CMSC 473/673
Natural Language Processing

Instructor: Lara J. Martin (she/they)

TA: Duong Ta (he)

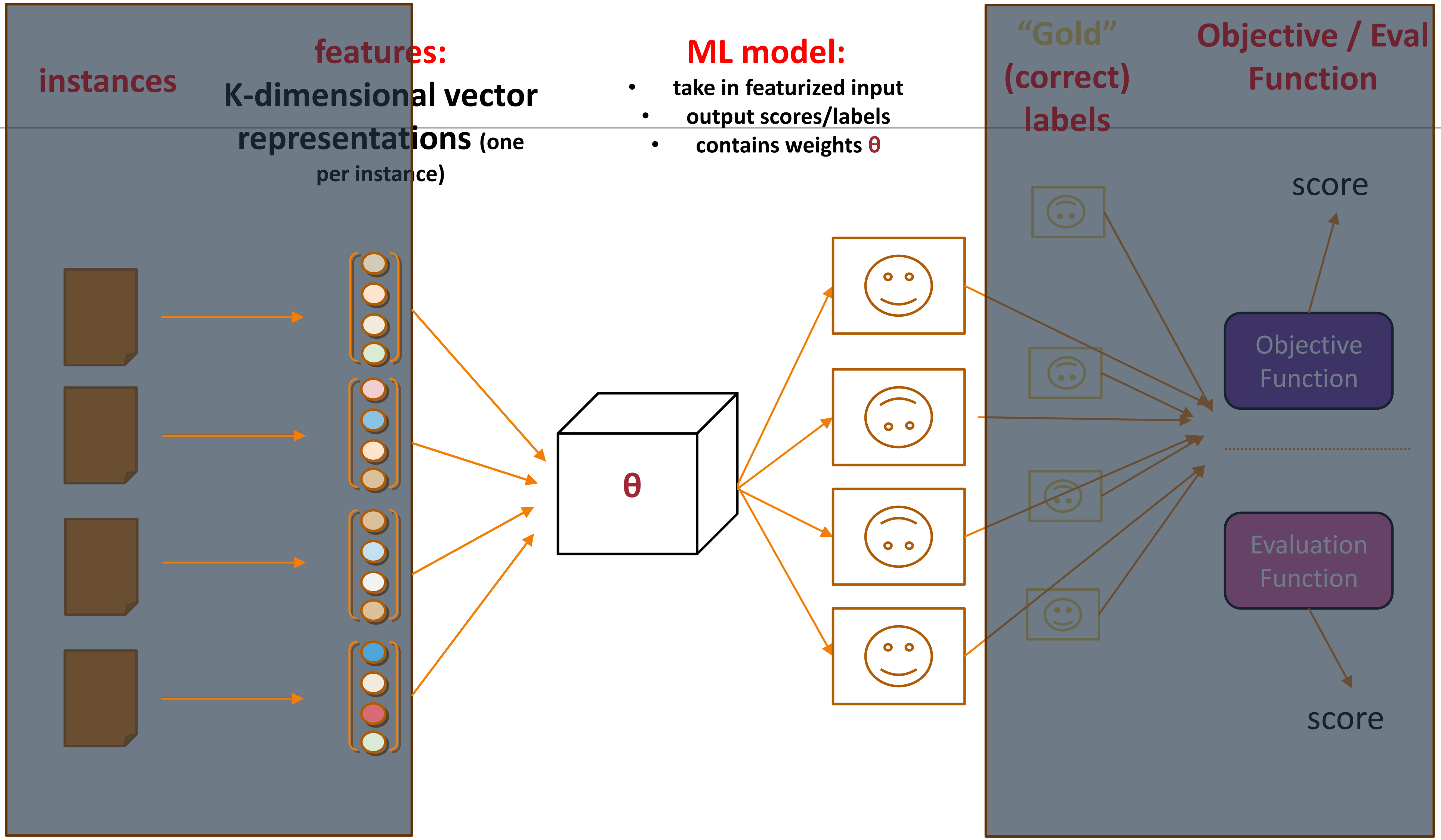
Slides modified from Dr. Frank Ferraro

Learning Objectives

Define an objective for LR modeling

Visualize the learning process for maxent models

Defining the Model



Review: Terminology (with variables)

Posterior probability:

$$p(\ Y = \text{label}_1\ |X) \text{ vs. } p(Y = \text{label}_0\ |X)$$

Conditional probabilities:

$$p(Y = \text{label}_1 \mid X) + p(Y = \text{label}_0 \mid X) = 1$$

$$p(Y = \text{label}_1 \mid X) \geq 0,$$
$$p(Y = \text{label}_0 \mid X) \geq 0$$

Posterior probability: probability of event Y with knowledge that X has occurred

NLP pg. 450

Conditional probability: probability of event Y, assuming event X happens too

NLP pg. 449

💡 Key Take-away 💡

We will ***learn*** this

$$\boldsymbol{p}(\boldsymbol{Y} \mid \boldsymbol{X})$$

Review: Turning Scores into Probabilities

score(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. , ENTAILED) > score(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. , NOT ENTAILED)

p(ENTAILED | 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.) > p(NOT ENTAILED | 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.)

KEY IDEA

Review: Turning Scores into Probabilities (More Generally)

$$\text{score}(x, y_1) > \text{score}(x, y_2)$$

$$p(y_1 | x) > p(y_2 | x)$$

KEY IDEA

Review: Maxent Modeling

p(ENTAILED | 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.) $\propto$

exp(Dot_product of Entailed weight_vec feature_vec(🖹))

K different weights…

for K different features

multiplied and then summed

Review: Maxent Modeling

$p($ ENTAILED $|$ 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. $) \propto$

$\exp(\theta_{\text{ENTAILED}}^{T} f($  $))$

K different weights…

for K different features

multiplied and then summed

Review: Different Notation, Same Meaning

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

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

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

Review: Representing a Linguistic "Blob"

User-defined

Integer representation/one-hot encoding

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

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

Model-produced

Dense embedding

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

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

Review: Bag-of-words as a Function

Based on some tokenization, turn an input document into an array (or dictionary or set) of its unique vocab items

Think of getting a BOW rep. as a function f

input: Document

output: Container of size E, indexable by each vocab type v

Some Bag-of-words Functions

Kind	Type of f_v	Interpretation
Binary	0, 1	Did *v* appear in the document?
Count-based	Natural number (int >= 0)	How often did *v* occur in the document?
Averaged	Real number (>=0, <= 1)	How often did *v* occur in the document, normalized by doc length?
TF-IDF (term frequency, inverse document frequency)	Real number (>= 0)	How frequent is a word, tempered by how prevalent it is across the corpus (to be covered later!)
…		

Q: Is this a reasonable representation?

Q: What are some tradeoffs (benefits vs. costs)?

Useful Terminology: n-gram

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

Colorless green ideas sleep furiously

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

Templated Features

Define a feature *f_clue*(🖹) for each clue you want to consider

Not a regular term

The feature *f_clue* **fires** if the clue applies to/can be found in 🖹

f_clue is often a target phrase (an n-gram)

Maxent Modeling:
Templated Binary Feature Functions

p(ENTAILED | 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.) ∝

exp($\text{weight}_{1\text{, Entailed}} * \text{applies}_1$(🗎) +
$\text{weight}_{1\text{, Entailed}} * \text{applies}_2$(🗎) +
$\text{weight}_{1\text{, Entailed}} * \text{applies}_3$(🗎) +
…))

$$\text{applies}_{\text{target}}(🗎) = \begin{cases} 1, \text{target } \mathit{matches} \ 🗎 \\ 0, \qquad \text{otherwise} \end{cases}$$

binary

Example of a Templated Binary Feature Functions

$$\text{applies}_{\text{target}}(\text{🖹}) = \begin{cases} 1, \text{target } \textit{matches } \text{🖹} \\ 0, \quad\quad \text{otherwise} \end{cases}$$

$$\text{applies}_{\text{ball}}(\text{🖹}) = \begin{cases} 1, \text{ball } \textit{in} \text{ both s and h of } \text{🖹} \\ 0, \quad\quad \text{otherwise} \end{cases}$$

TPS: Example of a Templated Binary Feature Functions

$$\text{applies}_{\text{target}}(\text{🗎}) = \begin{cases} 1, \text{target } matches \text{ 🗎} \\ 0, \quad \text{otherwise} \end{cases}$$

$$\text{applies}_{\text{ball}}(\text{🗎}) = \begin{cases} 1, \text{ball } in \text{ both s and h of 🗎} \\ 0, \quad \text{otherwise} \end{cases}$$

Q: If there are V vocab types and L label types:

1. How many features are defined if unigram targets are used (w/ each label)?

2. How many features are defined if bigram targets are used (w/ each label)?

3. How many features are defined if unigram and bigram targets are used (w/ each label)?

Example of a Templated Binary Feature Functions

$$\text{applies}_{\text{target}}(\text{🗎}) = \begin{cases} 1, & \text{target } \textit{matches } \text{🗎} \\ 0, & \text{otherwise} \end{cases}$$

$$\text{applies}_{\text{ball}}(\text{🗎}) = \begin{cases} 1, & \text{ball } \textit{in} \text{ both s and h of 🗎} \\ 0, & \text{otherwise} \end{cases}$$

Q: If there are V vocab types and L label types:

1. How many features are defined if unigram targets are used (w/ each label)?

$$VL$$

2. How many features are defined if bigram targets are used (w/ each label)?

3. How many features are defined if unigram and bigram targets are used (w/ each label)?

Example of a Templated Binary Feature Functions

$$\text{applies}_{\text{target}}(\text{🖹}) = \begin{cases} 1, \text{target } matches \text{ 🖹} \\ 0, \quad\quad \text{otherwise} \end{cases}$$

$$\text{applies}_{\text{ball}}(\text{🖹}) = \begin{cases} 1, \text{ball } in \text{ both s and h of 🖹} \\ 0, \quad\quad \text{otherwise} \end{cases}$$

