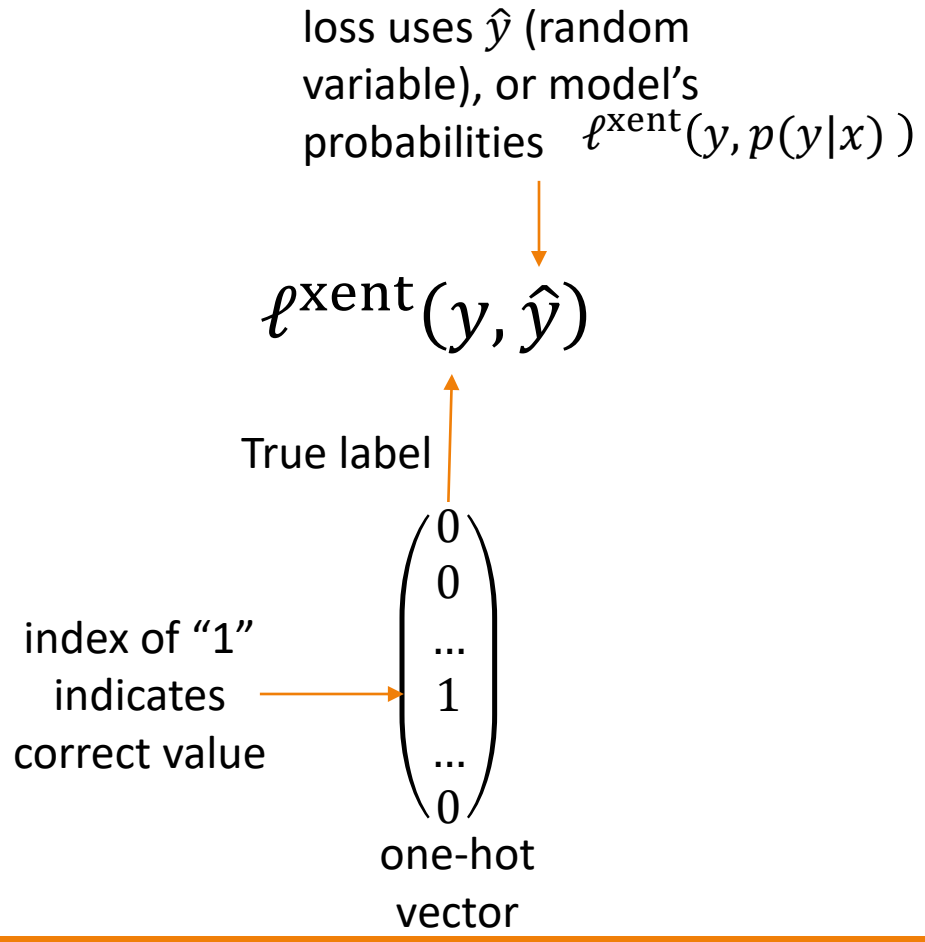
Maximize Log-Likelihood

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

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

$$= F(\theta)$$

Equivalent Version 2: *Minimize* Cross Entropy Loss



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

SLP3, ch. 5, pg. 12

Equivalent Version 2: *Minimize* Cross Entropy Loss

loss uses $\hat{y}$ (random variable), or model's probabilities

$\ell^{\text{xent}}(y, p(y|x))$

$$\ell^{\text{xent}}(y, \hat{y}) = -\sum_k y[k] \cdot \log p(y = k|x)$$

True label

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

Classification Log-likelihood (max) ≅ Cross Entropy Loss (min)

Log Likelihood

objective is concave

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

objective is convex

CROSSENTROPYLOSS

CLASS `torch.nn.CrossEntropyLoss(weight=None, size_average=None, ignore_index=-100, reduce=None, reduction='mean')` [SOURCE]

This criterion combines `LogSoftmax` and `NLLLoss` in one single class.

It is useful when training a classification problem with *C* classes. If provided, the optional argument `weight` should be a 1D *Tensor* assigning weight to each of the classes. This is particularly useful when you have an unbalanced training set.

The *input* is expected to contain raw, unnormalized scores for each class.

input has to be a Tensor of size either $(minibatch, C)$ or $(minibatch, C, d_1, d_2, ..., d_K)$ with $K \geq 1$ for the *K*-dimensional case (described later).

This criterion expects a class index in the range $[0, C-1]$ as the *target* for each value of a 1D tensor of size *minibatch*; if *ignore_index* is specified, this criterion also accepts this class index (this index may not necessarily be in the class range).

The loss can be described as:

$$\text{loss}(x, class) = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) = -x[class] + \log\left(\sum_j \exp(x[j])\right)$$

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

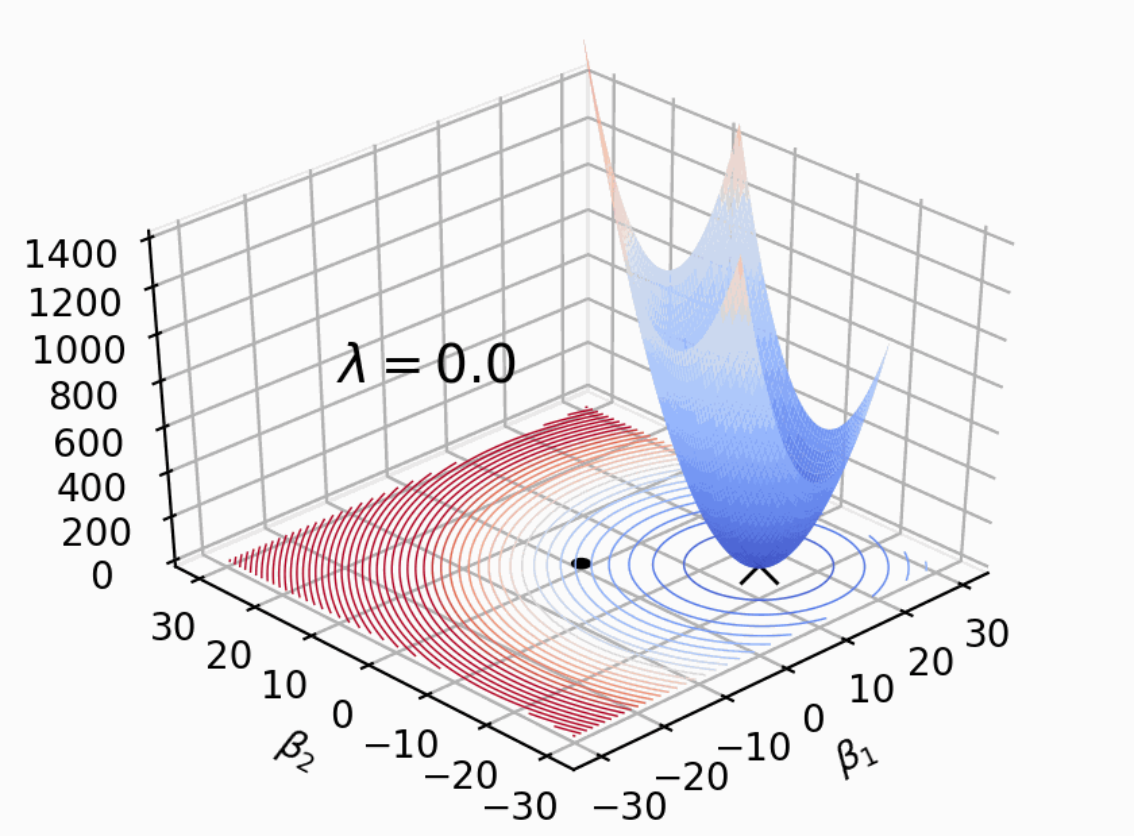
$$F(\theta) = \left(\sum_i \theta_{y_i}^T f(x_i) - \log Z(x_i) \right) - R(\theta)$$

