10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Stochastic Gradient Descent

+

Probabilistic Learning

(Logistic Regression)

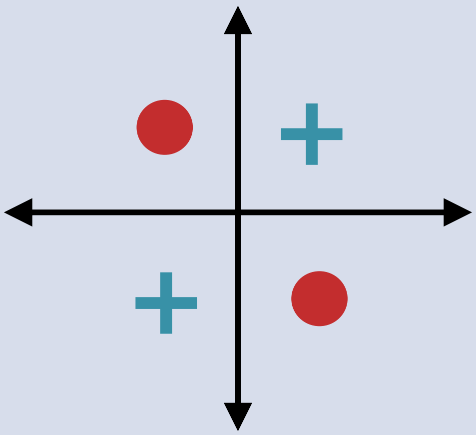
Matt Gormley
Lecture 9
Sep. 23, 2019

Q&A

Q: Why did we focus mostly on the Perceptron mistake bound for **linearly separable data;** isn't that an unrealistic setting?

A: Not at all! Even if your data isn't linearly separable to begin with, we can often add features to make it so.

x_1	x_2	y
+1	+1	+
+1	-1	-
-1	+1	-
-1	-1	+



Exercise: Add another feature to transform this nonlinearly separable data into linearly separable data.

Reminders

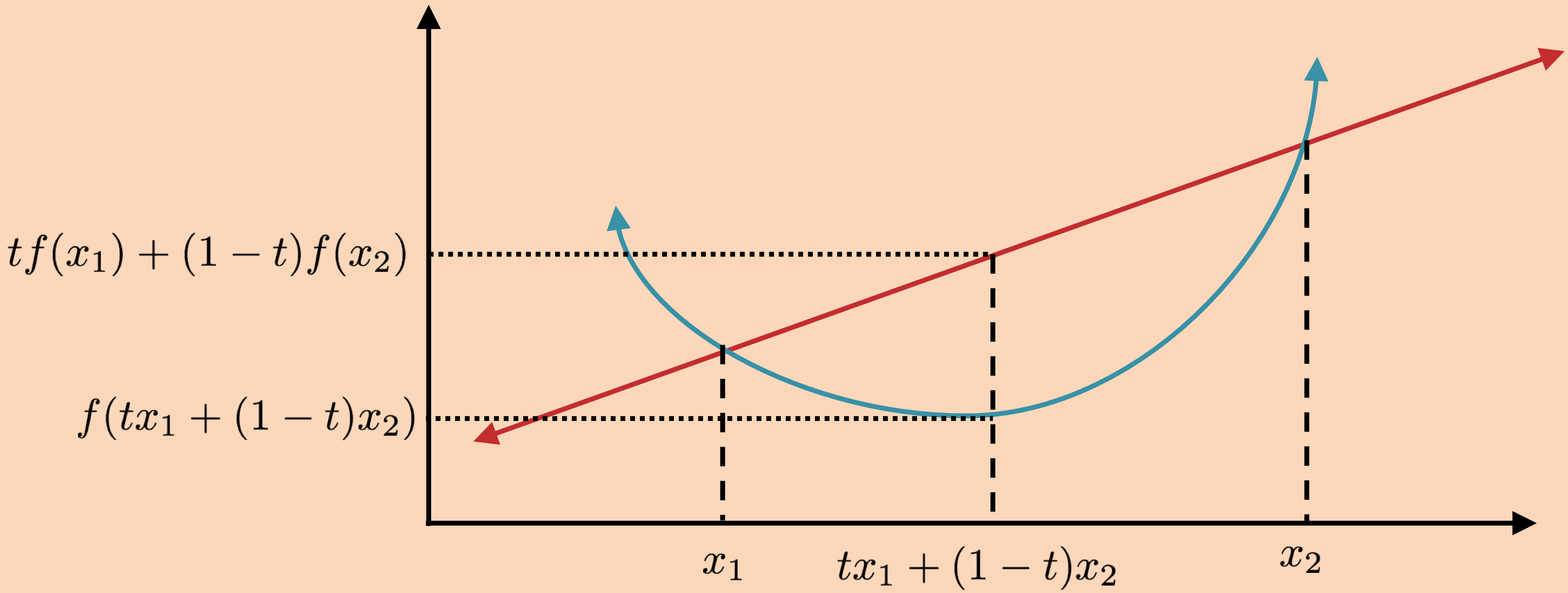
- Homework 3: KNN, Perceptron, Lin.Reg.
  - Out: Wed, Sep. 18
  - Due: Wed, Sep. 25 at 11:59pm
- Midterm Exam 1
  - Thu, Oct. 03, 6:30pm – 8:00pm
- Homework 4: Logistic Regression
  - Out: Wed, Sep. 25
  - Due: Fri, Oct. 11 at 11:59pm
- Today's In-Class Poll
  - http://p9.mlcourse.org

CONVEXITY

Convexity

Function $f : \mathbb{R}^M \rightarrow \mathbb{R}$ is **convex**
if $\forall\ \mathbf{x}_1 \in \mathbb{R}^M, \mathbf{x}_2 \in \mathbb{R}^M, 0 \leq t \leq 1$:

$$f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2) \leq t f(\mathbf{x}_1) + (1-t) f(\mathbf{x}_2)$$

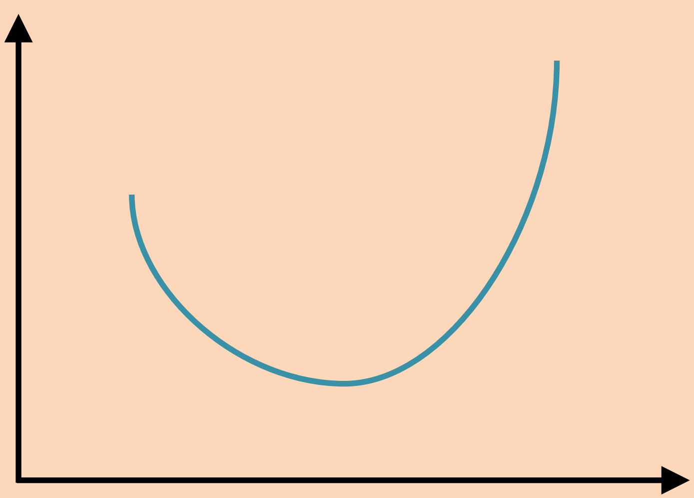


Convexity

Suppose we have a function $f(x) : \mathcal{X} \rightarrow \mathcal{Y}$.

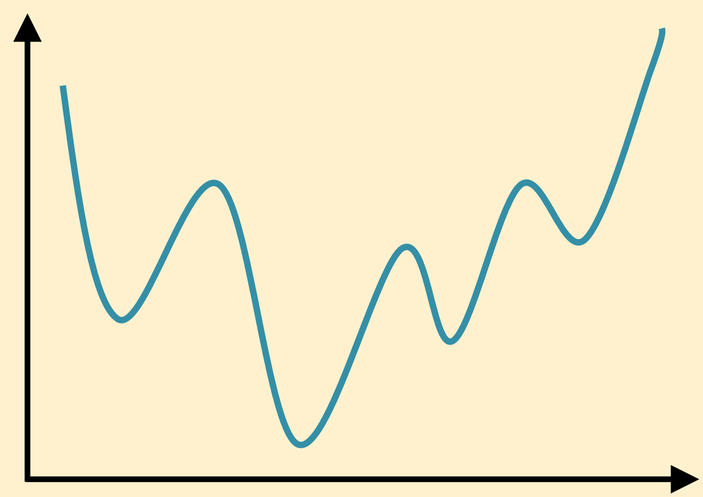
- The value x^* is a **global minimum** of f iff $f(x^*) \leq f(x), \forall x \in \mathcal{X}$.
- The value x^* is a **local minimum** of f iff $\exists \epsilon$ s.t. $f(x^*) \leq f(x), \forall x \in [x^* - \epsilon, x^* + \epsilon]$.