Q: If there are V vocab types and L label types:

1. How many features are defined if unigram targets are used (w/ each label)?

VL

2. How many features are defined if bigram targets are used (w/ each label)?

V^2L

3. How many features are defined if unigram and bigram targets are used (w/ each label)?

Example of a Templated Binary Feature Functions

$$\text{applies}_{\text{target}}(🗎) = \begin{cases} 1, \text{target } matches \text{ 🗎} \\ 0, \quad \text{otherwise} \end{cases}$$

$$\text{applies}_{\text{ball}}(🗎) = \begin{cases} 1, \text{ball } in \text{ both s and h of 🗎} \\ 0, \quad \text{otherwise} \end{cases}$$

Q: If there are V vocab types and L label types:

1. How many features are defined if unigram targets are used (w/ each label)?

VL

2. How many features are defined if bigram targets are used (w/ each label)?

V^2L

3. How many features are defined if unigram and bigram targets are used (w/ each label)?

$(\mathrm{V} + V^2)L$

Defining the Objective

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

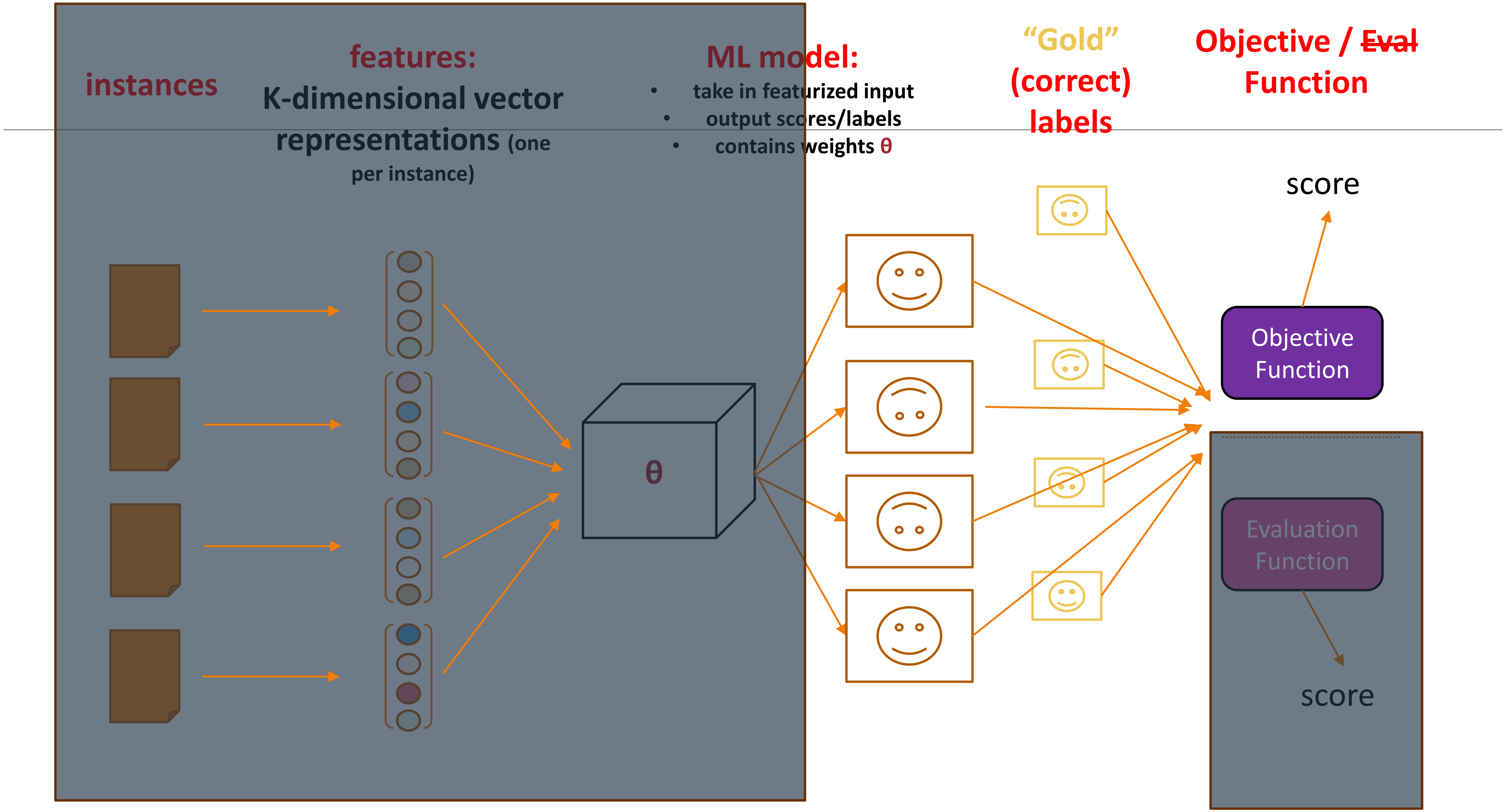
probabilistic model

$$\Downarrow$$

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

objective

Defining the Objective



Primary Objective: Likelihood

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

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

In classification, this is p(label | 🗎)

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

Objective = Full Likelihood? (Classification)

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

These values can have very small magnitude ➔ underflow

Differentiating this product could be a pain

Logarithms

(0, 1] ➔ (-∞, 0]

Products ➔ Sums

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

$$\log(a/b) = \log(a) - \log(b)$$

Inverse of exp

$$\log(\exp(x)) = x$$

Think-Pair-Share

How might you find the log of this?

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

Log-Likelihood (Classification)

Wide range of (negative) numbers

Sums are more stable

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

Products ➔ Sums

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

log(a/b) = log(a) – log(b)

Maximize Log-Likelihood (Classification)

Original maxent equation

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

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

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

Inverse of exp

log(exp(x)) = x

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

Maximize Log-Likelihood (Classification)

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

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

$$= F(\theta)$$

Equivalent Version 2:
Minimize Cross Entropy Loss

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

Classifier output

True probability (i.e., correct output)

$$L^{\text{xent}}(\hat{y}, y) =$$

index of "1" indicates correct value

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

one-hot vector

objective is convex
(when f(x) is not learned)

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

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

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

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

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

Regularization: Preventing Extreme Values

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

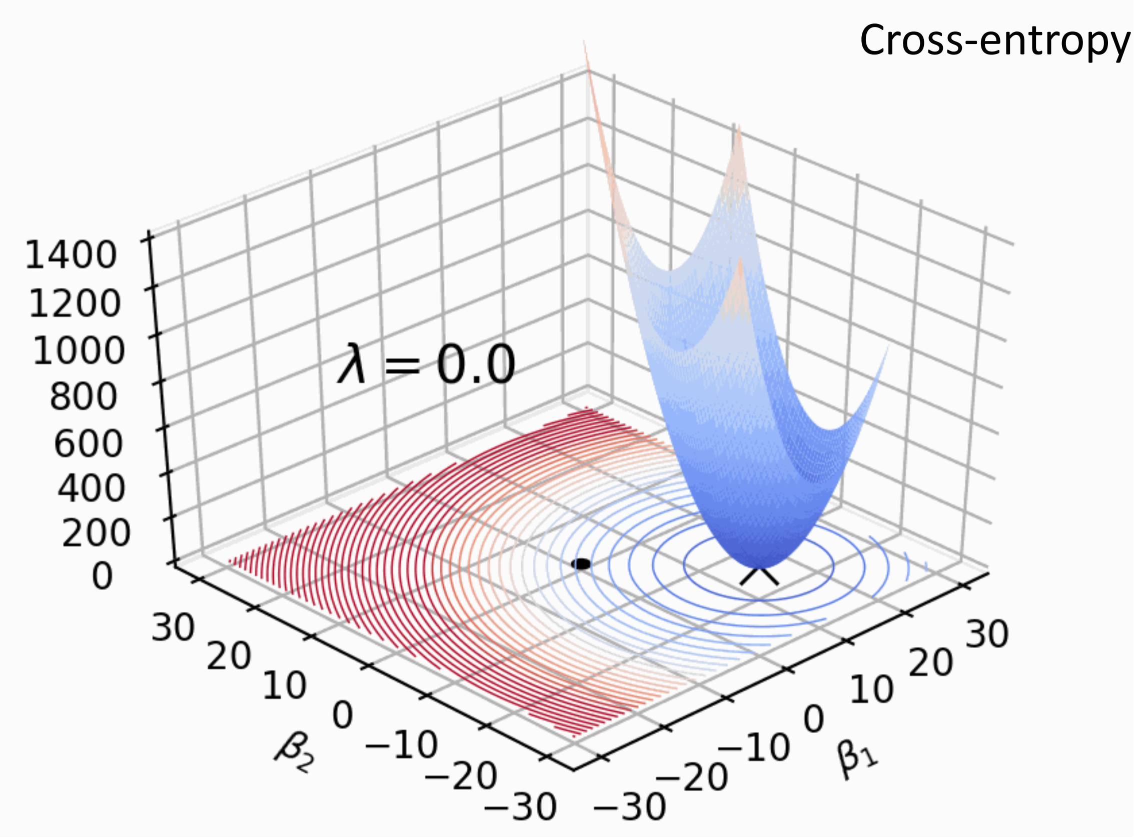
With fixed/predefined features, the values of θ determine how "good" or "bad" our objective learning is