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

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

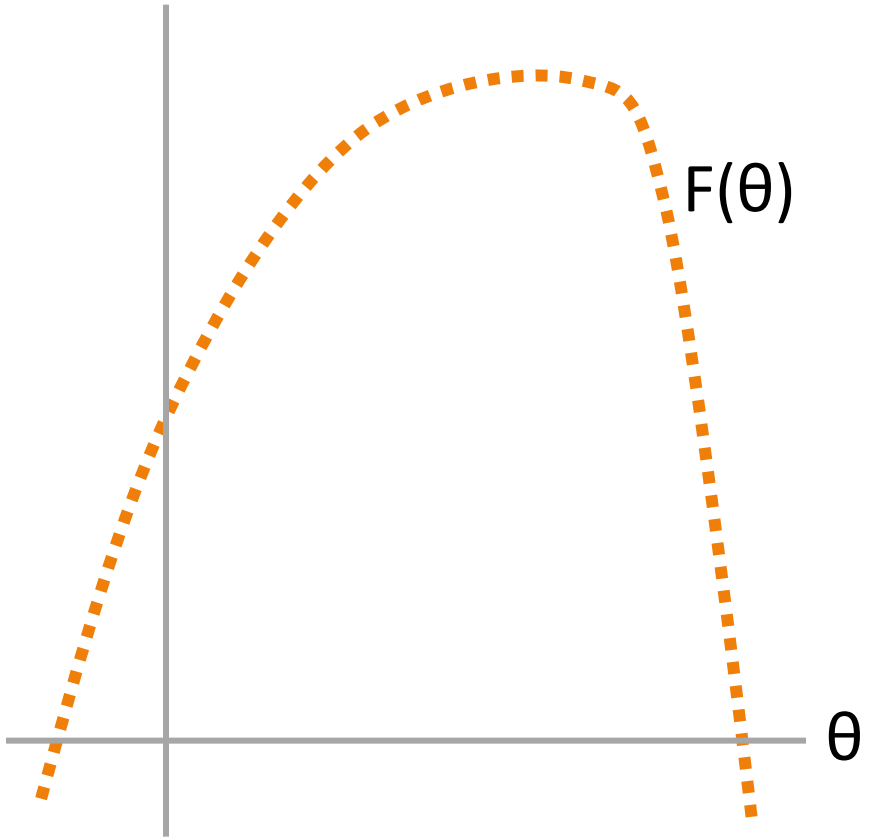
Outline

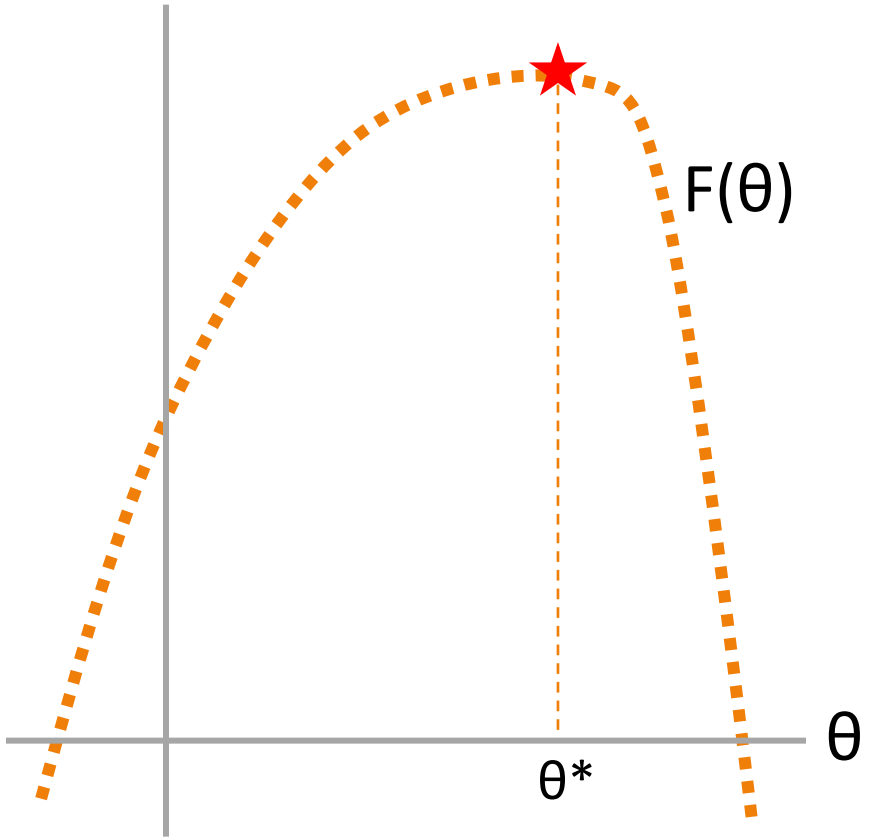
Maximum Entropy classifiers

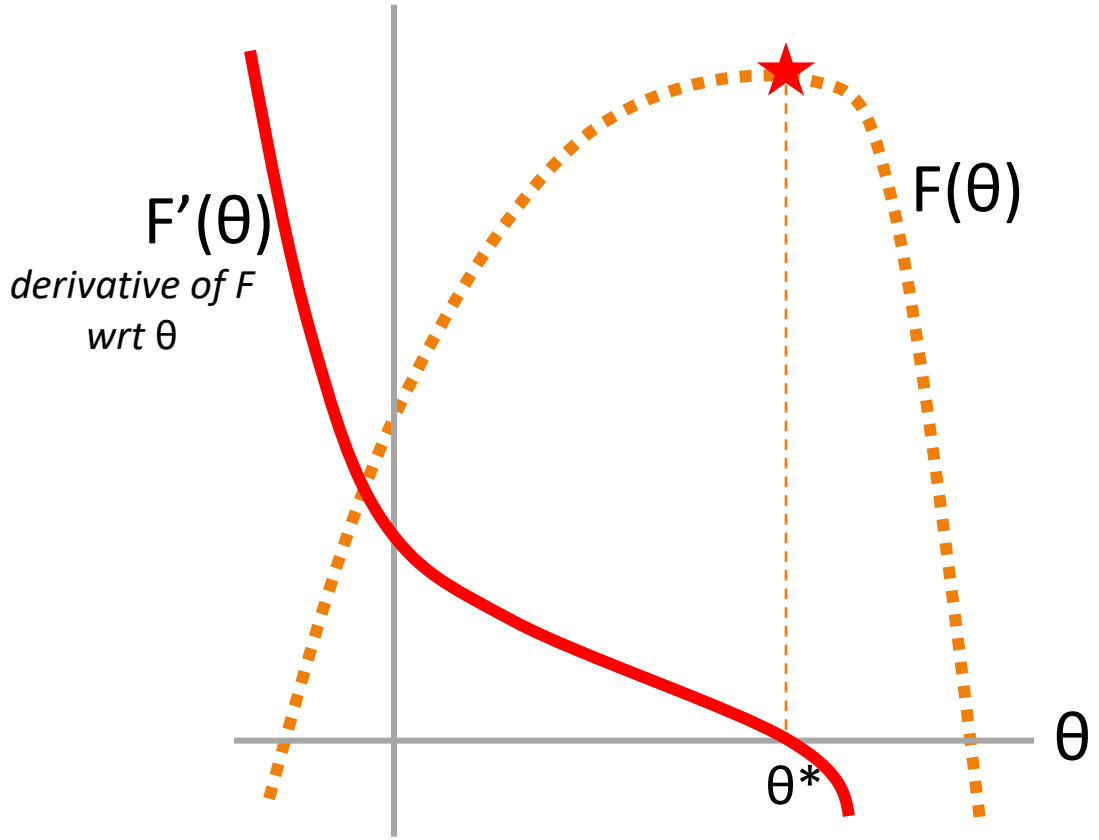
- Defining the model: Discriminatively
- Defining the objective
- Learning: Optimizing the objective
- Defining the model: Generatively

How will we optimize F(θ)?

Calculus.