Convex Function



- Each **local minimum** is a **global minimum**

Nonconvex Function

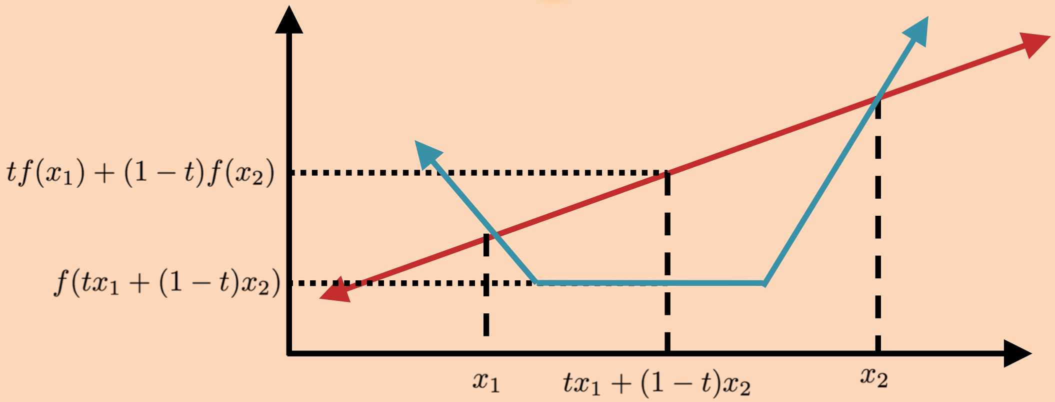


- A *nonconvex* function is **not convex**
- Each **local minimum** is **not** necessarily a **global minimum**

Convexity

Function $f : \mathbb{R}^M \to \mathbb{R}$ is **convex**
if $\forall\ \mathbf{x}_1 \in \mathbb{R}^M, \mathbf{x}_2 \in \mathbb{R}^M, 0 \le t \le 1$:

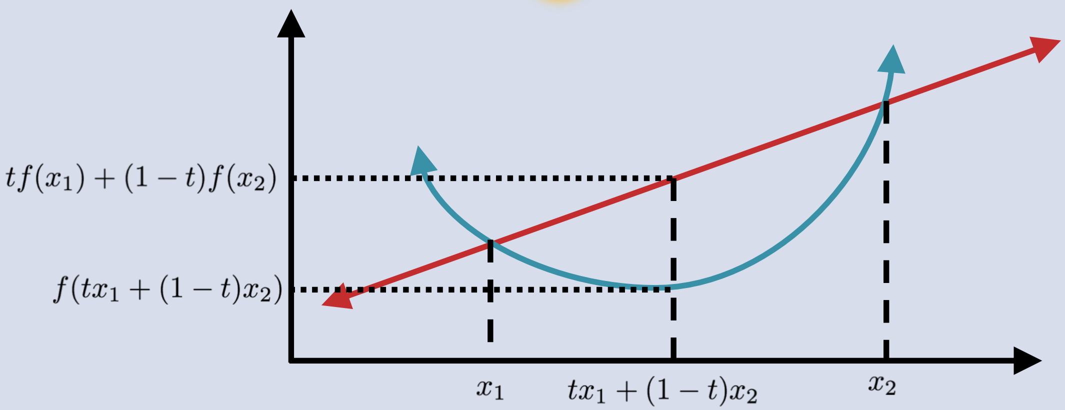
$$f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2) \le tf(\mathbf{x}_1) + (1-t)f(\mathbf{x}_2)$$



Each **local minimum** of a **convex** function is also a **global minimum.**

Function $f : \mathbb{R}^M \to \mathbb{R}$ is **strictly convex**
if $\forall\ \mathbf{x}_1 \in \mathbb{R}^M, \mathbf{x}_2 \in \mathbb{R}^M, 0 \le t \le 1$:

$$f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2) < tf(\mathbf{x}_1) + (1-t)f(\mathbf{x}_2)$$



A **strictly convex** function has a **unique global minimum.**

Convexity

The **Mean Squared Error** function, which we minimize for learning the parameters of Linear Regression, **is convex**!

Solving Linear Regression

Question:

True or False: If Mean Squared Error (i.e. $\frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - h(\mathbf{x}^{(i)}))^2$) has a unique minimizer (i.e. argmin), then Mean Absolute Error (i.e. $\frac{1}{N}\sum_{i=1}^{N}|y^{(i)} - h(\mathbf{x}^{(i)})|$) must also have a unique minimizer.

Answer:

GRADIENT DESCENT

Motivation: Gradient Descent

To solve the Ordinary Least Squares problem we compute:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(y^{(i)} - (\boldsymbol{\theta}^T\mathbf{x}^{(i)}))^2$$

$$= (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{Y})$$

The resulting shape of the matrices:

$$\underbrace{(\underbrace{\mathbf{X}^T}_{M\times N}\ \underbrace{\mathbf{X}}_{N\times M})^{-1}}_{M\times M}\ \underbrace{(\underbrace{\mathbf{X}^T}_{M\times N}\ \underbrace{\mathbf{Y}}_{N\times 1})}_{M\times 1}$$

Background: Matrix Multiplication Given matrices $\mathbf{A}$ and $\mathbf{B}$

- If $\mathbf{A}$ is $q \times r$ and $\mathbf{B}$ is $r \times s$, computing $\mathbf{AB}$ takes $O(qrs)$
- If $\mathbf{A}$ and $\mathbf{B}$ are $q \times q$, computing $\mathbf{AB}$ takes $O(q^{2.373})$
- If $\mathbf{A}$ is $q \times q$, computing A^{-1} takes $O(q^{2.373})$.

Computational Complexity of OLS:

$\mathbf{X}^T\mathbf{X}$	$O(M^2N)$
$(\quad)^{-1}$	$O(M^{2.373})$
$\mathbf{X}^T\mathbf{Y}$	$O(MN)$
$(\quad)^{-1}(\quad)$	$O(M^2)$
total	$O(M^2N + M^{2.373})$

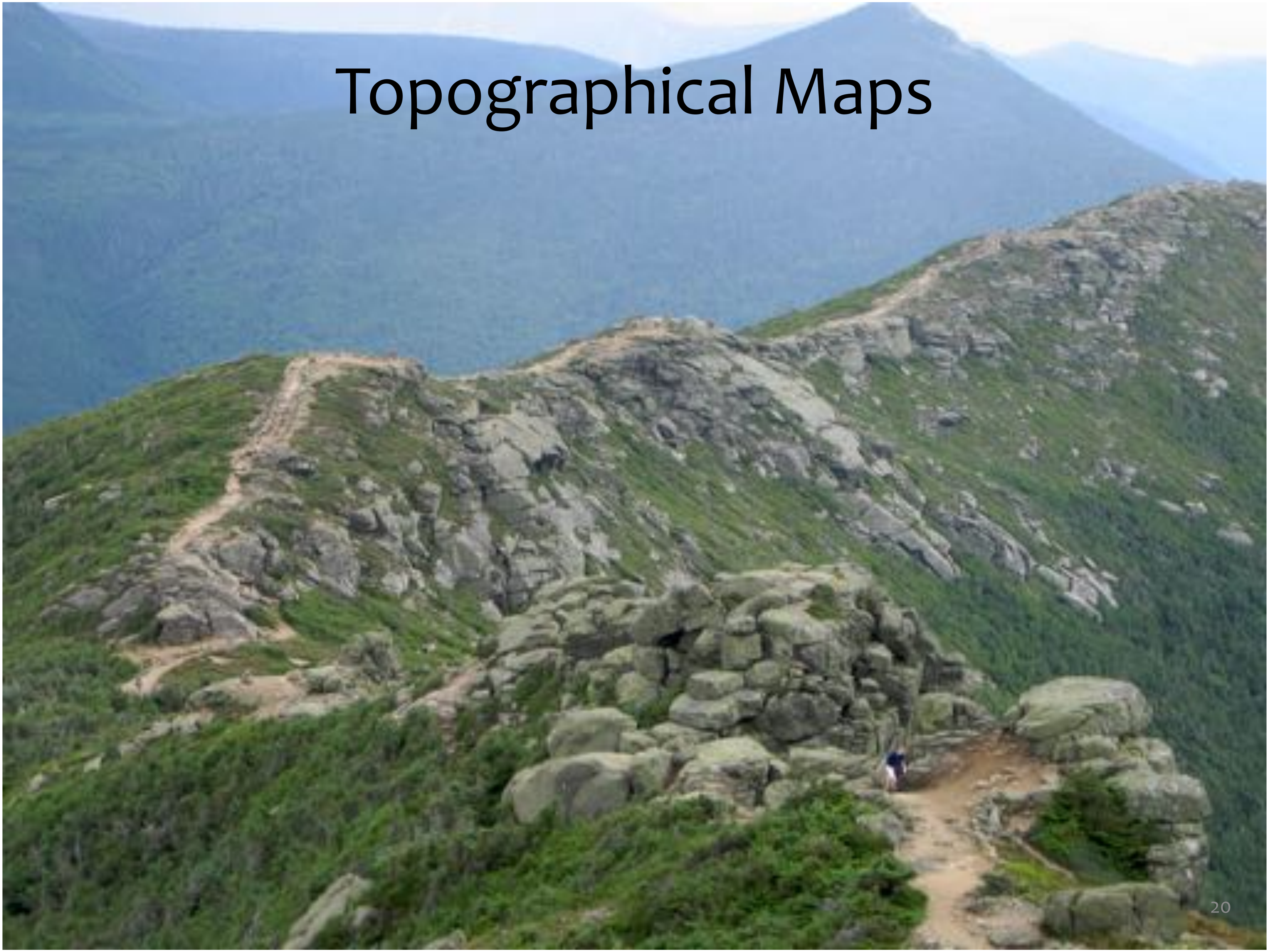
Linear in # of examples, N
Polynomial in # of features, M