- 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

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

Cross-entropy loss

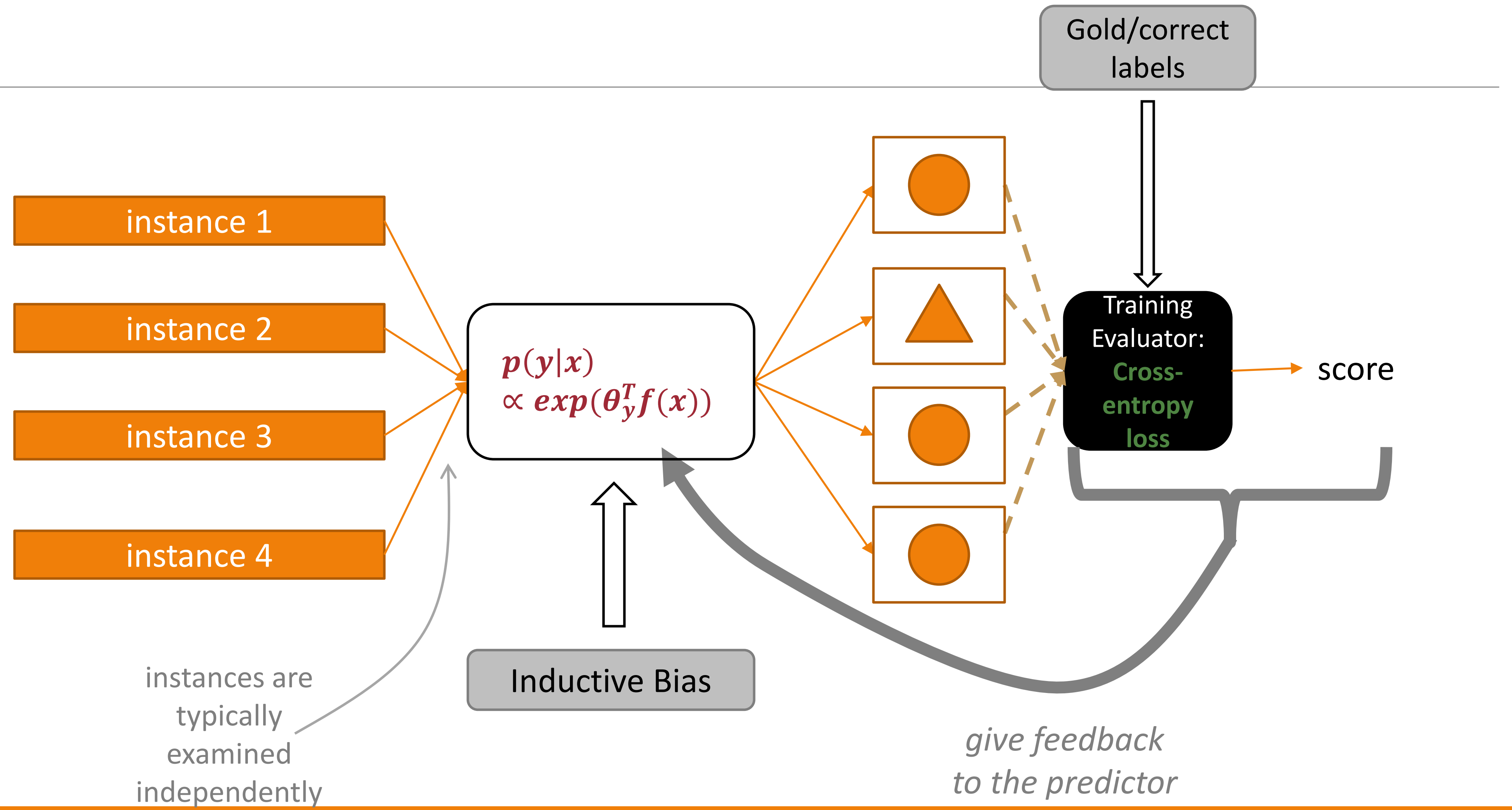


https://explained.ai/regularization/

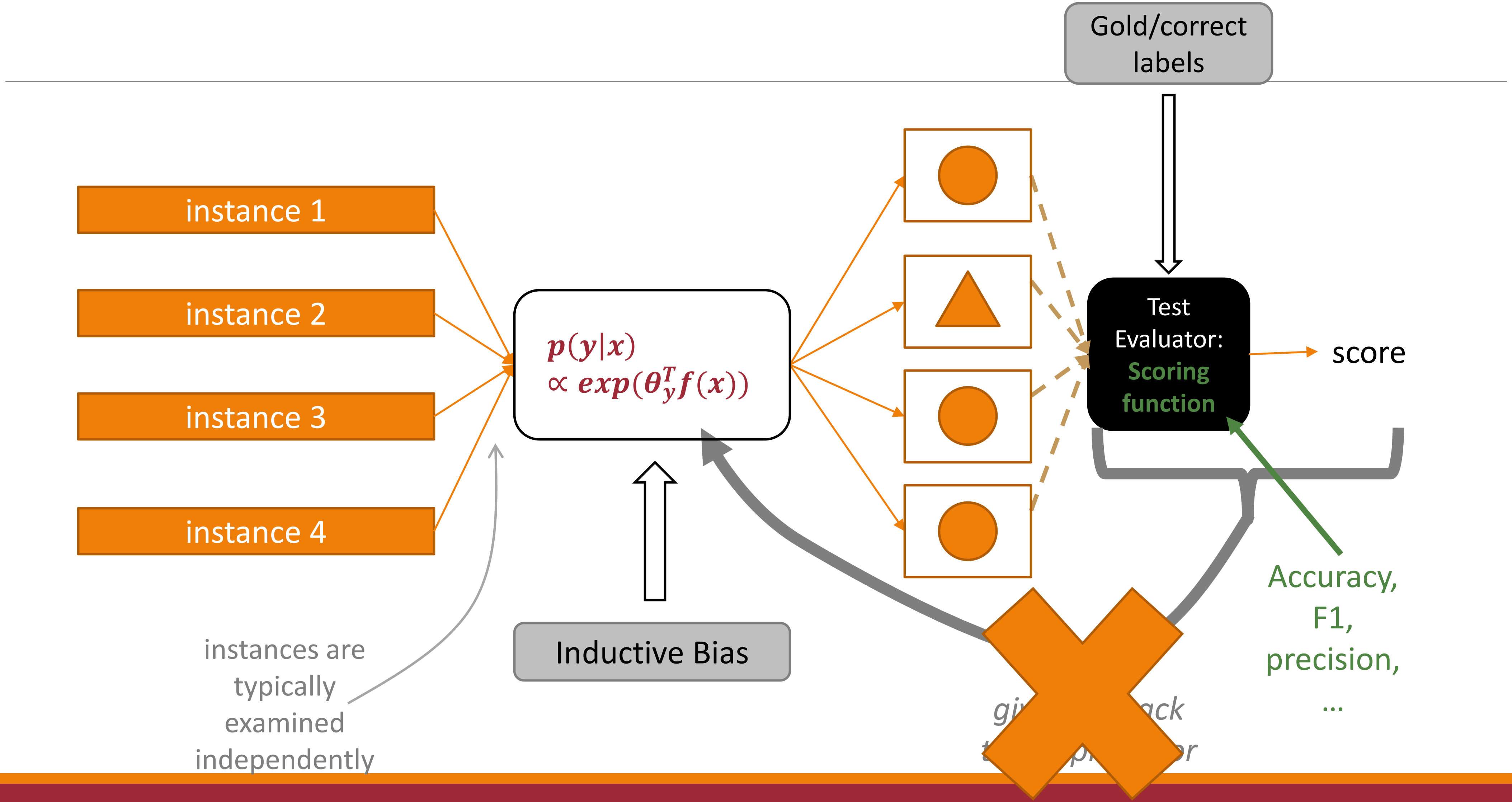
Optimizing the objective

GRADIENT OPTIMIZATION

How do we learn?