F(θ)
θ


F(θ)
θ
θ*


F’(θ)
derivative of F
wrt θ
F(θ)
θ
θ*

Example
(Best case, solve for roots of the derivative)

$$F(x) = -(x-2)^2$$

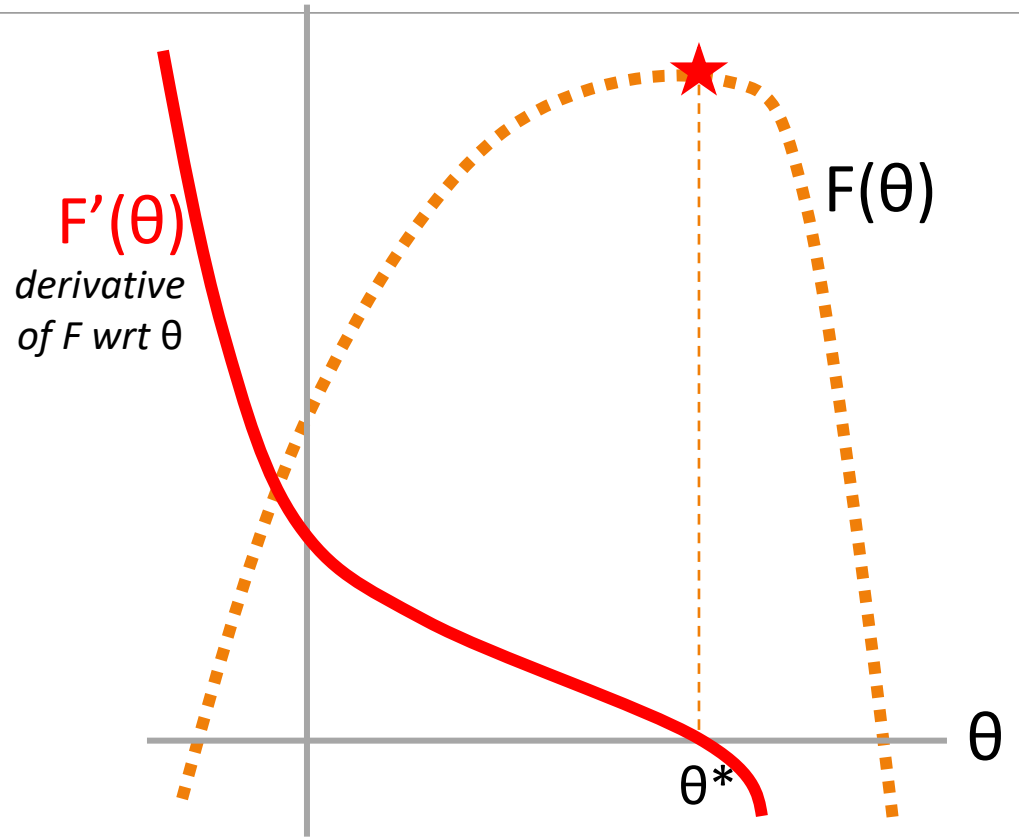
differentiate

$$F'(x) = -2x + 4$$

Solve F'(x) = 0

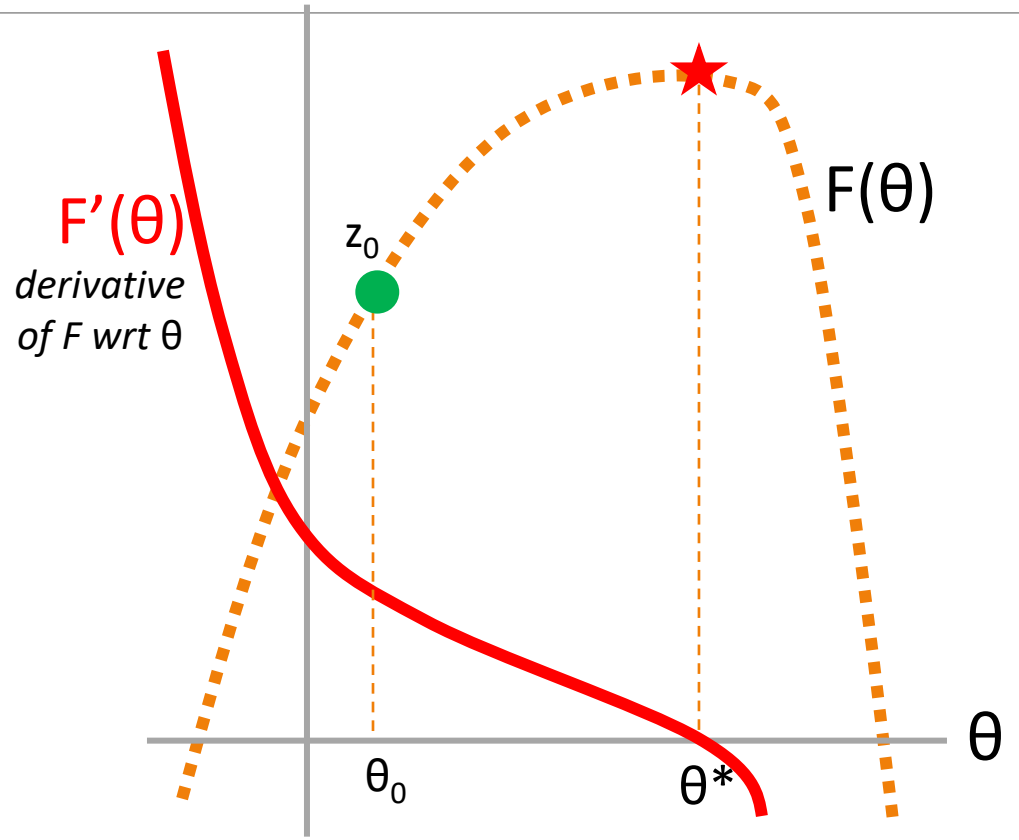
$$x = 2$$

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



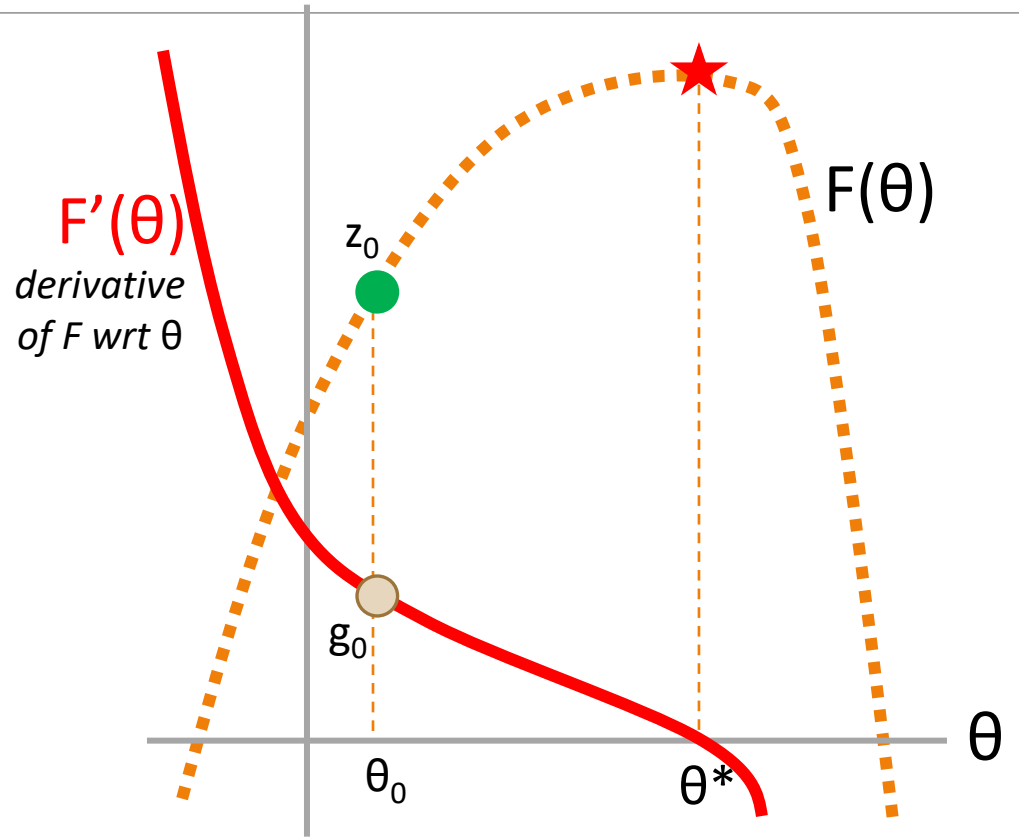
What if you can't find the roots? Follow the derivative