Motivation: Gradient Descent

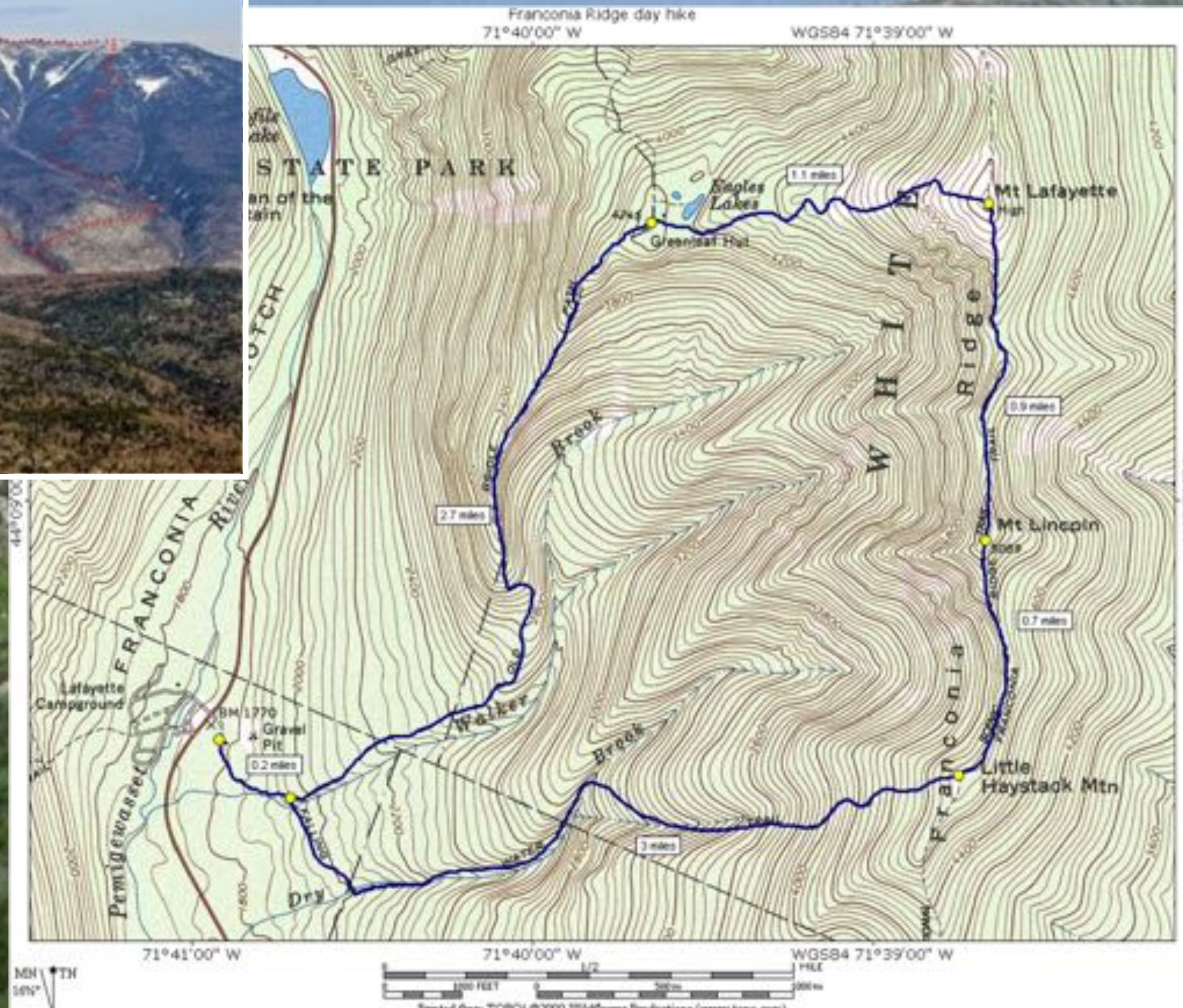
Cases to consider gradient descent:

1. What if we **can not** find a closed-form solution?
2. What if we **can**, but it’s inefficient to compute?
3. What if we **can**, but it’s numerically unstable to compute?