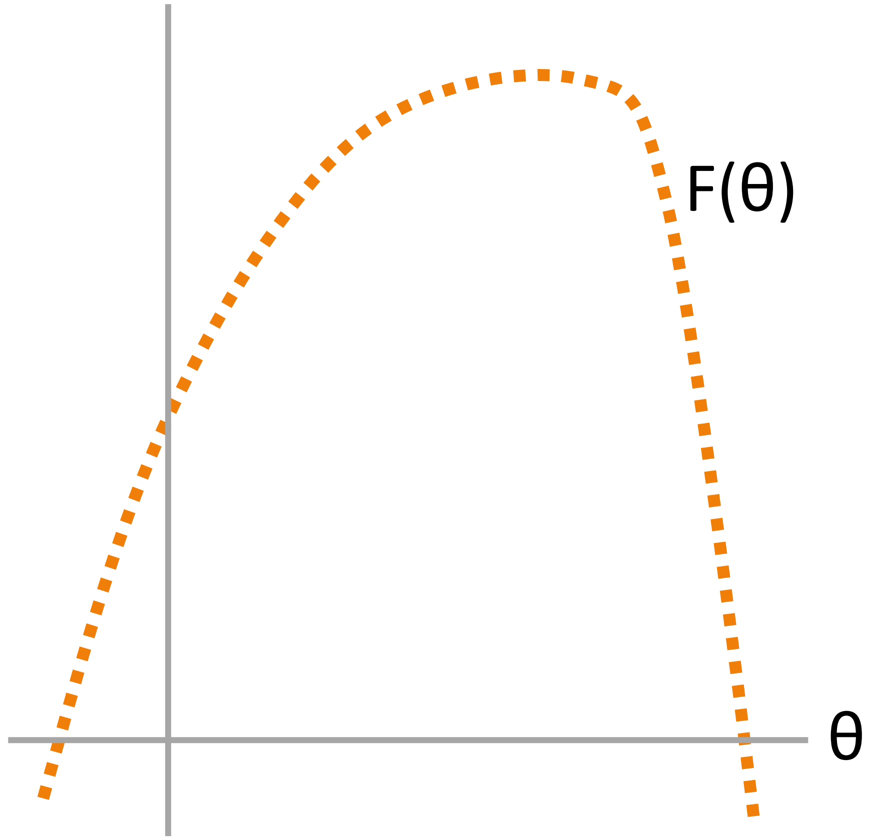


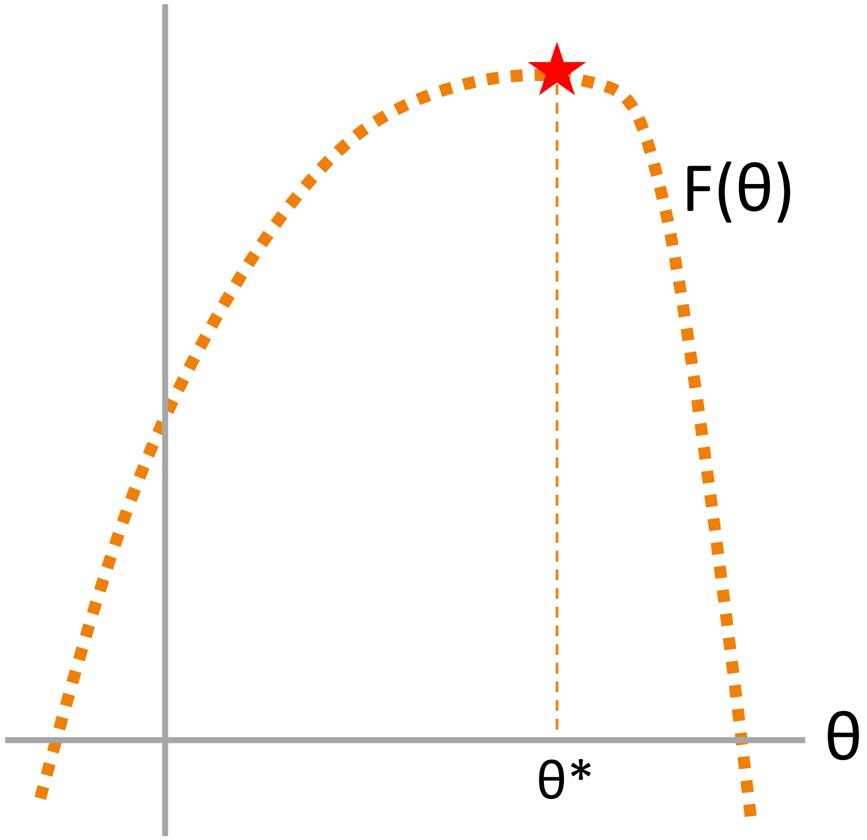
How do we evaluate (or use)?
Change the eval function.



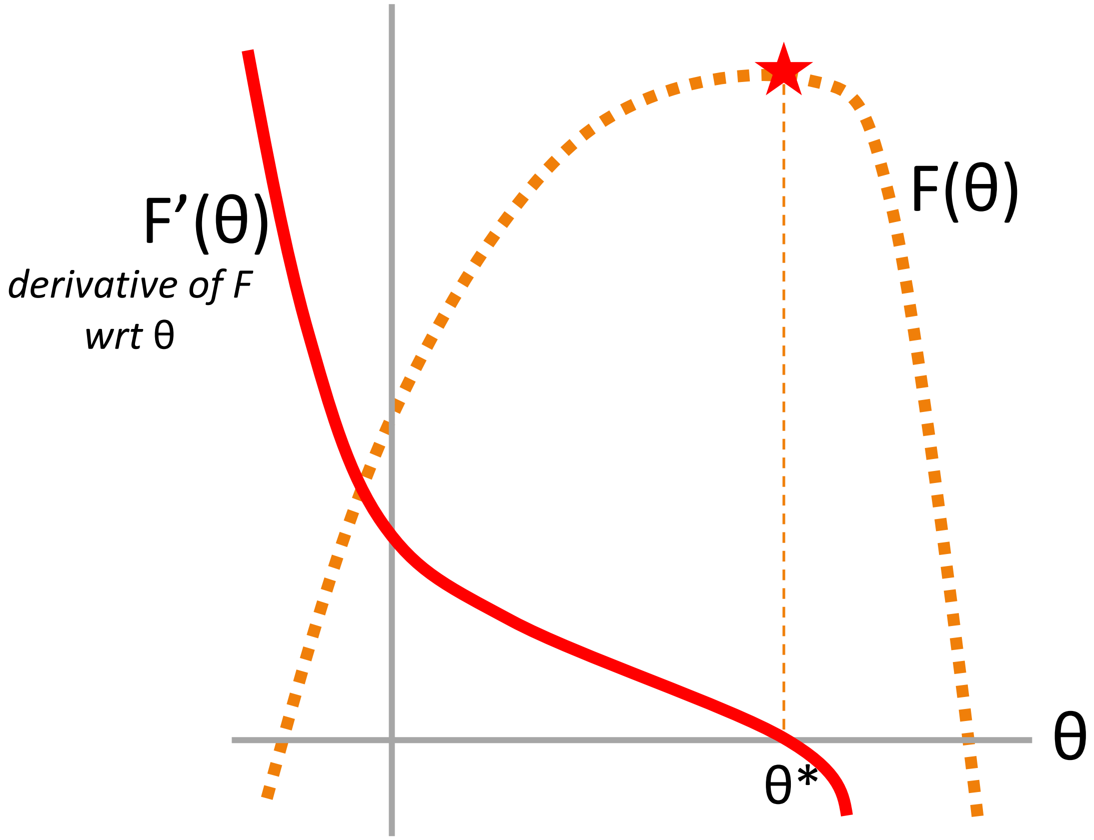
How will we optimize F(θ)?

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)

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

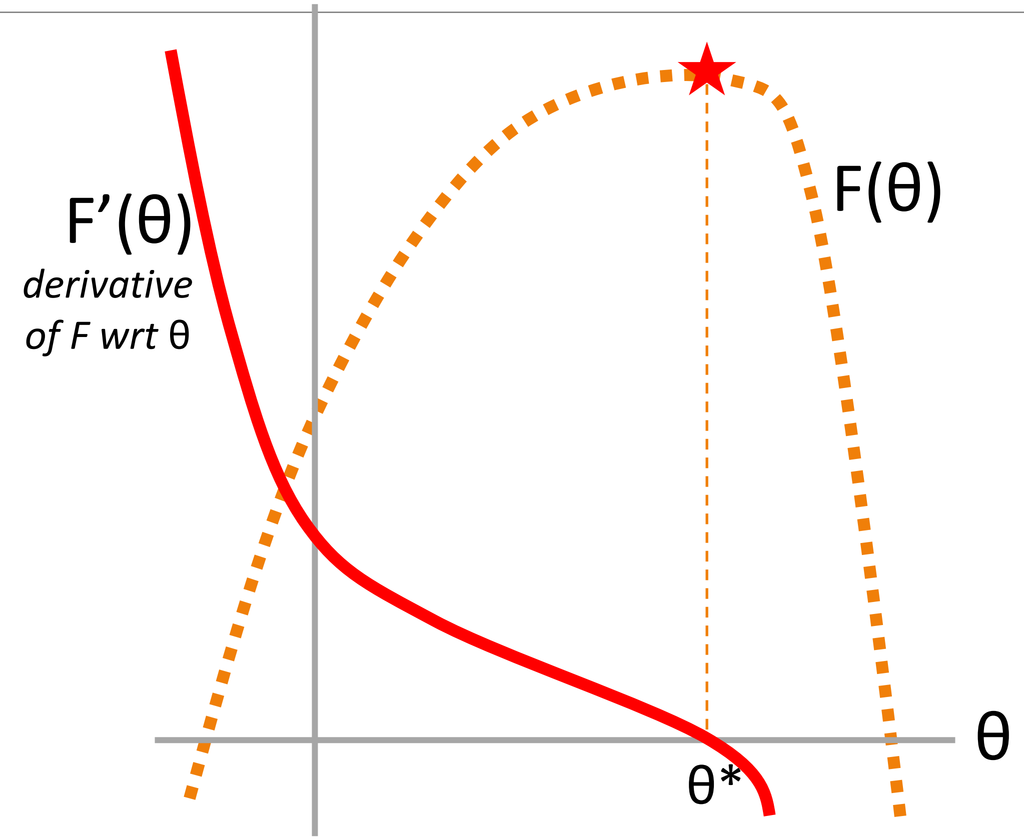
$$\Big\downarrow \textit{ differentiate}$$