Set t = 0
Pick a starting value θ_t
Until converged:
1. Get value $z_t = F(\theta_t)$



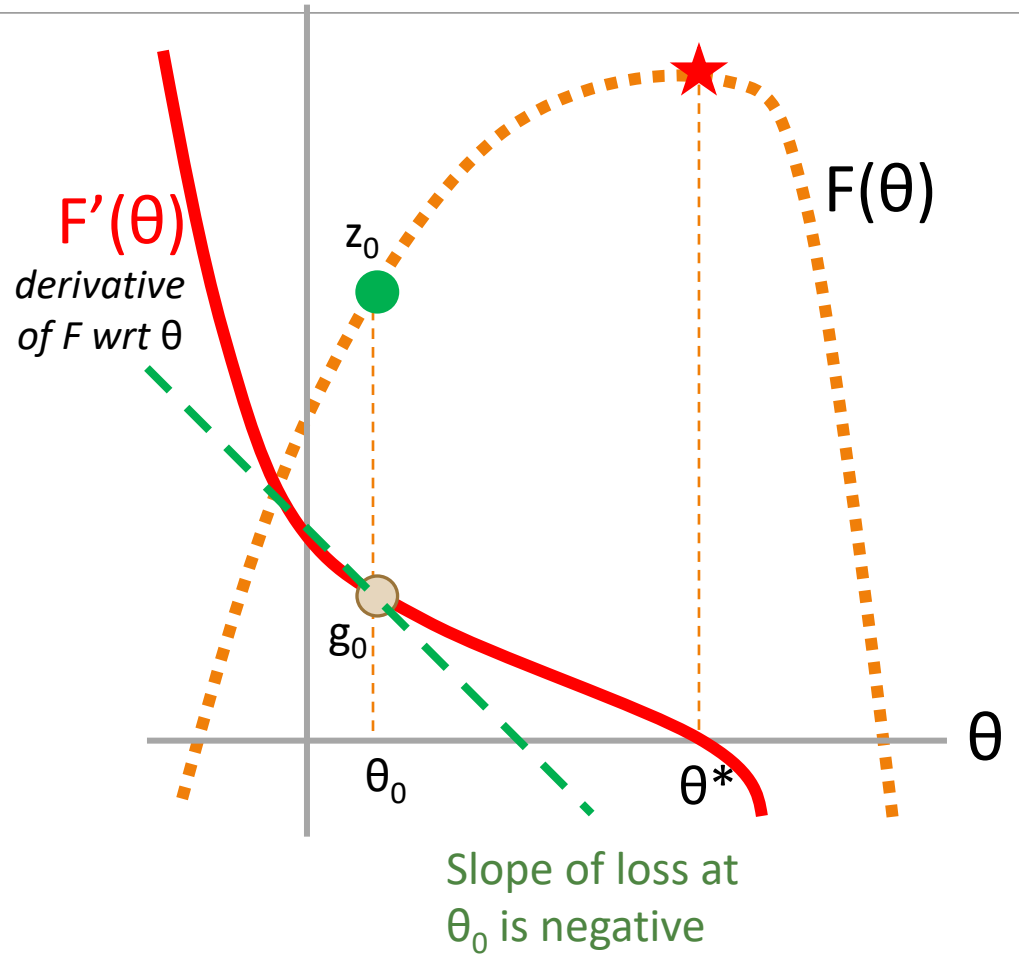
What if you can't find the roots? Follow the derivative

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

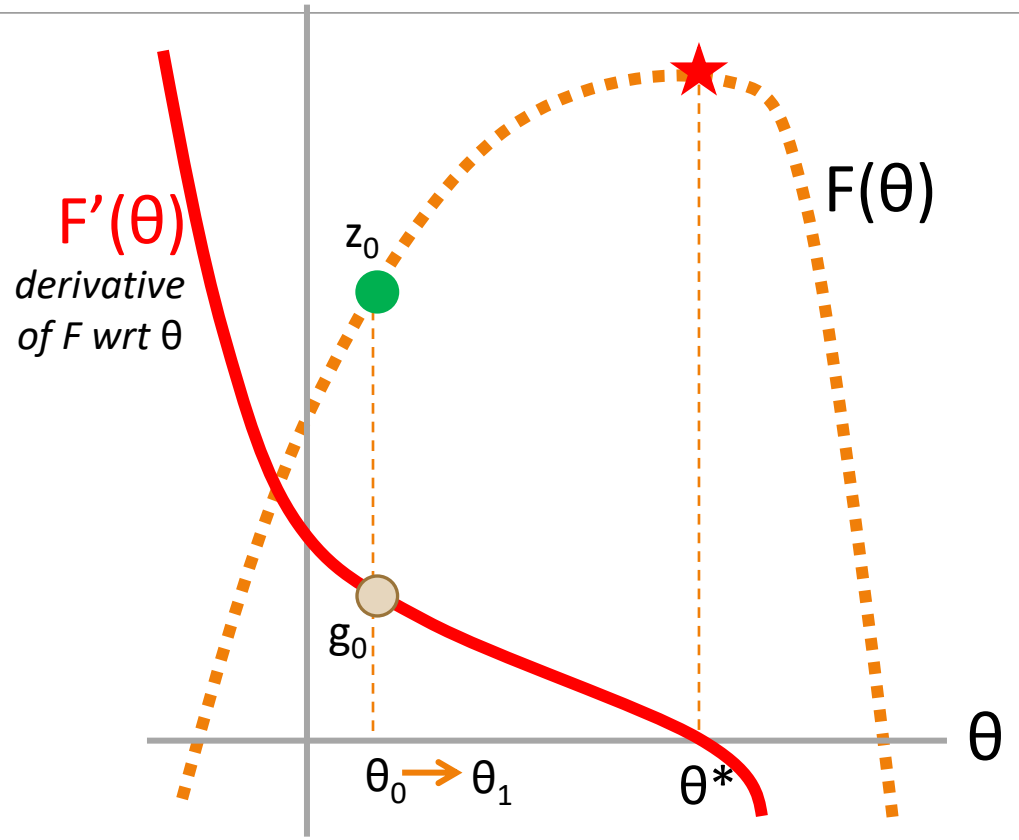
Set t = 0
Pick a starting value θ_t
Until converged:
1. Get value $z_t = F(\theta_t)$
2. Get derivative $g_t = F'(\theta_t)$



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

Set t = 0
Pick a starting value θ_t
Until converged:

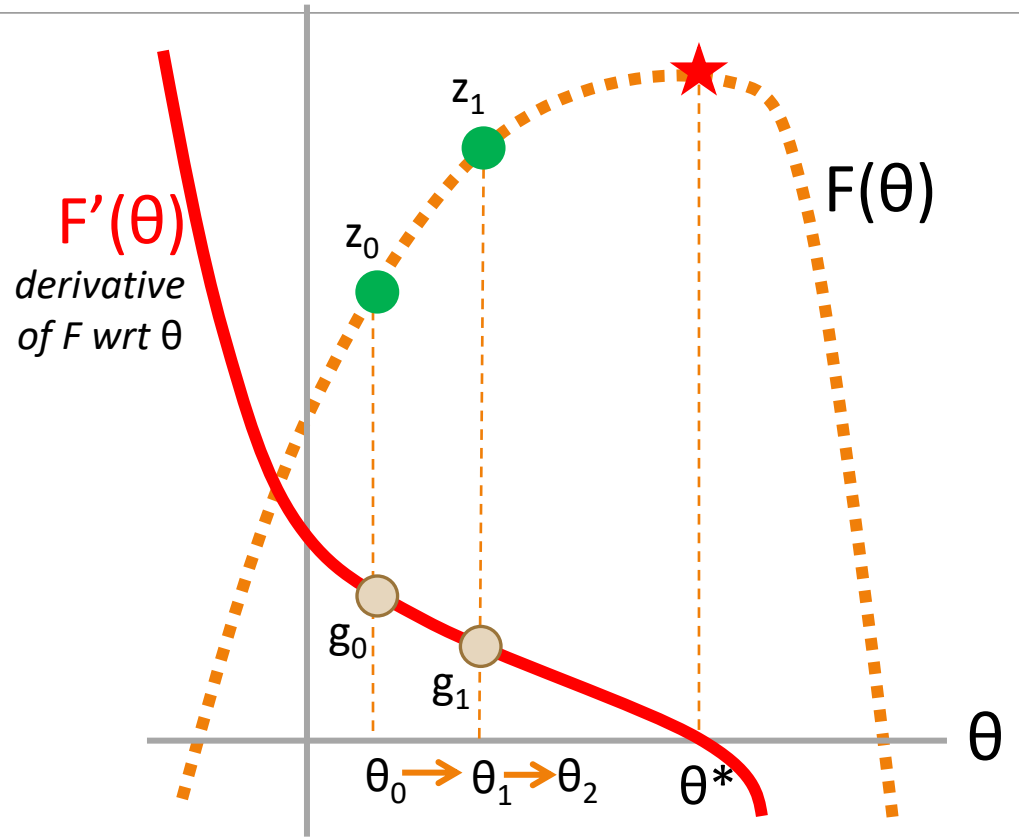
1. Get value $z_t = F(\theta_t)$
2. Get derivative $g_t = F'(\theta_t)$
3. Get scaling factor (learning rate) ρ_t
4. Set $\theta_{t+1} = \theta_t + \rho_t * g_t$
5. Set t += 1



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

Set t = 0
Pick a starting value θ_t
Until converged:

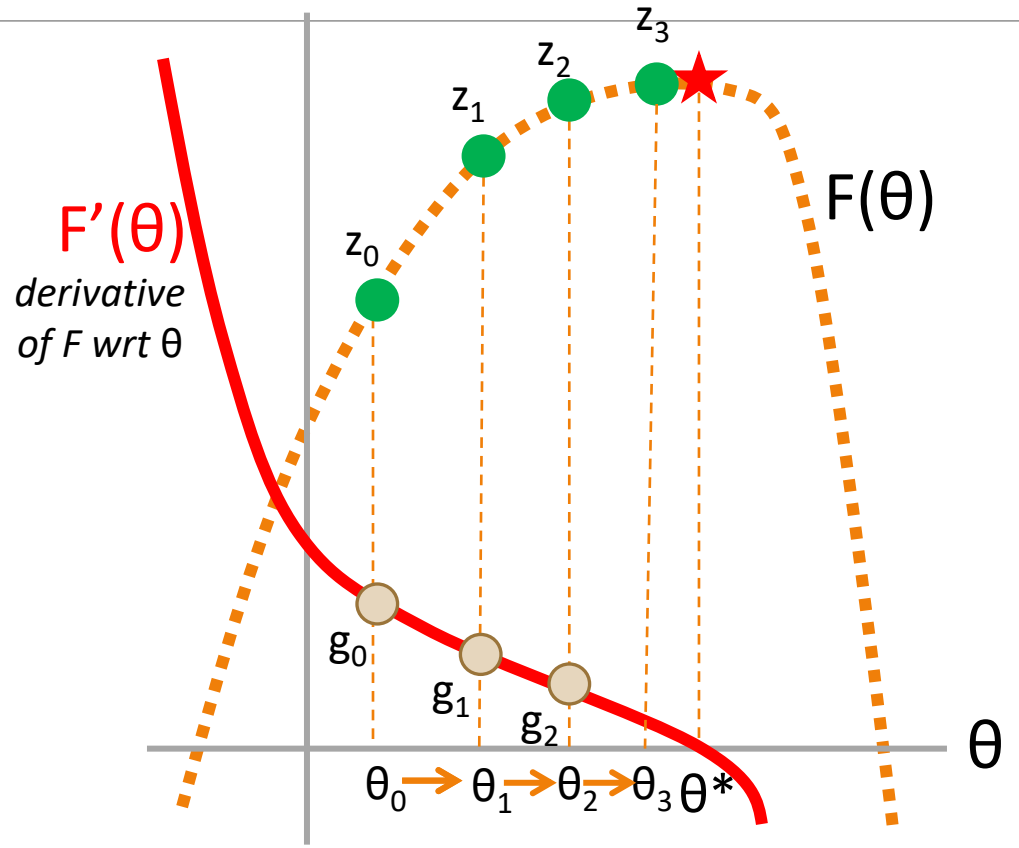
1. Get value $z_t = F(\theta_t)$
2. Get derivative $g_t = F'(\theta_t)$
3. Get scaling factor (learning rate) ρ_t
4. Set $\theta_{t+1} = \theta_t + \rho_t * g_t$
5. Set t += 1



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

Set t = 0
Pick a starting value θ_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) ρ_t
4. Set $\theta_{t+1} = \theta_t + \rho_t * g_t$
5. Set t += 1