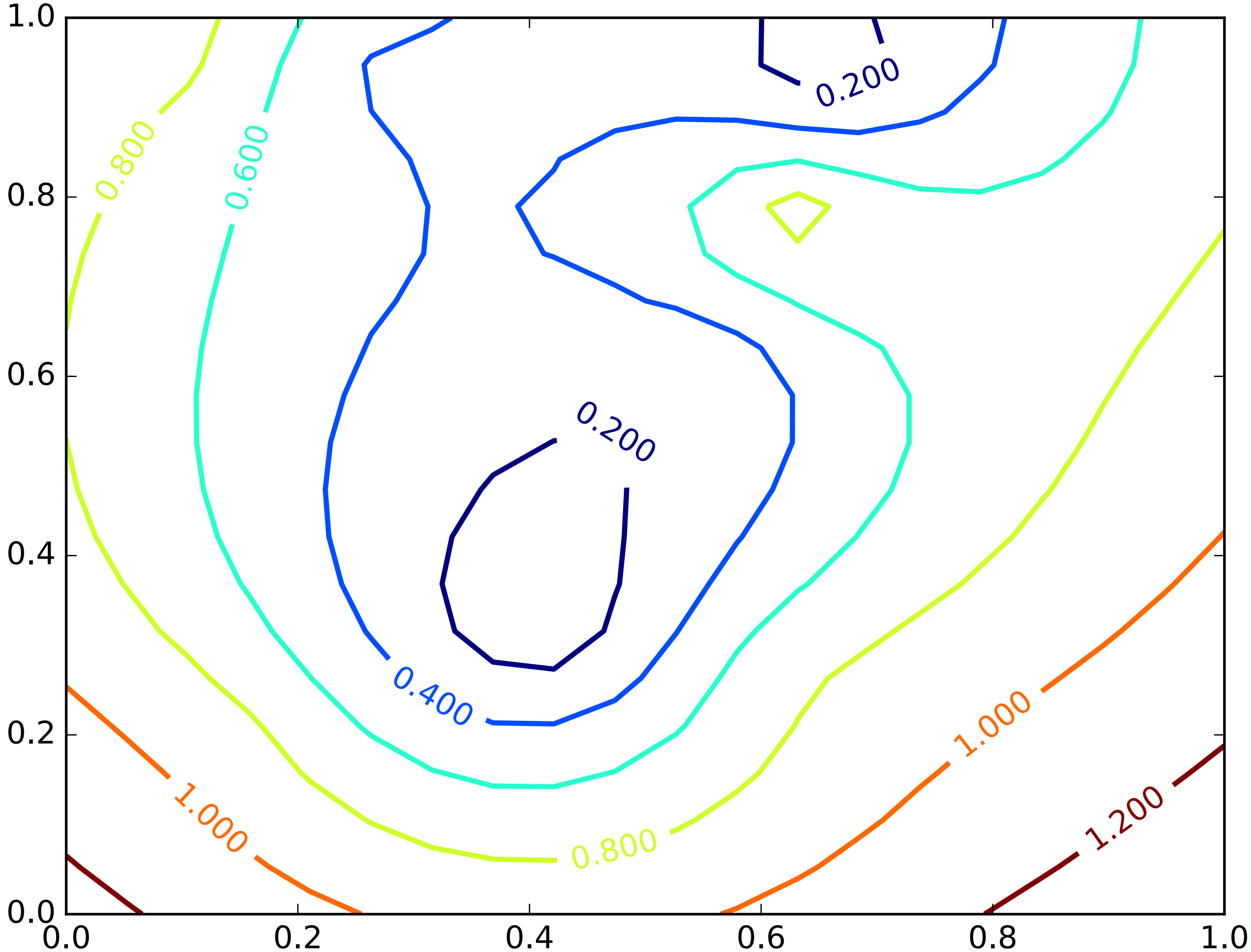
Topographical Maps



Topographical Maps

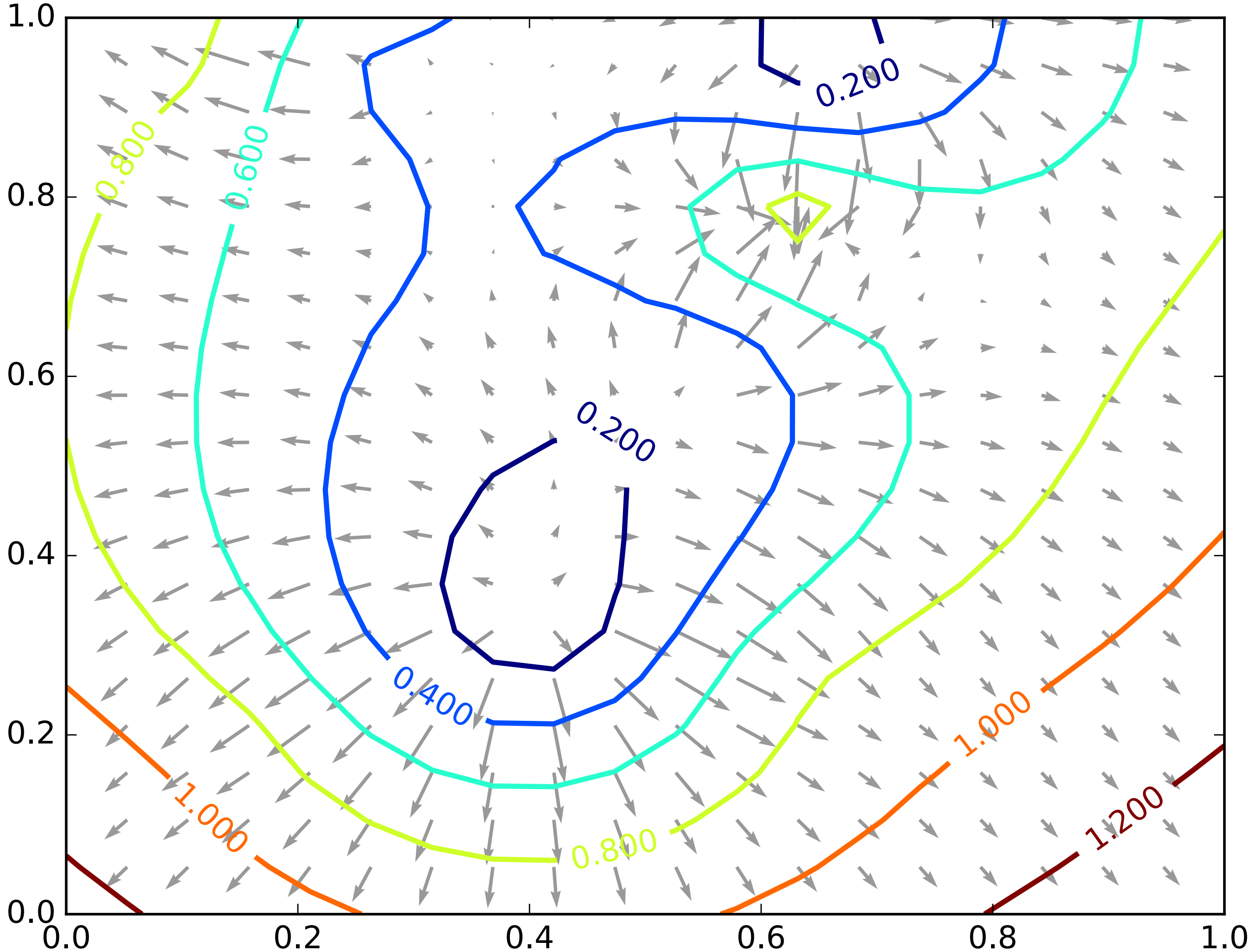


Gradients



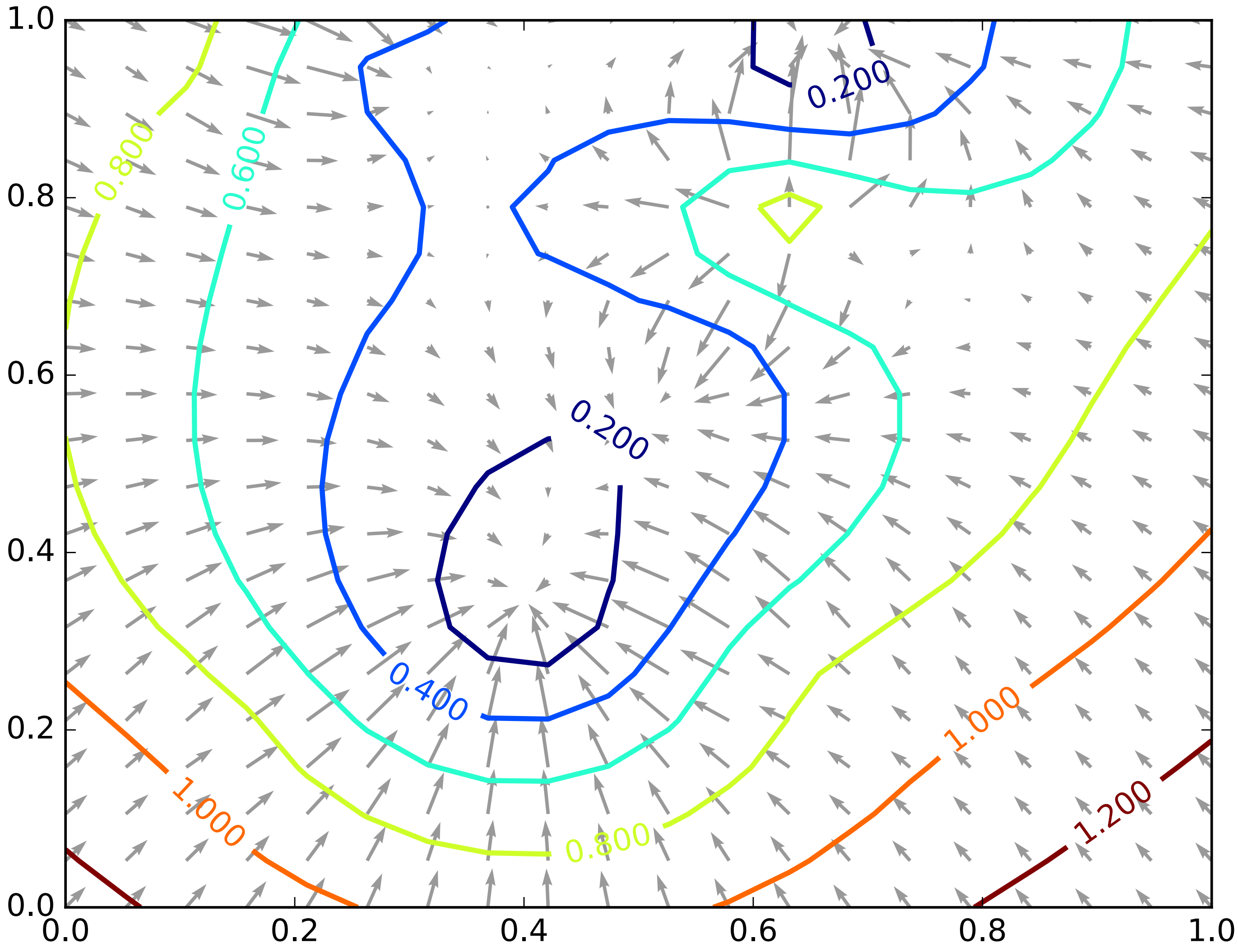
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
0.200
0.200
0.400
0.600
0.800
0.800
1.000
1.000
1.200