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

$$\Big\downarrow \textit{ Solve } F'(x) = 0$$

$$x = 2$$

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



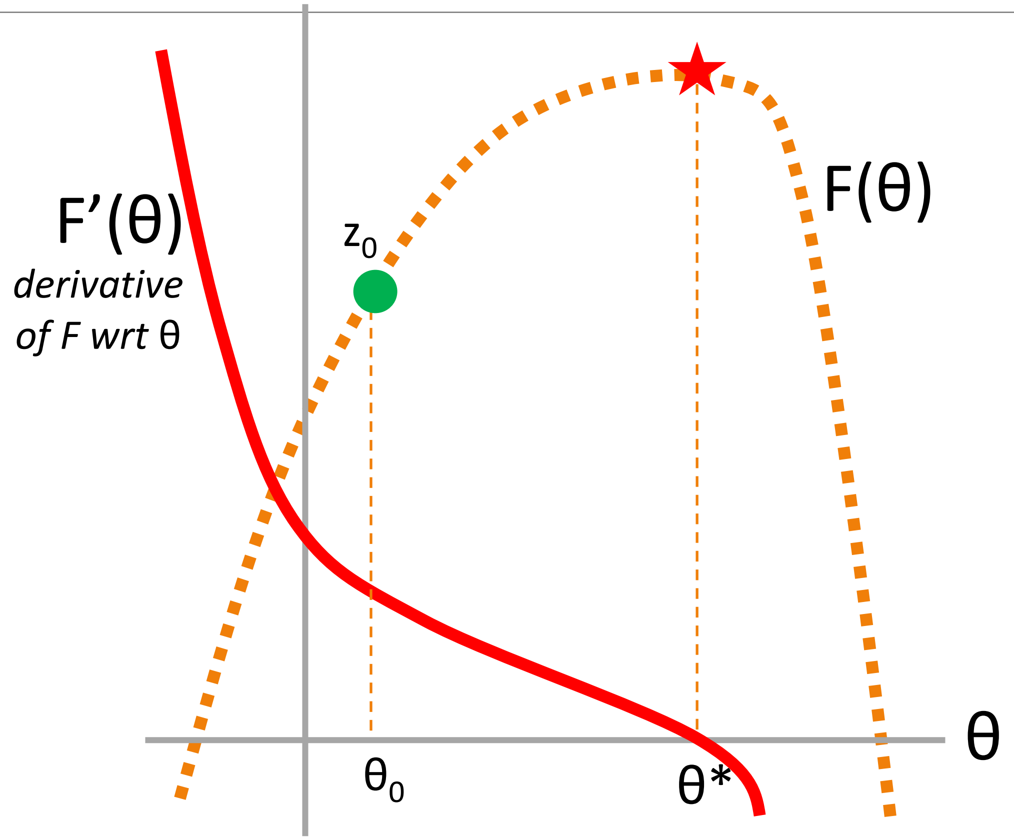
What if you can't find the roots? Follow the derivative

Set t = 0

Pick a starting value θ_t

Until converged:

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



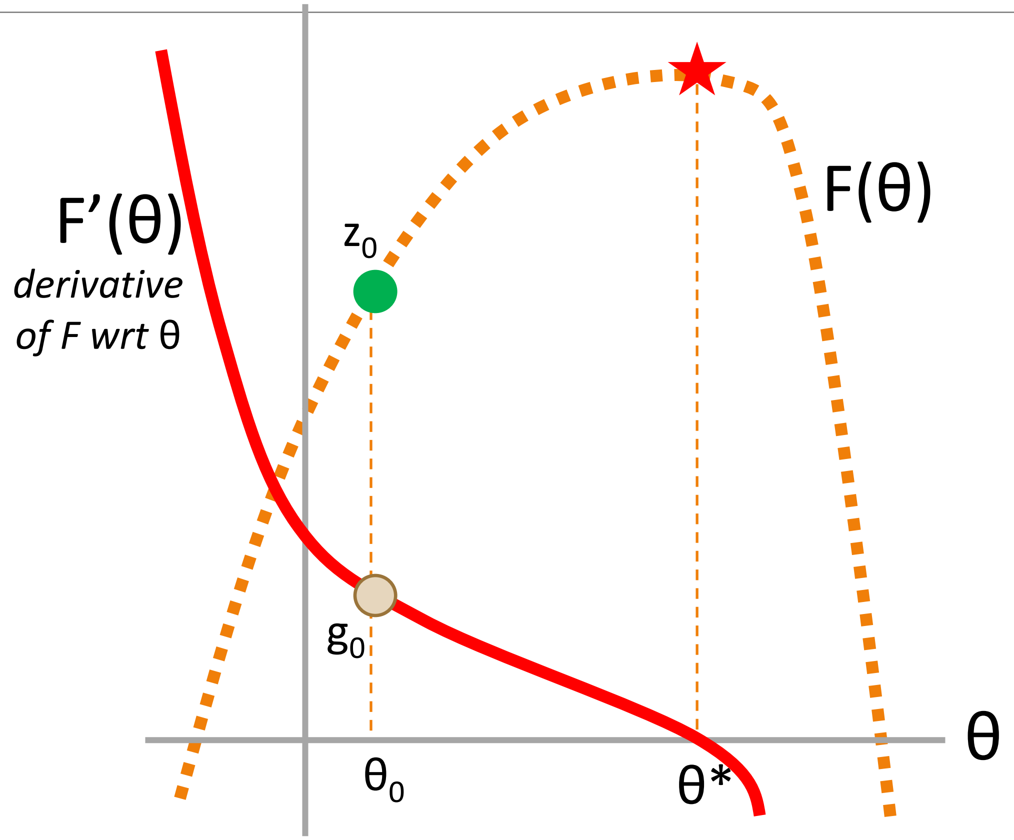
What if you can’t find the roots? Follow the derivative

Set t = 0

Pick a starting value θ_t

Until converged:

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



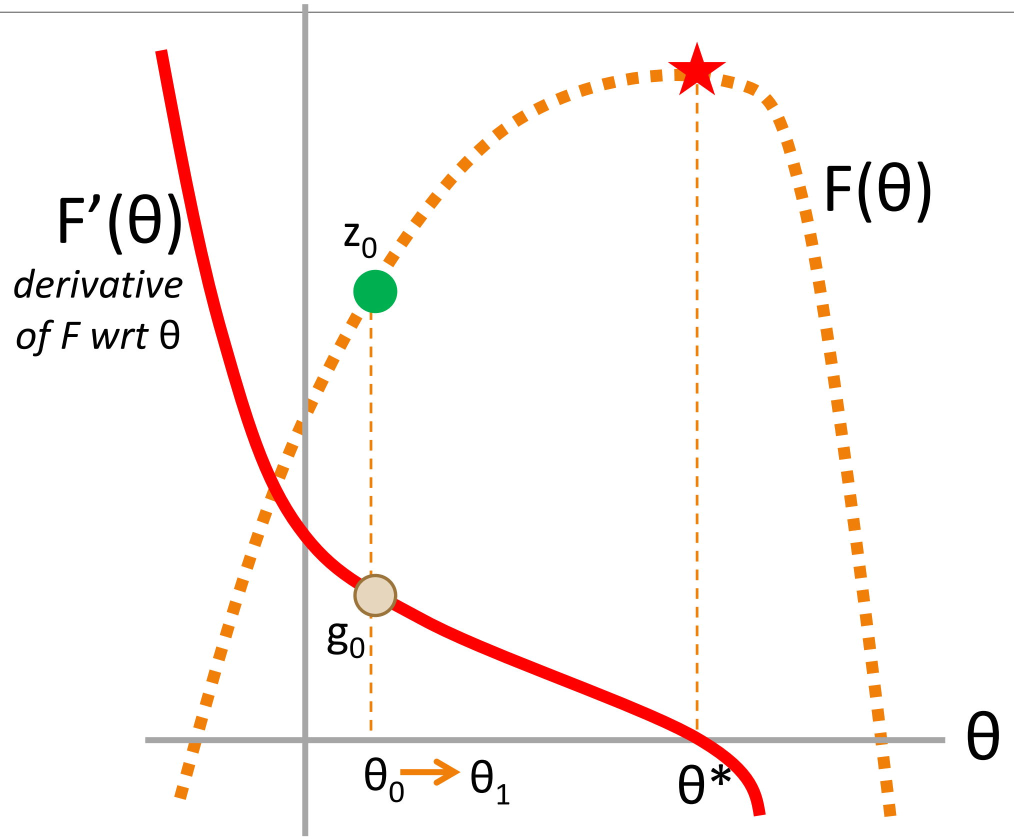
What if you can't find the roots? Follow the derivative

Set t = 0

Pick a starting value θ_t

Until converged:

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



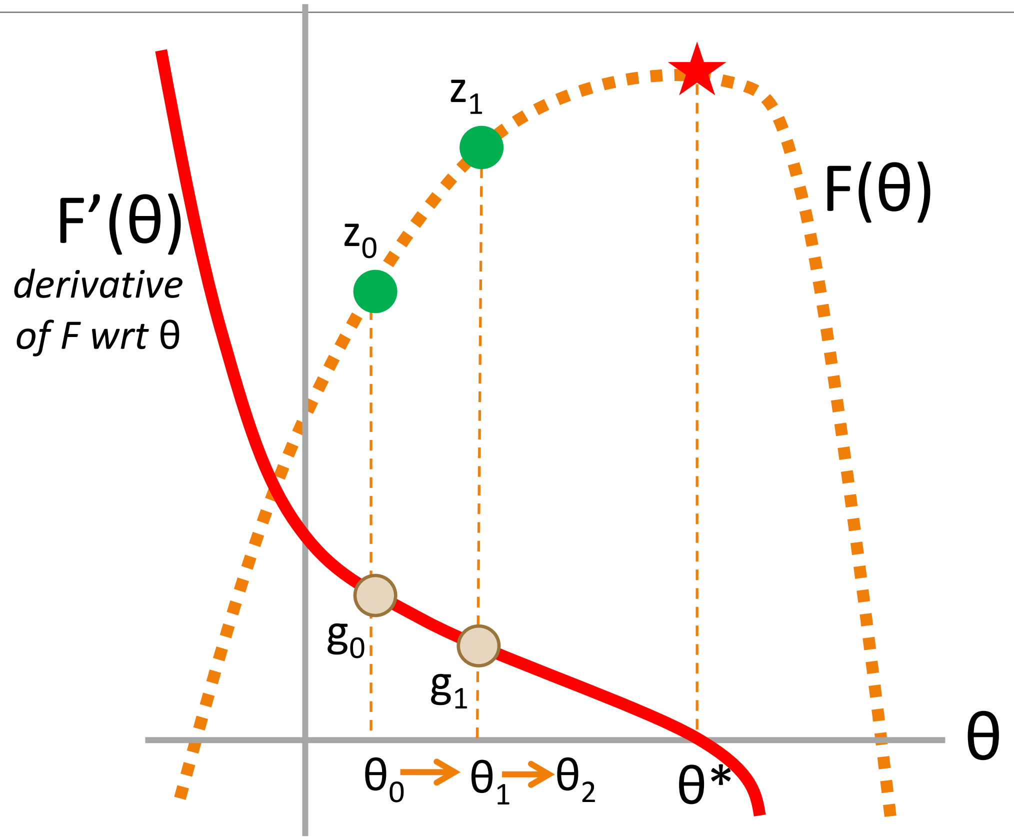
What if you can't find the roots? Follow the derivative