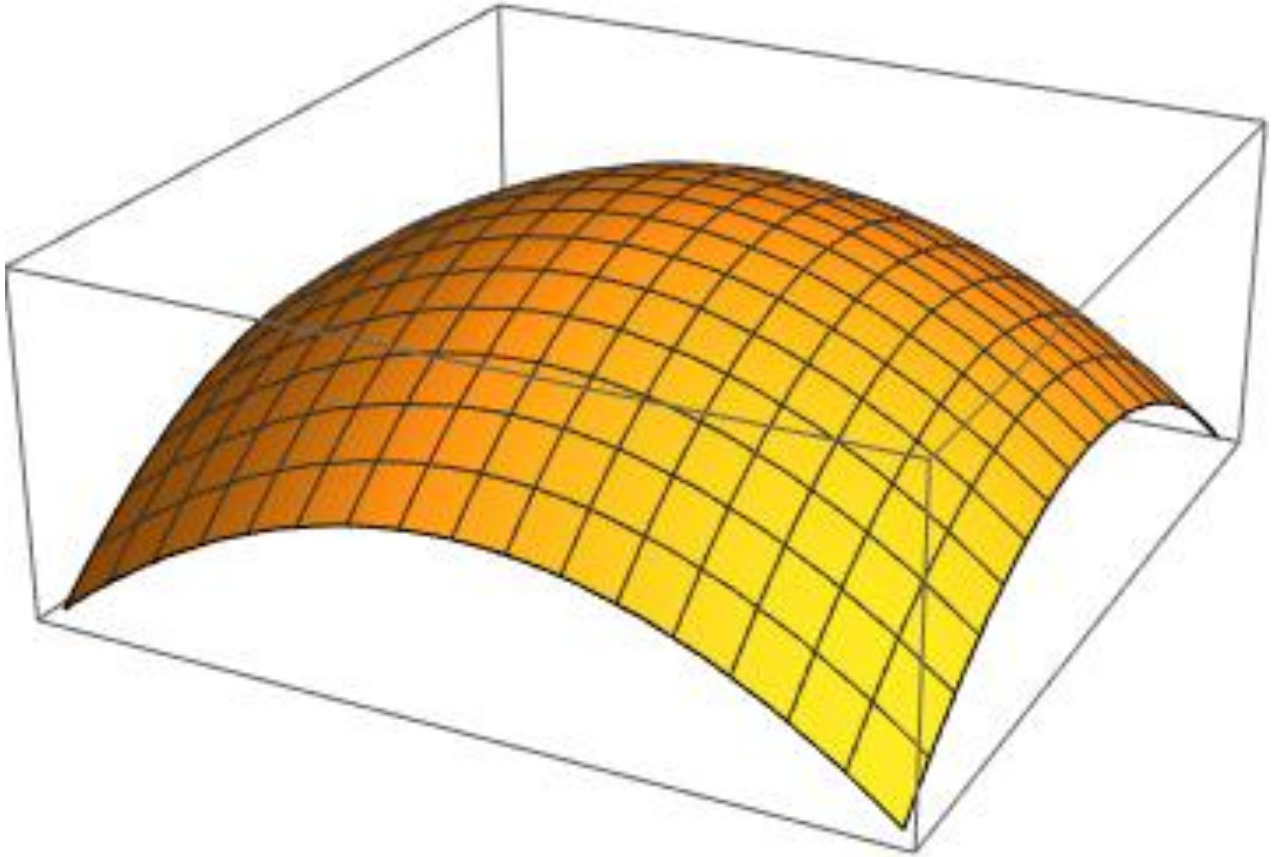
Gradient = Multi-variable derivative

K-dimensional input

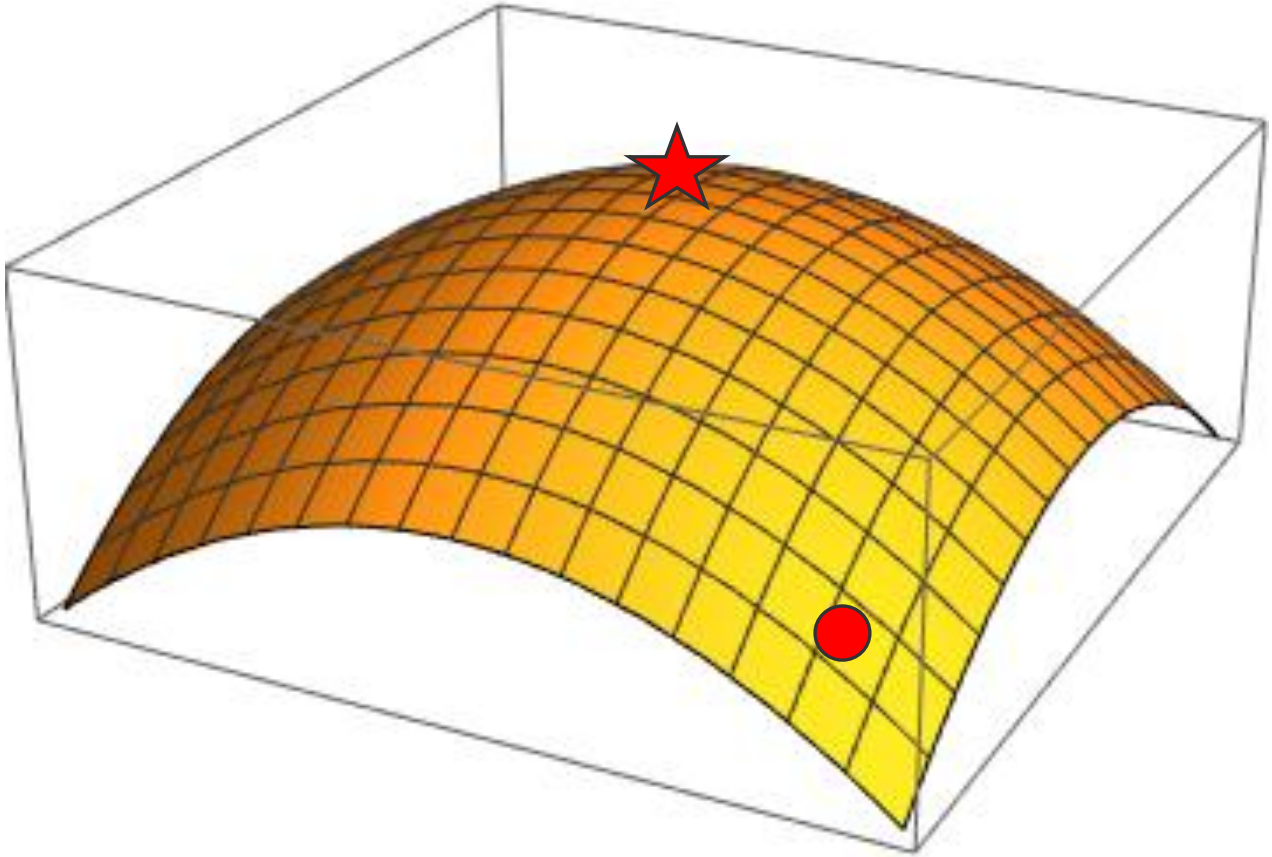
$$\nabla_\theta F(\theta) = \left(\frac{\partial F}{\partial \theta_1}, \frac{\partial F}{\partial \theta_2}, \ldots, \frac{\partial F}{\partial \theta_K} \right)$$