Gradients



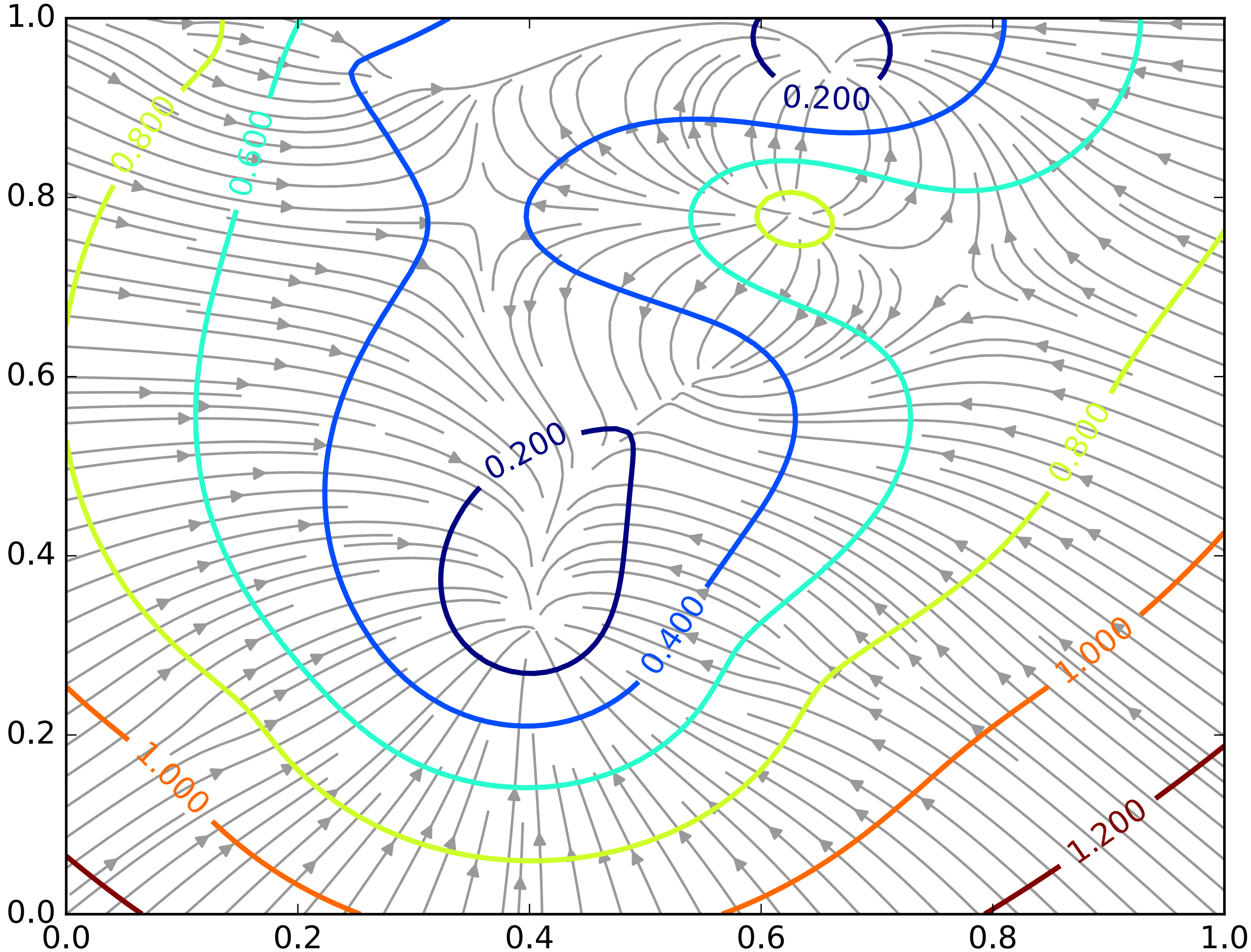
These are the **gradients** that
Gradient **Ascent** would follow.

(Negative) Gradients



These are the **negative** gradients that Gradient **Descent** would follow.

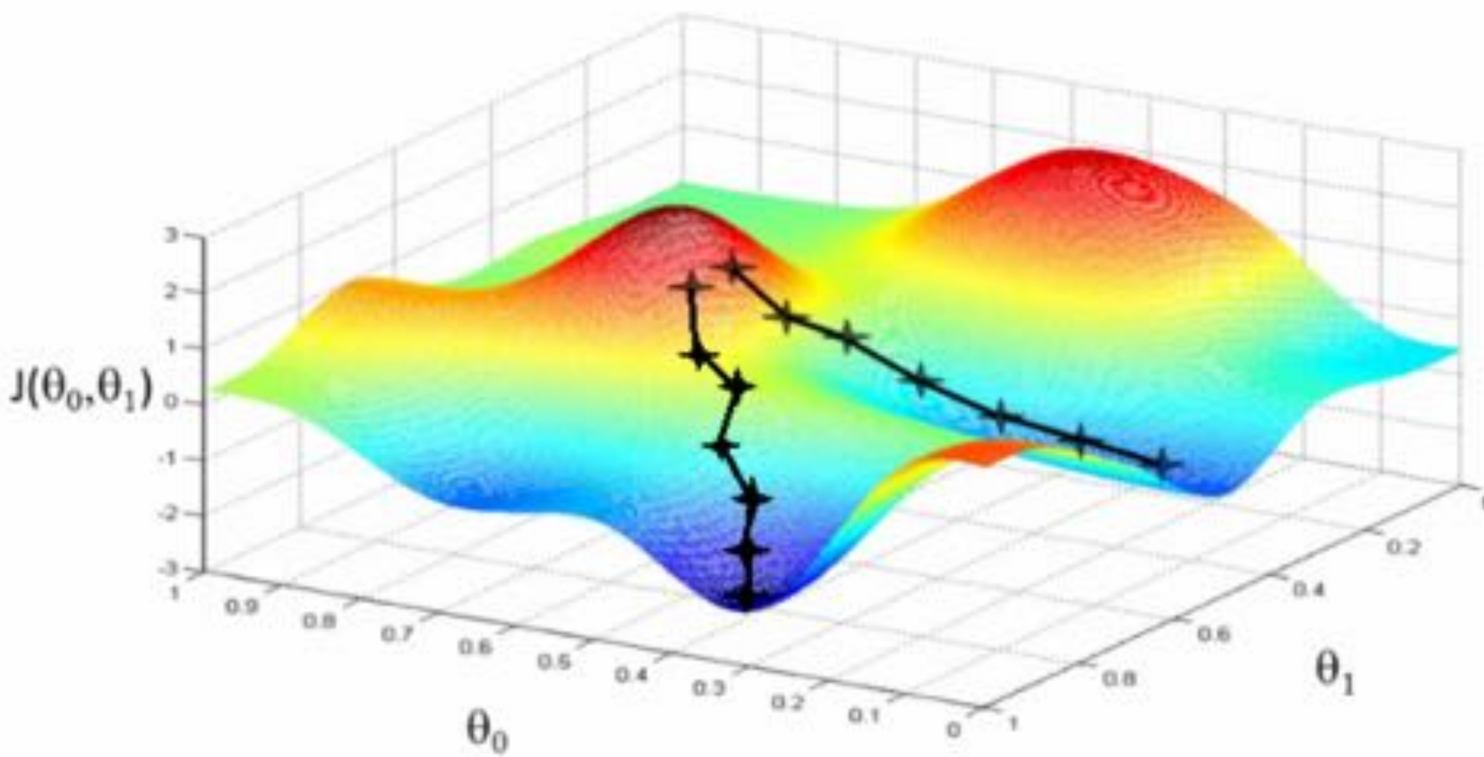
(Negative) Gradient *Paths*



Shown are the **paths** that Gradient Descent would follow if it were making **infinitesimally small steps.**

Pros and cons of gradient descent

- Simple and often quite effective on ML tasks
- Often very scalable
- Only applies to smooth functions (differentiable)
- Might find a local minimum, rather than a global one



Slide courtesy of William Cohen

Gradient Descent

Chalkboard

- Gradient Descent Algorithm
- Details: starting point, stopping criterion, line search

Gradient Descent

Algorithm 1 Gradient Descent

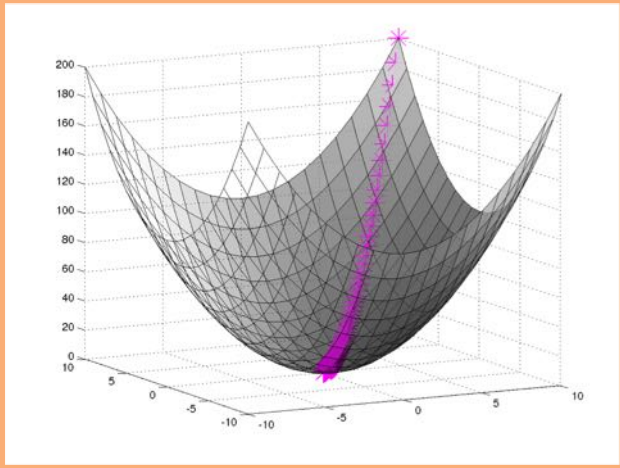
1: **procedure** GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)

2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$

3: $\quad$**while** not converged **do**

4: $\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

5: $\quad$**return** $\boldsymbol{\theta}$



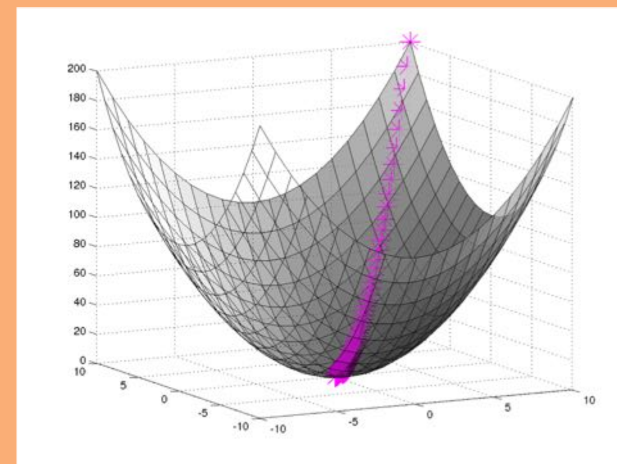
In order to apply GD to Linear Regression all we need is the **gradient** of the objective function (i.e. vector of partial derivatives).