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 ρ_t
4. Set $\theta_{t+1} = \theta_t + \rho_t * g_t$
5. Set t += 1



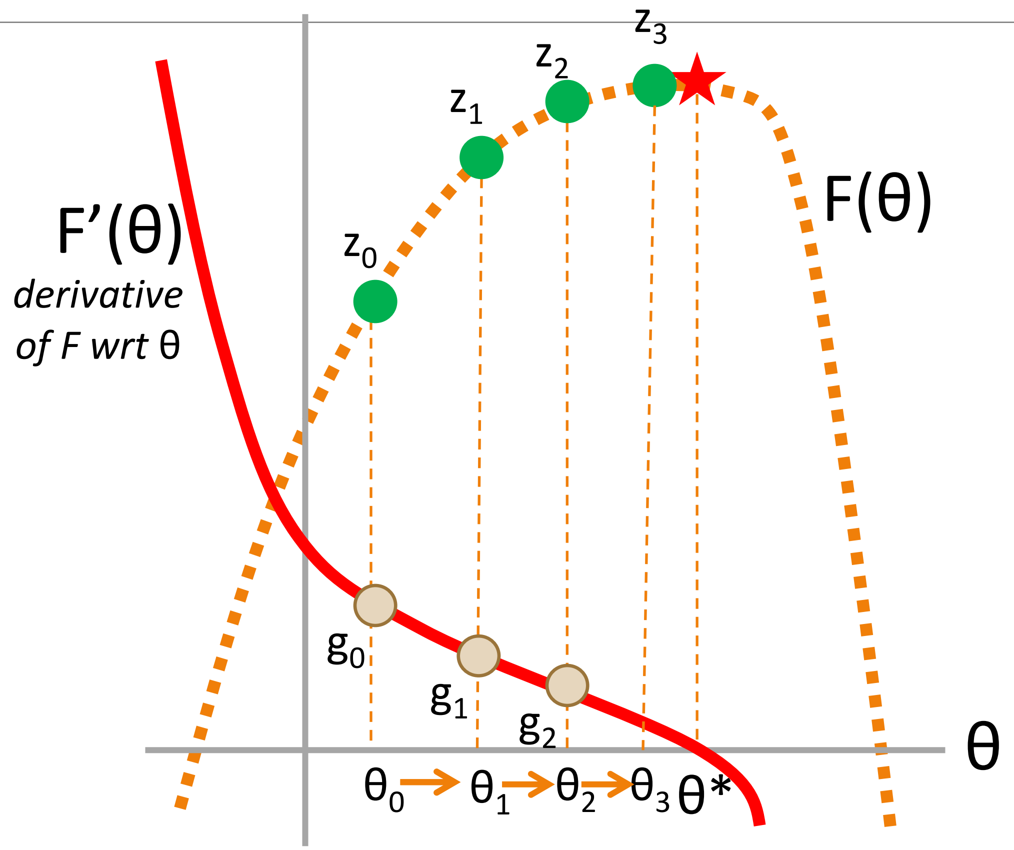
What if you can’t find the roots? Follow the derivative

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 ρ_t
4. Set $\theta_{t+1} = \theta_t + \rho_t * g_t$
5. Set t += 1



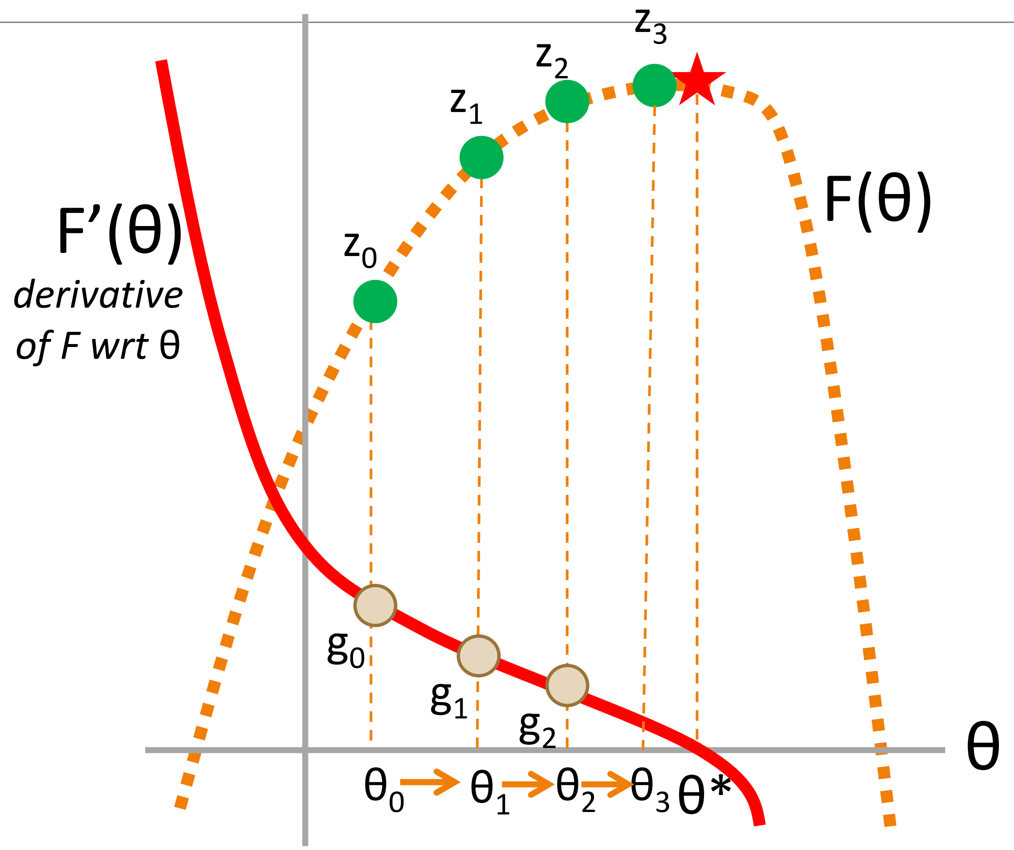
What if you can't find the roots? Follow the derivative

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 ρ_t**
4. Set $\theta_{t+1} = \theta_t + \rho_t * g_t$
5. Set $t \mathrel{+}= 1$