K-dimensional output

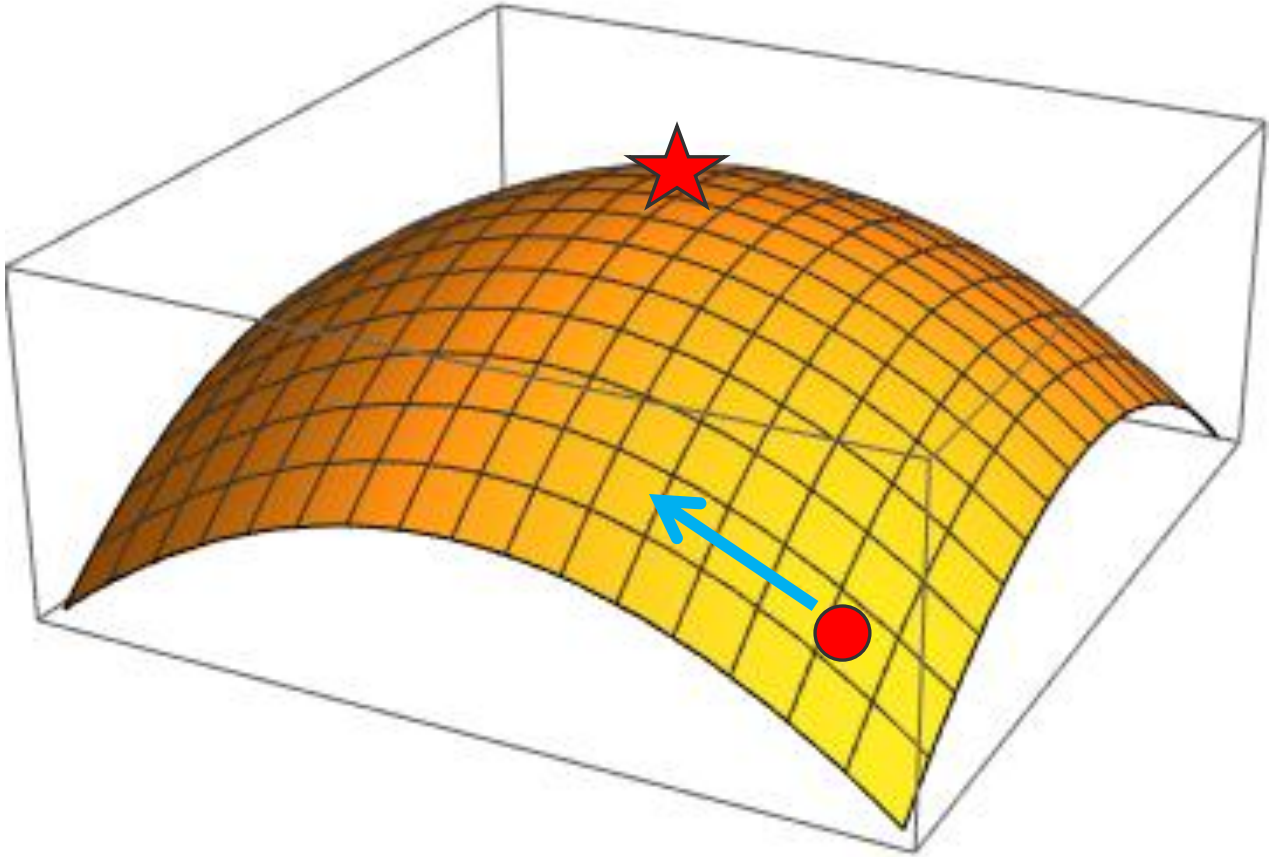
Gradient Ascent



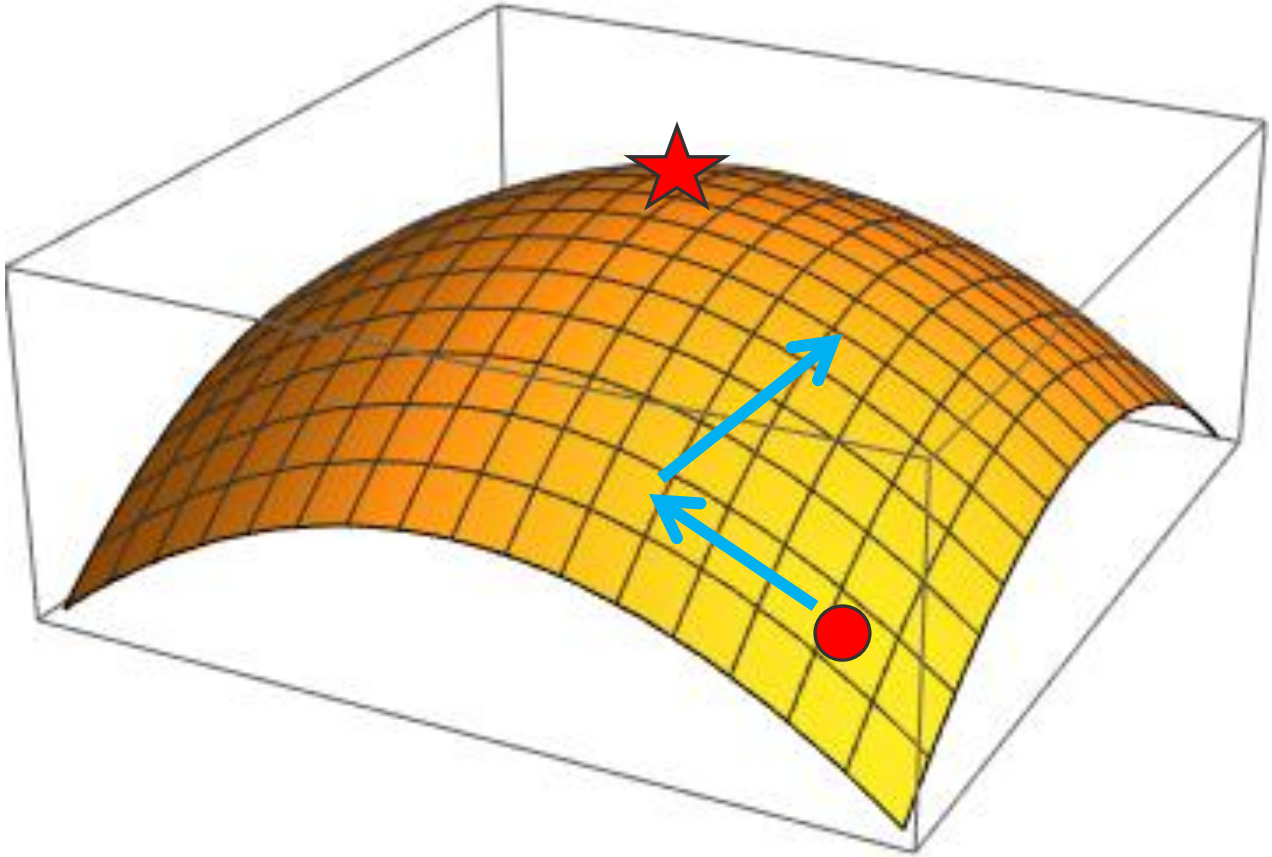
Gradient Ascent



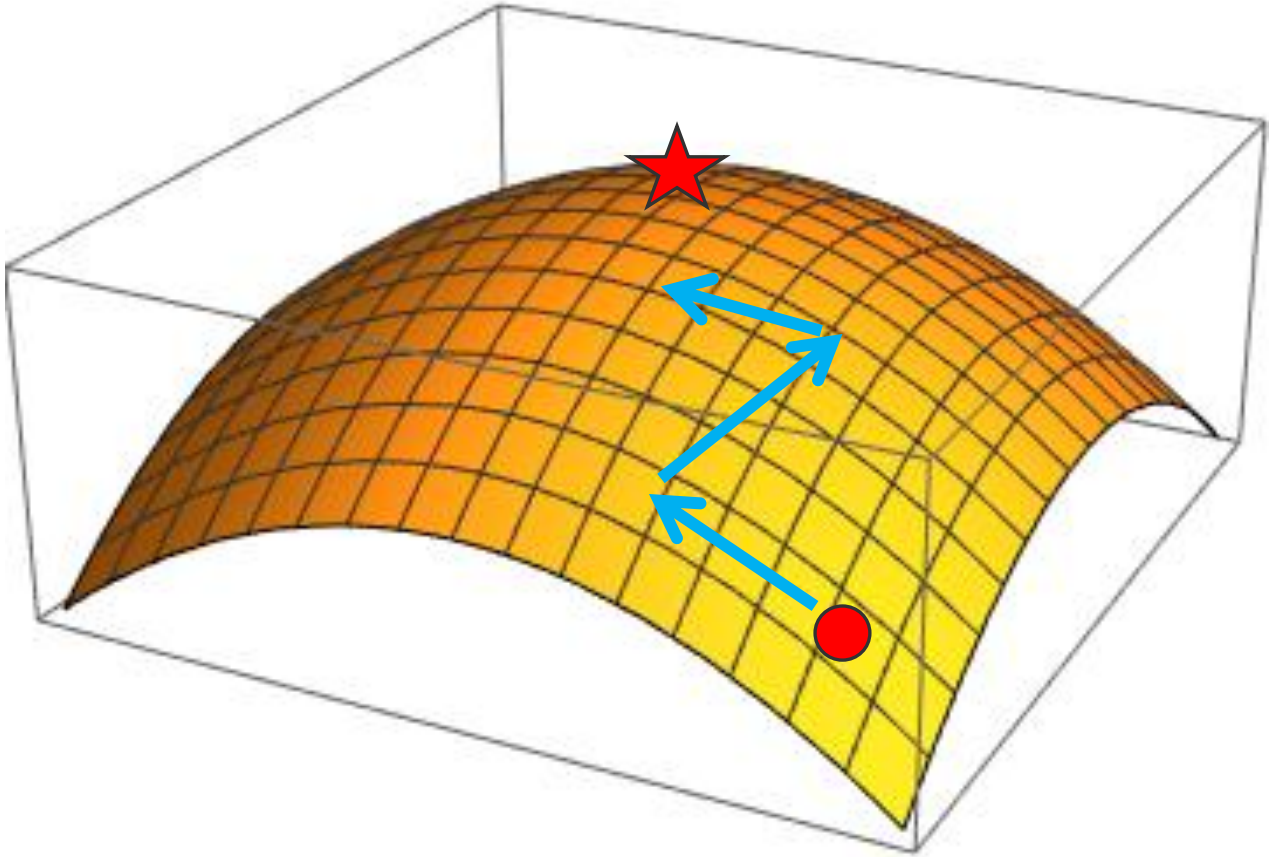
Gradient Ascent



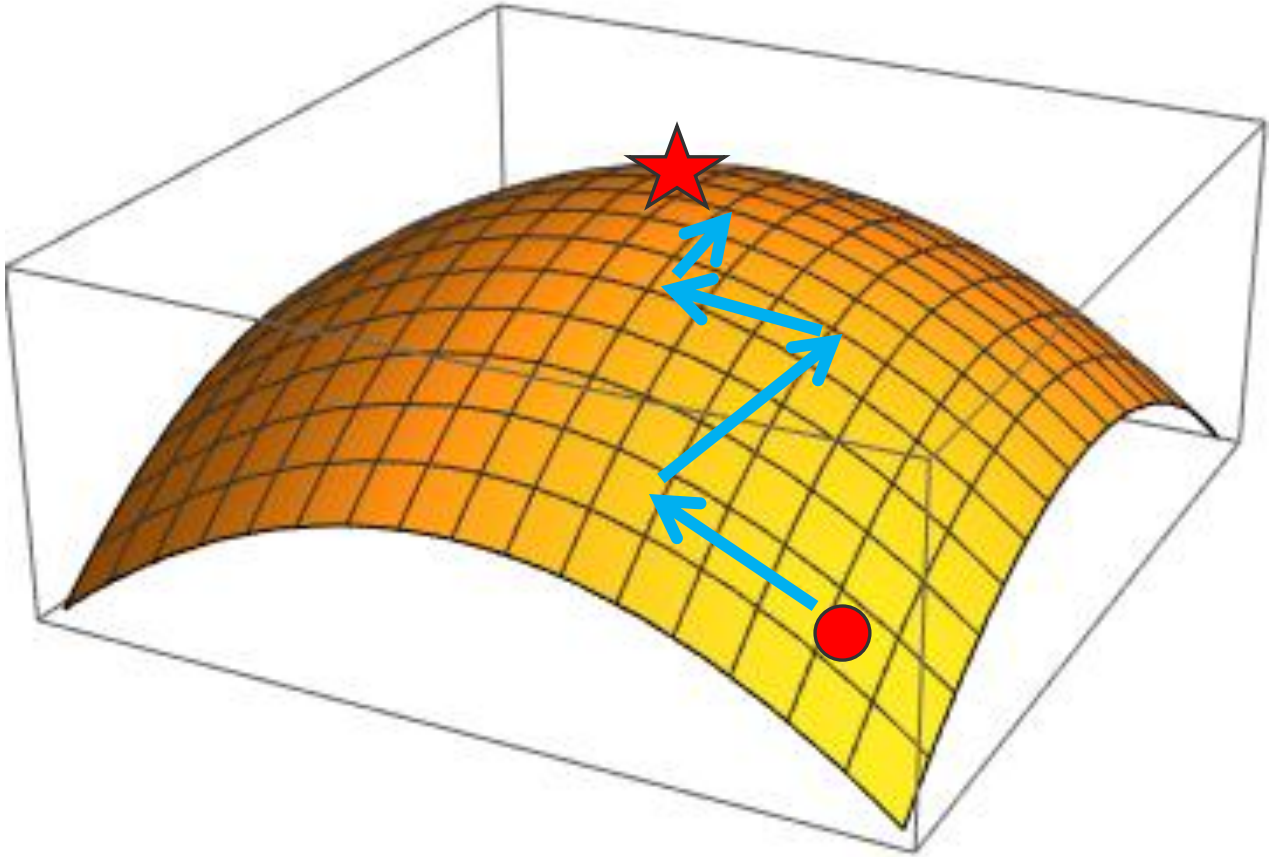
Gradient Ascent



Gradient Ascent



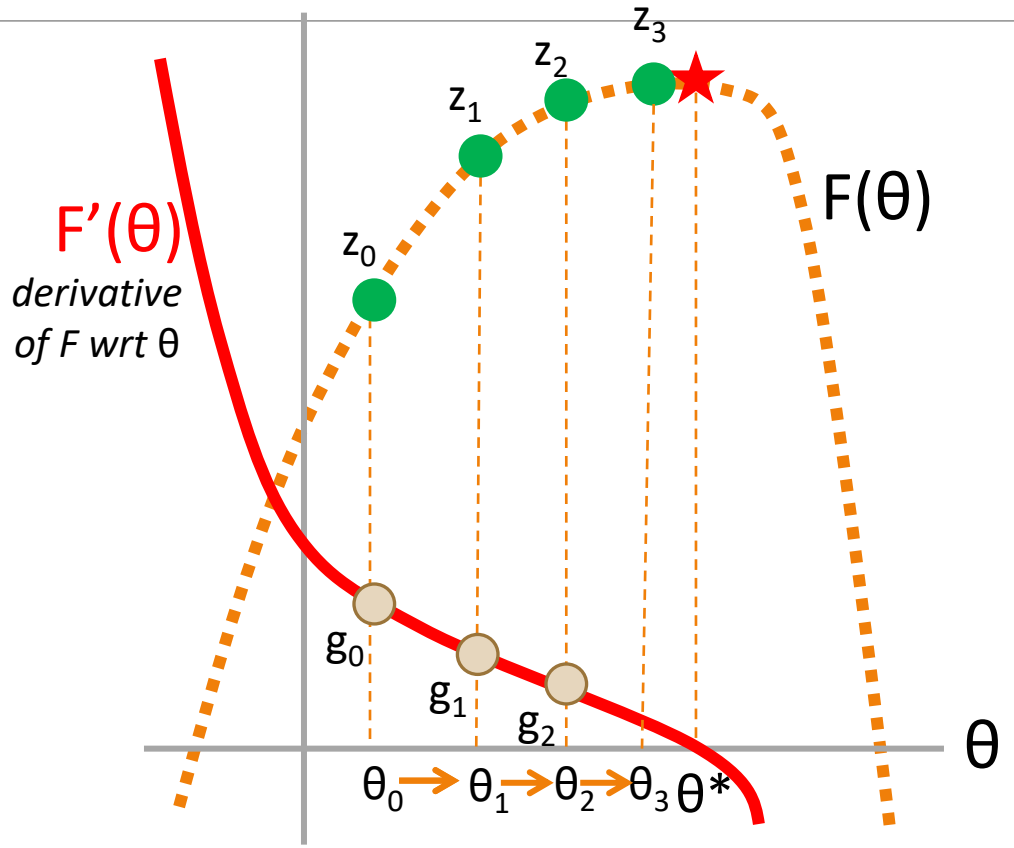
Gradient Ascent



What if you can't find the roots? Follow the **gradient**

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

K-dimensional vectors



Outline

Maximum Entropy classifiers

- Defining the model: Discriminatively
- Defining the objective
- Learning: Optimizing the objective
- Defining the model: Generatively

Maxent Models for Classification: Discriminatively or Generatively Trained

Directly model the posterior

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

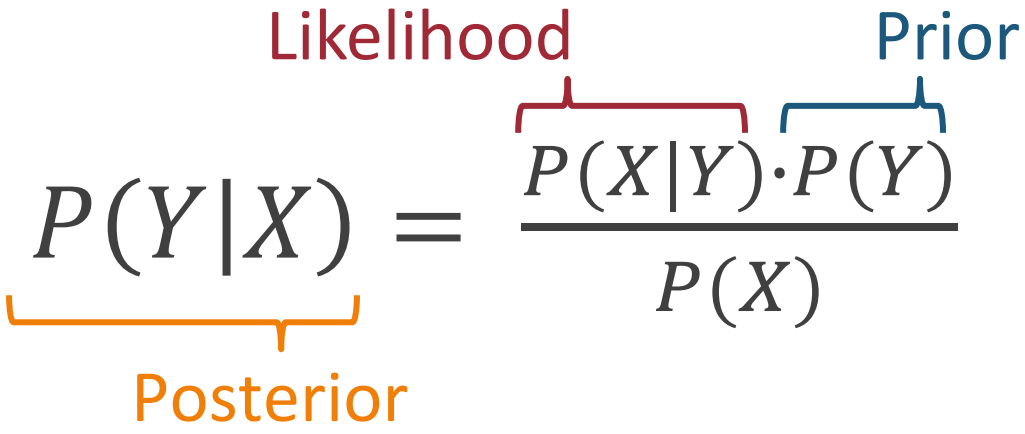
Discriminatively trained classifier

Model the posterior with Bayes rule

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

Generatively trained classifier with maxent-based language model

Bayes' Rule



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

Bayes' Rule

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

$$P(\text{ENTAILED} \mid \text{s: Michael Jordan, coach Phil Jackson and the star cast, including Scottie Pippen, took the Chicago Bulls to six National Basketball Association championships. h: The Bulls basketball team is based in Chicago.})$$

$$= \frac{P(\text{s: Michael Jordan, coach Phil Jackson and the star cast, including Scottie Pippen, took the Chicago Bulls to six National Basketball Association championships. h: The Bulls basketball team is based in Chicago.} \mid \text{ENTAILED}) \cdot P(\text{ENTAILED})}{P(\text{s: Michael Jordan, coach Phil Jackson and the star cast, including Scottie Pippen, took the Chicago Bulls to six National Basketball Association championships. h: The Bulls basketball team is based in Chicago.})}$$

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 (back in 2025…)