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J(\boldsymbol{\theta}) \end{bmatrix}$$

Gradient Descent

Algorithm 1 Gradient Descent

> **procedure** GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
> $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
>   **while** not converged **do**
> $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
>   **return** $\boldsymbol{\theta}$



There are many possible ways to detect **convergence**. For example, we could check whether the L2 norm of the gradient is below some small tolerance.

$$||\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})||_2 \leq \epsilon$$

Alternatively we could check that the reduction in the objective function from one iteration to the next is small.

STOCHASTIC GRADIENT DESCENT

Gradient Descent

Algorithm 1 Gradient Descent

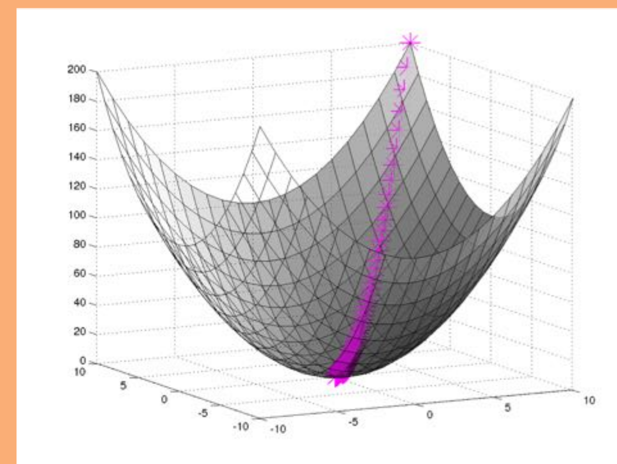
procedure GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$

while not converged **do**

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

return $\boldsymbol{\theta}$

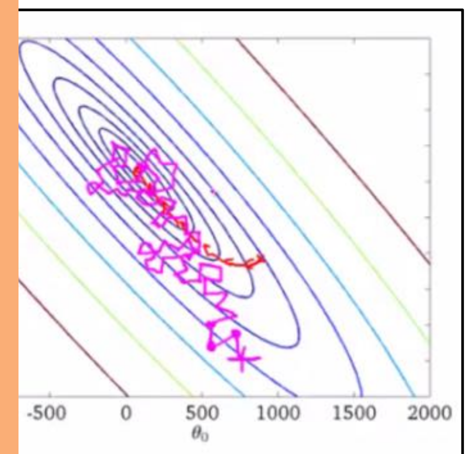


M

Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

procedure SGD($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
    while not converged **do**
        for $i \sim$ Uniform($\{1, 2, \ldots, N\}$) **do**
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
    return $\boldsymbol{\theta}$



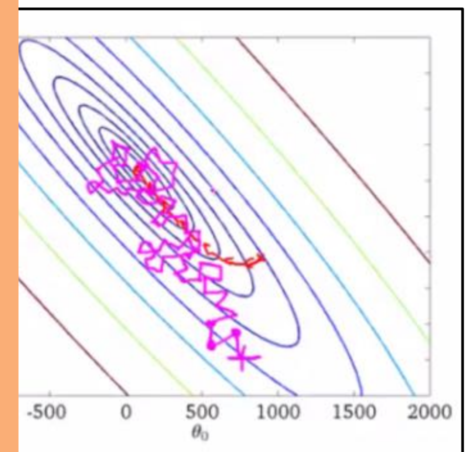
We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

procedure SGD($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
  while not converged **do**
    for $i \sim$ Uniform($\{1, 2, \ldots, N\}$) **do**
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
  return $\boldsymbol{\theta}$



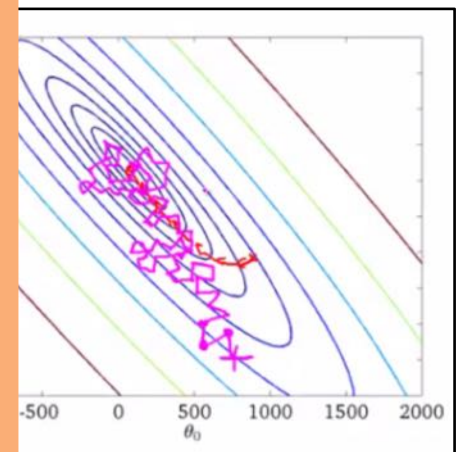
We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

procedure SGD($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
    while not converged **do**
        for $i \in$ shuffle($\{1, 2, \ldots, N\}$) **do**
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
    return $\boldsymbol{\theta}$

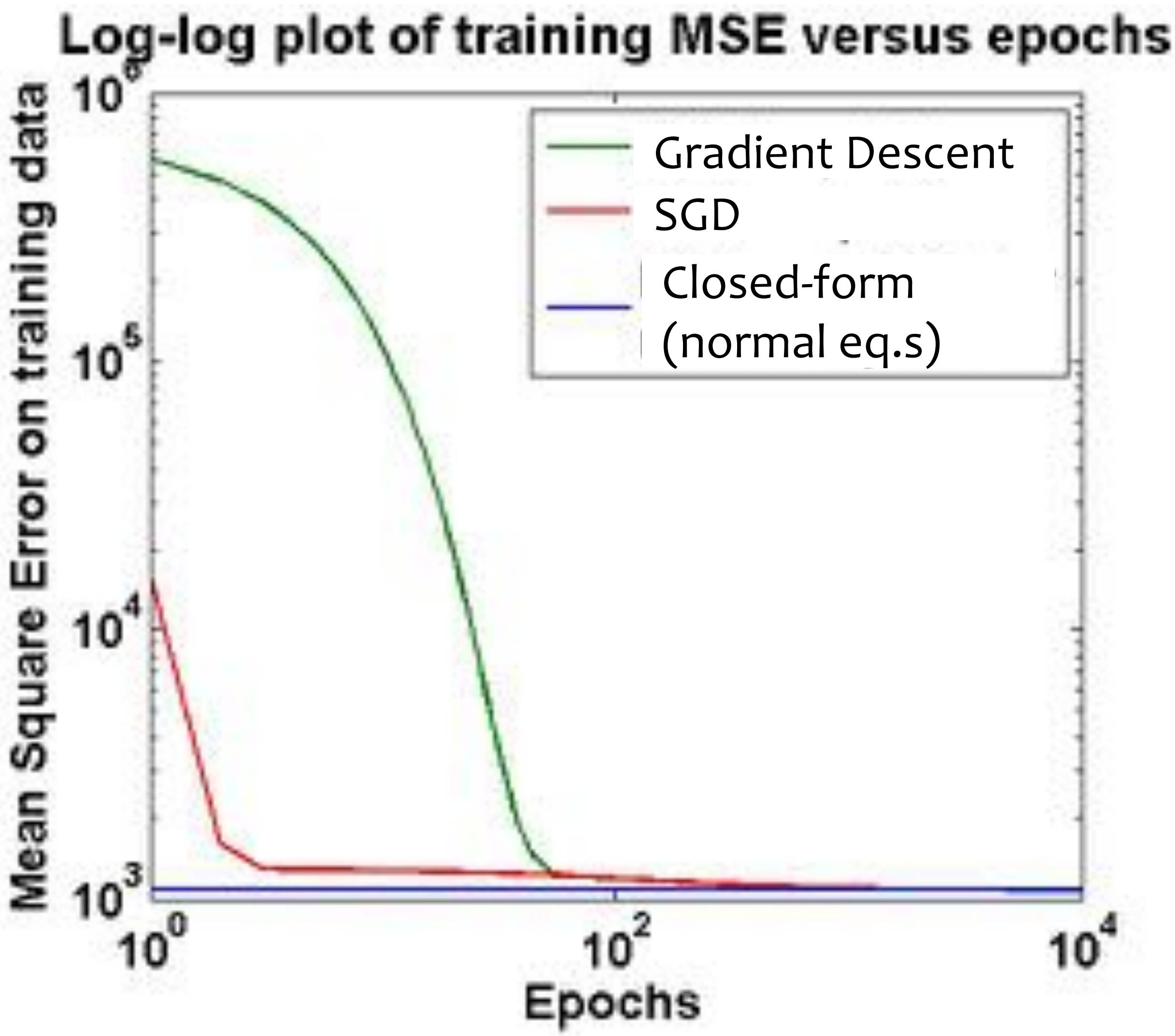


In practice, it is common to implement SGD using sampling **without** replacement (i.e. shuffle({1,2,… N}), even though most of the theory is for sampling **with** replacement (i.e. Uniform({1,2,… N}).