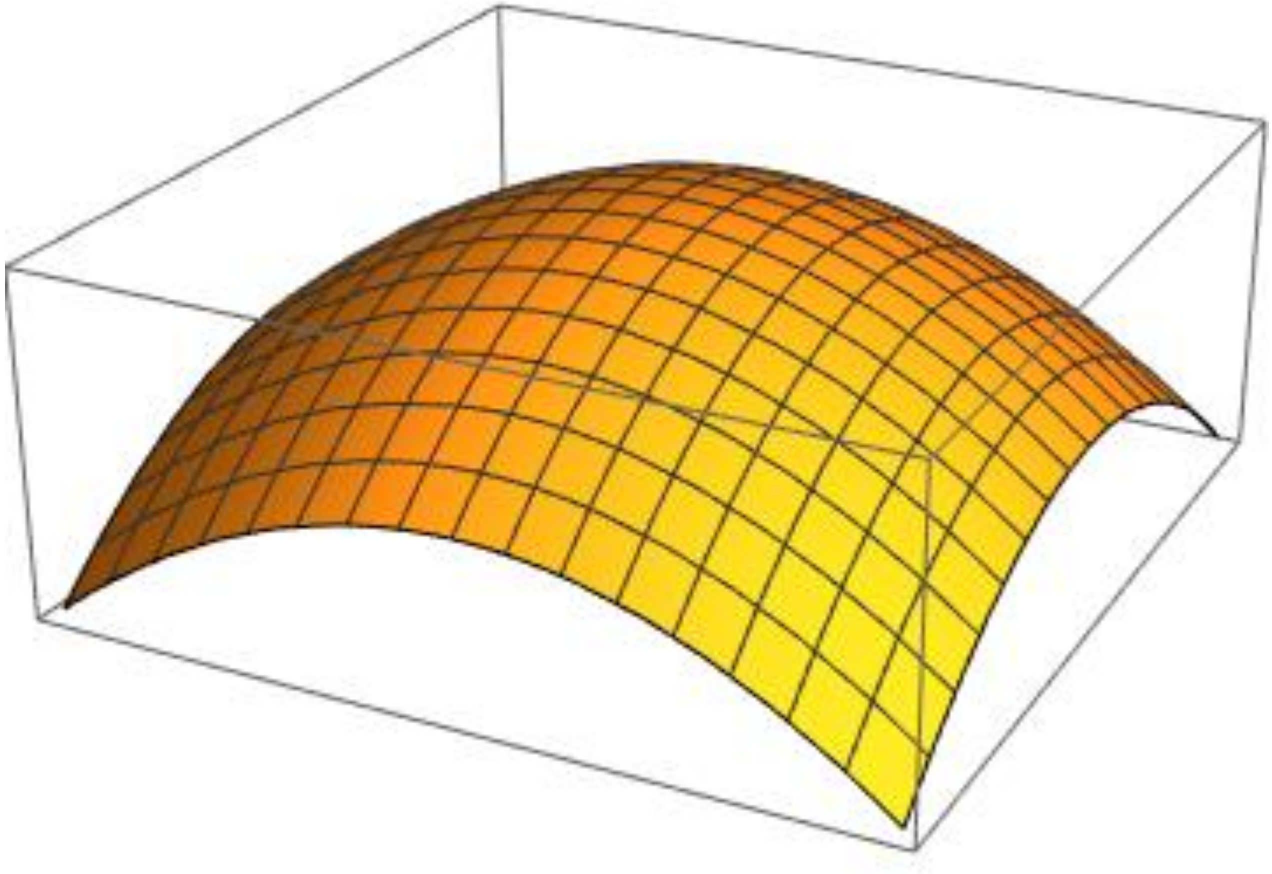


Gradient = Multi-variable derivative

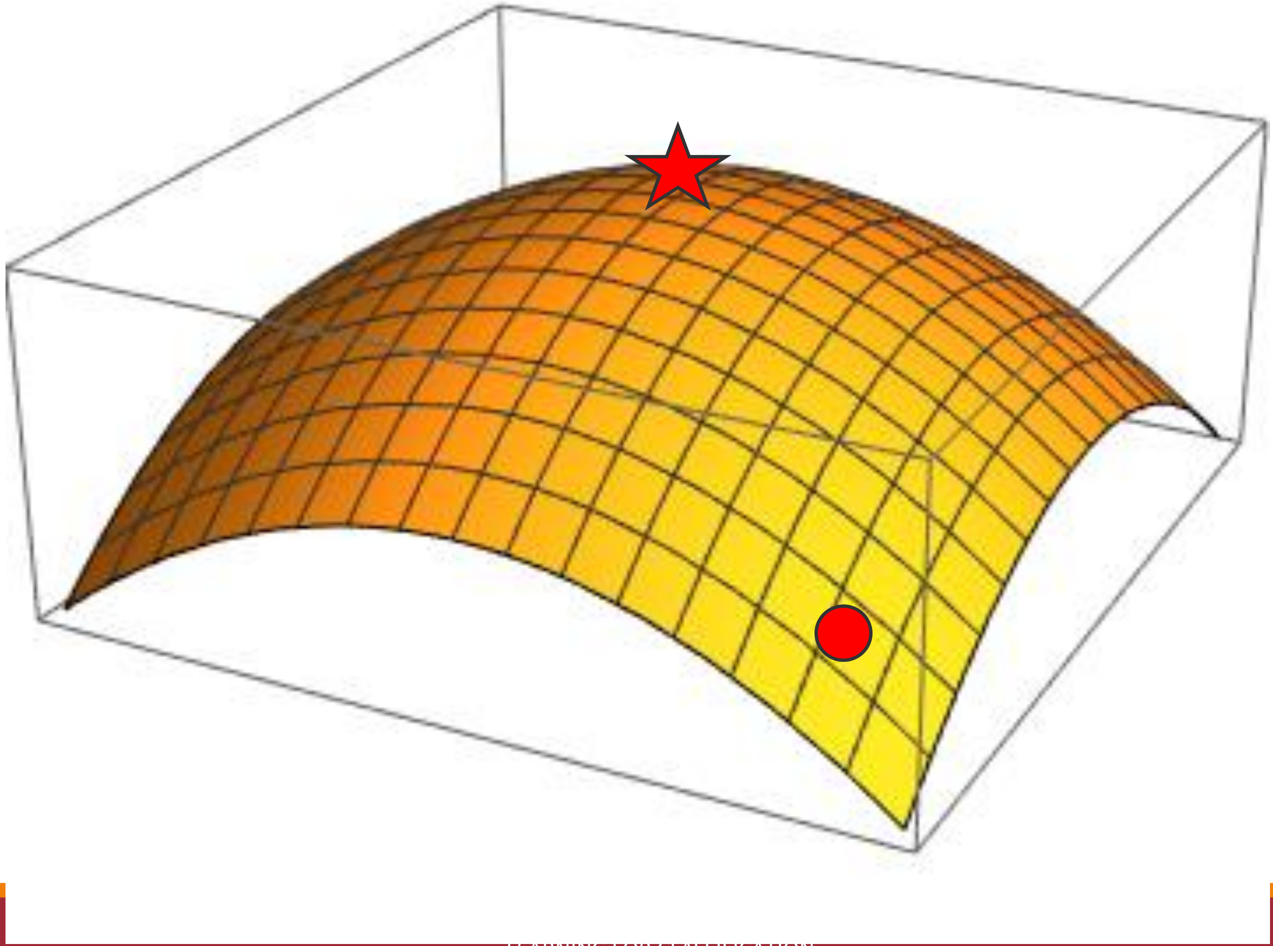
K-dimensional input

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

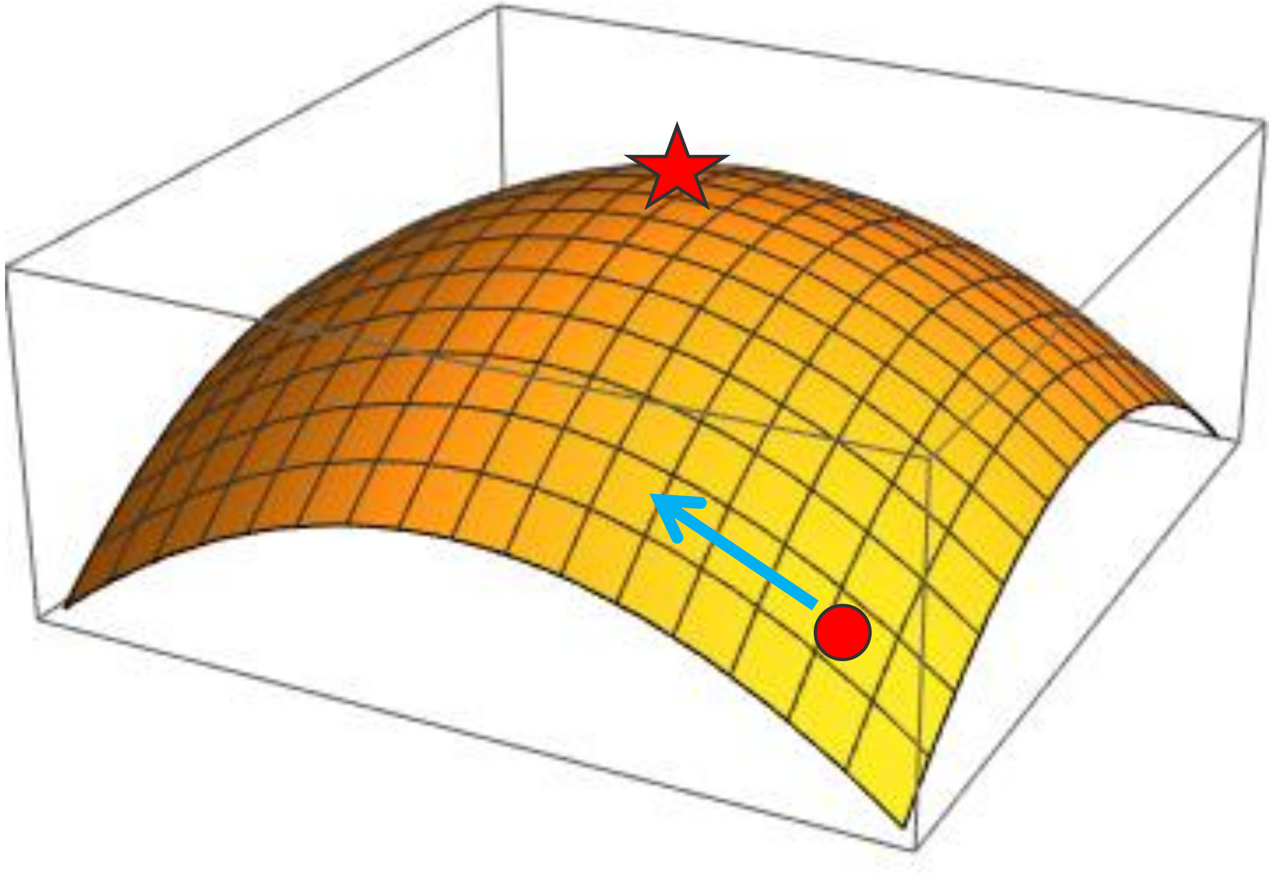