We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

Convergence Curves

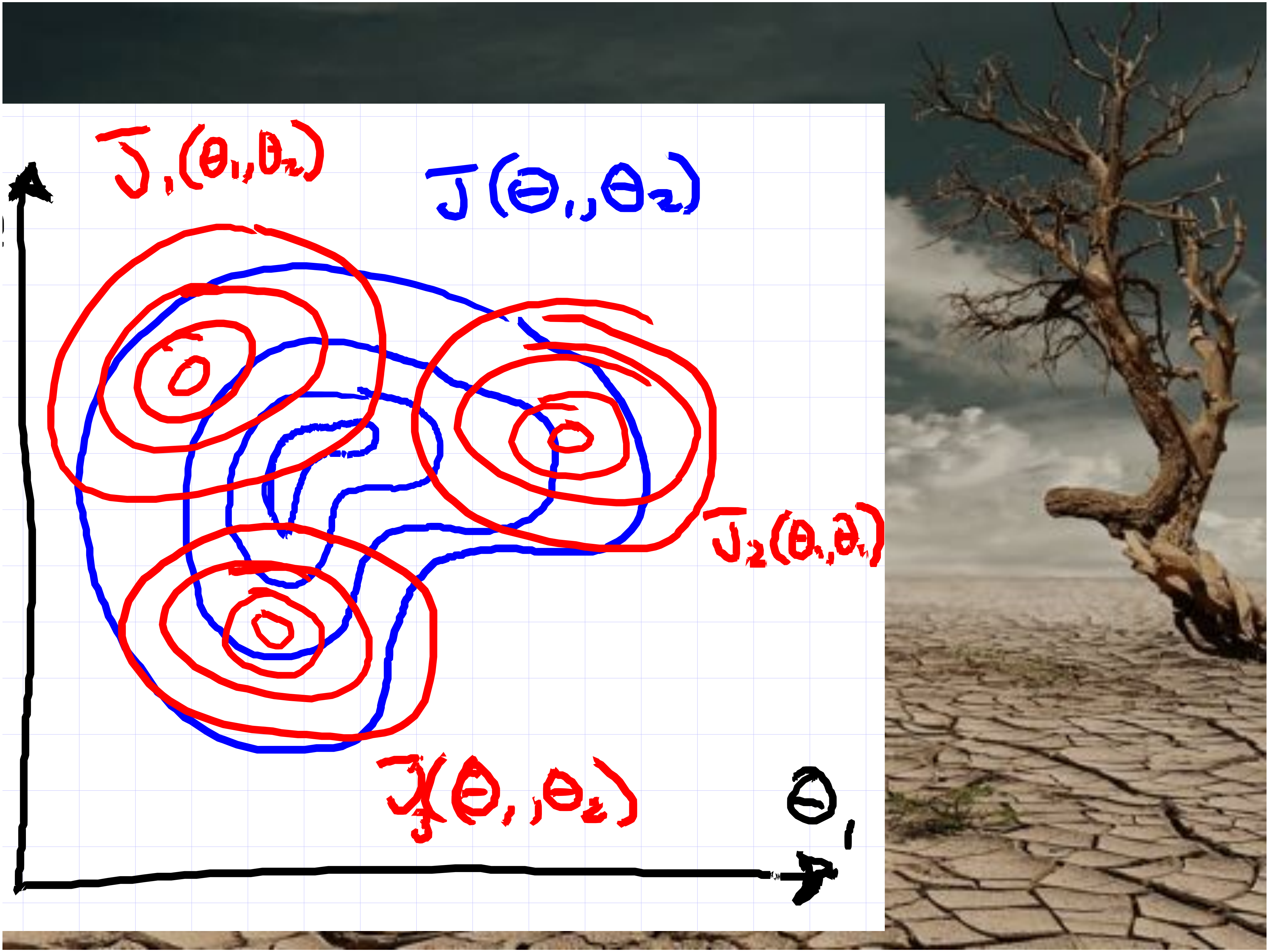


- *Def*: an **epoch** is a single pass through the training data

1. For GD, only **one update** per epoch
2. For SGD, ***N* updates** per epoch
   N = (# train examples)

- SGD reduces MSE much more rapidly than GD
- For GD / SGD, training MSE is initially large due to uninformed initialization

Figure adapted from Eric P. Xing



J1(θ1,θ2)
J(θ1,θ2)
J2(θ1,θ2)
J3(θ1,θ2)
θ1

Expectations of Gradients

$$\frac{dJ(\vec{\theta})}{d\theta_j} = \frac{1}{N}\sum_{i=1}^{N}\frac{d}{d\theta_j}\left(J_i(\vec{\theta})\right)$$

$$\nabla J(\vec{\theta}) = \begin{bmatrix} \vdots \\ \text{—} \\ \vdots \end{bmatrix} = \frac{1}{N}\sum_{i=1}^{N}\nabla J_i(\vec{\theta})$$

j^{th}

Recall: for any discrete r.v. X

$$E_X[f(X)] \triangleq \sum_x P(X=x) f(x)$$

Q: What is the expected value of a randomly chosen $\nabla J_i(\theta)$?

Let $I \sim \text{Uniform}(\{1,\ldots,N\})$

$\Rightarrow P(I=i) = \frac{1}{N}$ if $i \in \{1 \ldots N\}$

$$E_I[\nabla J_I(\vec{\theta})] = \sum_{i=1}^{N} P(I=i)\nabla J_i(\vec{\theta})$$

$$= \frac{1}{N}\sum_{i=1}^{N}\nabla J_i(\vec{\theta})$$

$$= \nabla J(\vec{\theta})$$

Convergence of Optimizers

Convergence Analysis:

Def: convergence is when $J(\vec{\theta}) - J(\vec{\theta}^*) < \epsilon$ ← true unknown min

not covered → Newton's Method

Methods	Steps to Converge	Computation per iteration
Newton's Method	$O(\ln \ln 1/\epsilon)$	$\nabla J(\theta)$ $\nabla^2 J(\theta)$ ← $O(NM^2)$
GD	$O(\ln 1/\epsilon)$	$\nabla J(\theta)$ ← $O(NM)$
SGD	$O(1/\epsilon)$ *	$\nabla J_i(\theta)$ ← $O(M)$

* "almost sure" convergence

lots of caveats and conditions

way less computation

Takeaway: SGD has much slower asymptotic convergence but is often faster in practice.

Optimization Objectives

You should be able to…

- Apply gradient descent to optimize a function
- Apply stochastic gradient descent (SGD) to optimize a function
- Apply knowledge of zero derivatives to identify a closed-form solution (if one exists) to an optimization problem
- Distinguish between convex, concave, and nonconvex functions
- Obtain the gradient (and Hessian) of a (twice) differentiable function

PROBABILISTIC LEARNING

Probabilistic Learning

Function Approximation

Previously, we assumed that our output was generated using a **deterministic target function:**

$$\mathbf{x}^{(i)} \sim p^*(\cdot)$$

$$y^{(i)} = c^*(\mathbf{x}^{(i)})$$

Our goal was to learn a hypothesis h(**x**) that best approximates c^*(**x**)

Probabilistic Learning

Today, we assume that our output is **sampled** from a conditional **probability distribution:**

$$\mathbf{x}^{(i)} \sim p^*(\cdot)$$

$$y^{(i)} \sim p^*(\cdot|\mathbf{x}^{(i)})$$

Our goal is to learn a probability distribution p(y|**x**) that best approximates p^*(y|**x**)

Robotic Farming

	Deterministic	**Probabilistic**
Classification (binary output)	Is this a picture of a wheat kernel?	Is this plant drought resistant?
Regression (continuous output)	How many wheat kernels are in this picture?	What will the yield of this plant be?

Bayes Optimal Classifier

Whiteboard

- Bayes Optimal Classifier
- Reducible / irreducible error
- Ex: Bayes Optimal Classifier for 0/1 Loss

Maximum Likelihood Estimation

The principle of Maximum likelihood estimation (MLE):

Choose parameters that make the data "most likely".