Gradient Ascent



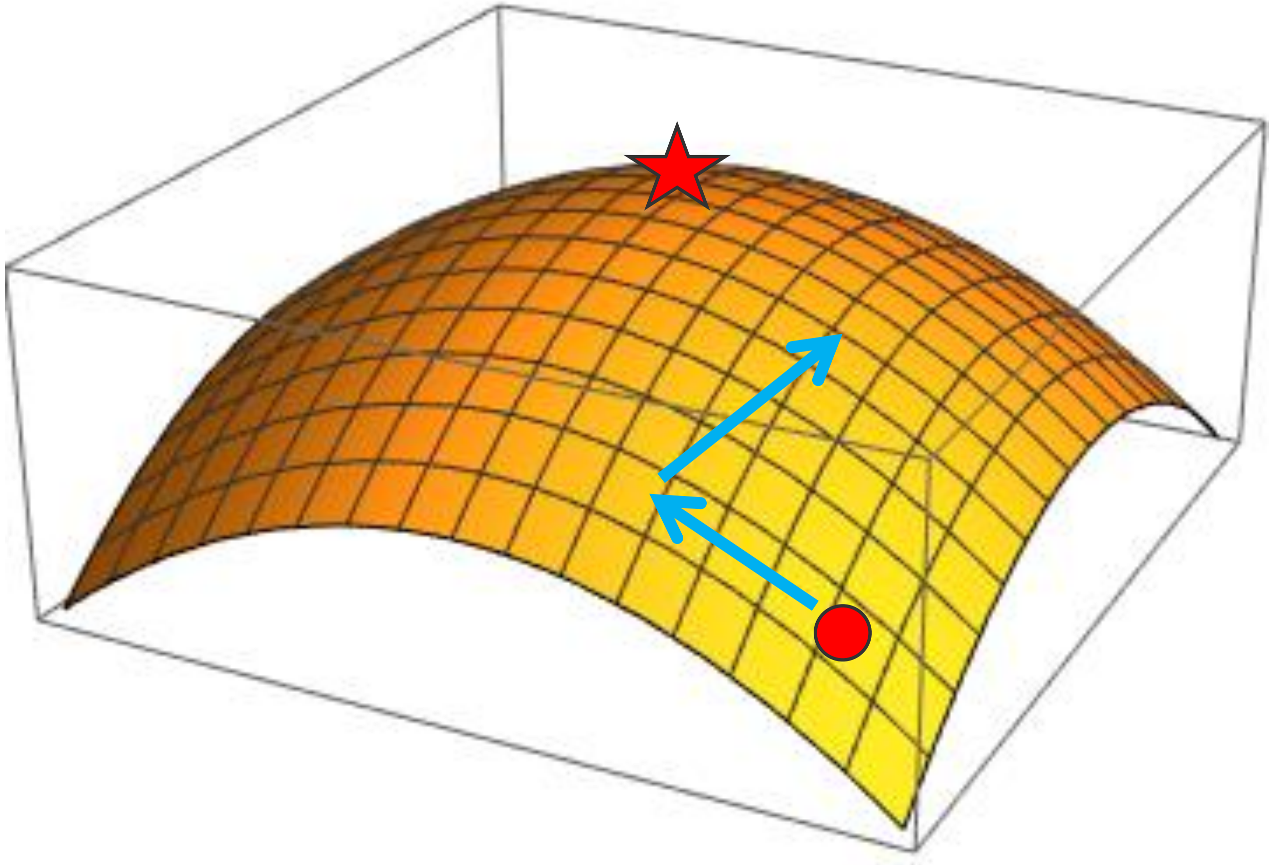
Gradient Ascent



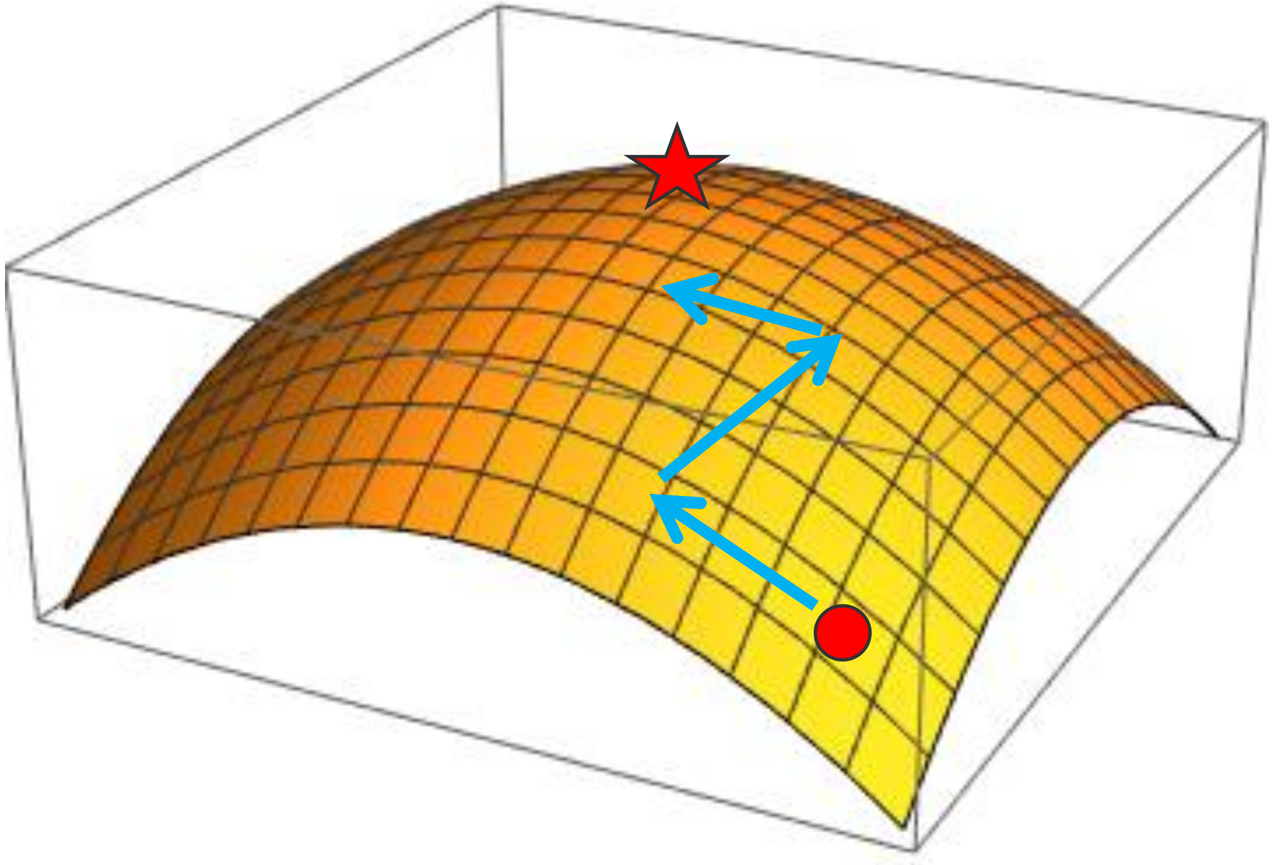
Gradient Ascent



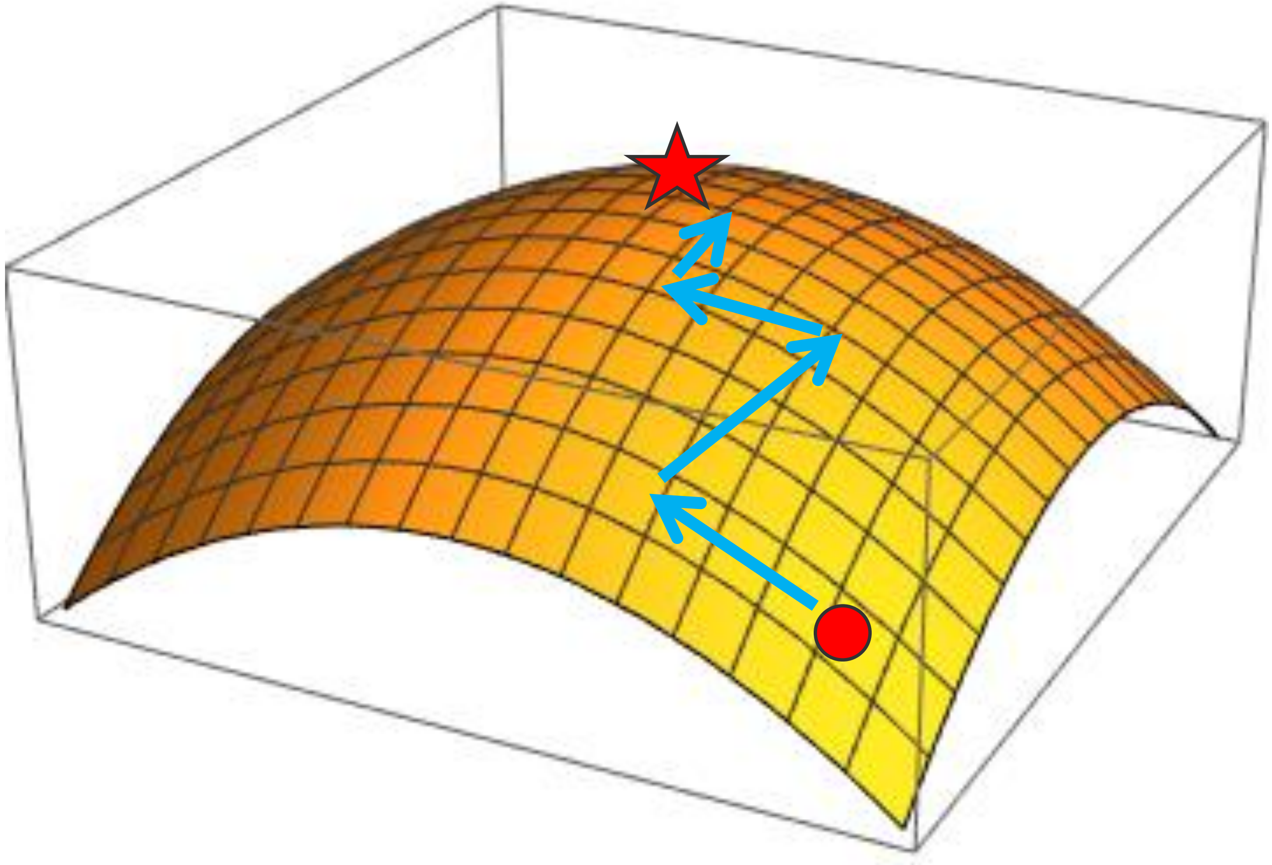
Gradient Ascent



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