Assumptions: Data generated iid from distribution $p^*(x|\vec{\theta}^*)$ and comes from a family of distr. parameterized $\theta \in \Theta$ ← set of possible parameters

Formally:

$$\theta_{MLE} = \underset{\theta \in \Theta}{\operatorname{argmax}}\; p(D|\theta)$$

$$= \underset{\theta \in \Theta}{\operatorname{argmax}}\; \log p(D|\theta)$$

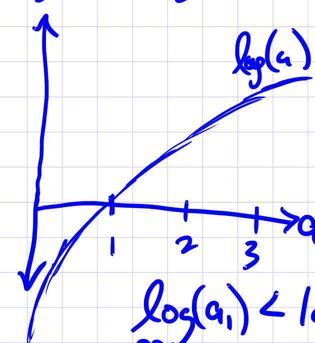
$$= \underset{\theta \in \Theta}{\operatorname{argmax}}\; \ell(\theta)$$

usually a continuous optimization

where $\ell(\theta) \triangleq \log p(D|\theta)$ "log-likelihood"

treat as function of θ where D is constant

since log is monotonic



$\log(a_1) < \log(a_2)$ iff $a_1 < a_2$

$\Rightarrow \log(f(a_1)) < \log(f(a_2))$ iff $f(a_1) < f(a_2)$

Learning from Data (Frequentist)

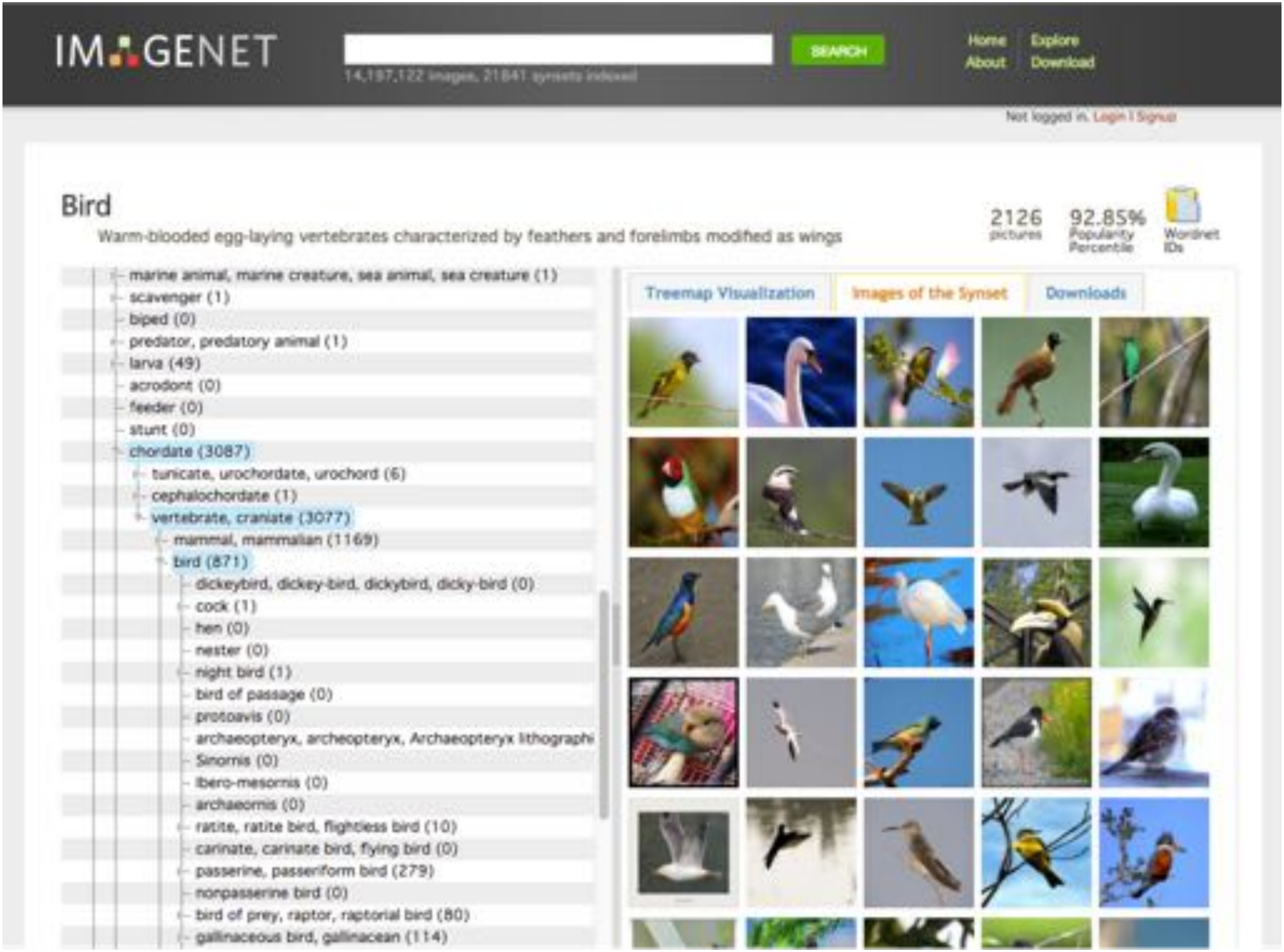
Whiteboard

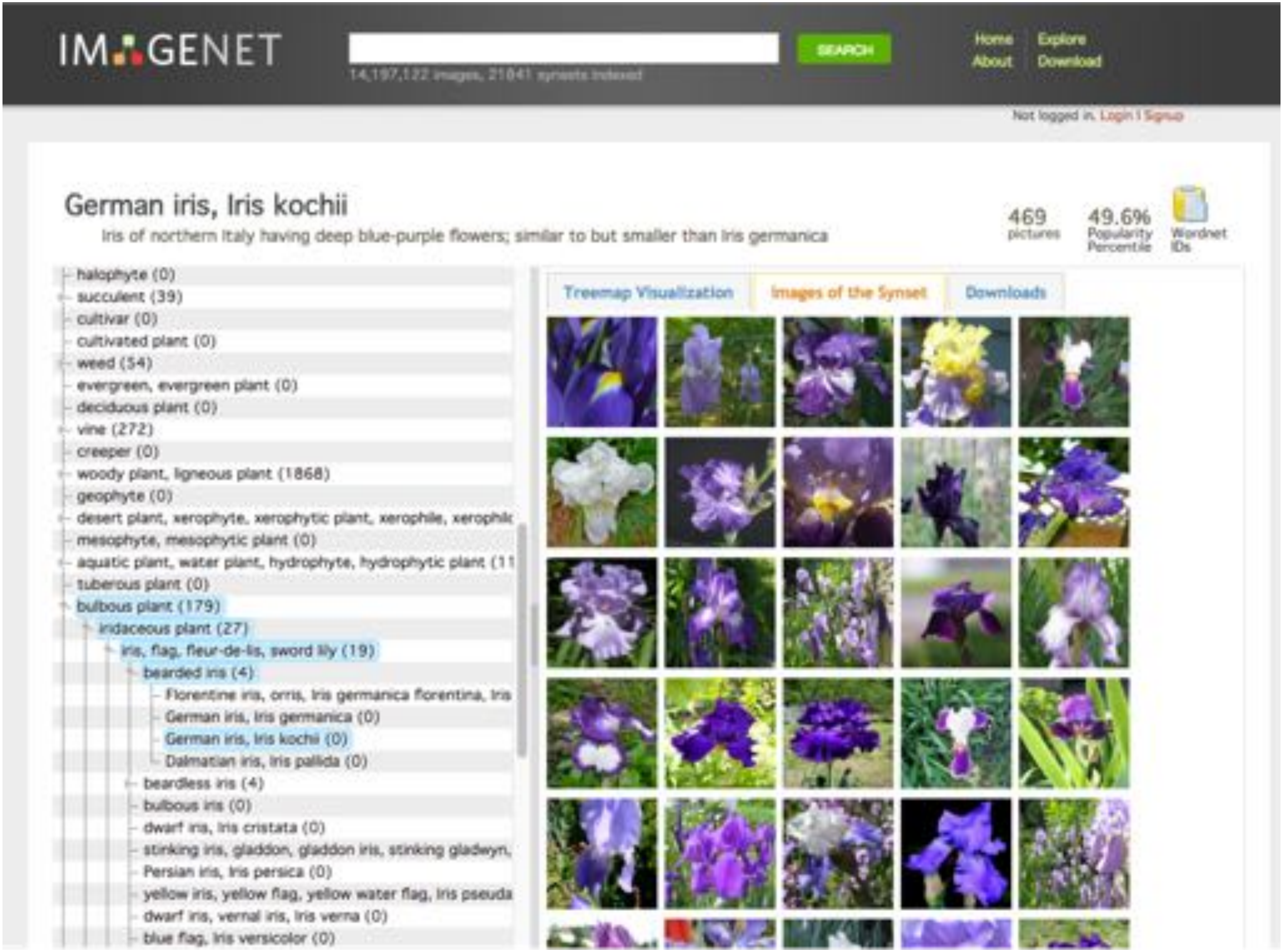
- Principle of Maximum Likelihood Estimation (MLE)
- Strawmen:
  - Example: Bernoulli
  - Example: Gaussian
  - Example: Conditional #1
    (Bernoulli conditioned on Gaussian)
  - Example: Conditional #2
    (Gaussians conditioned on Bernoulli)

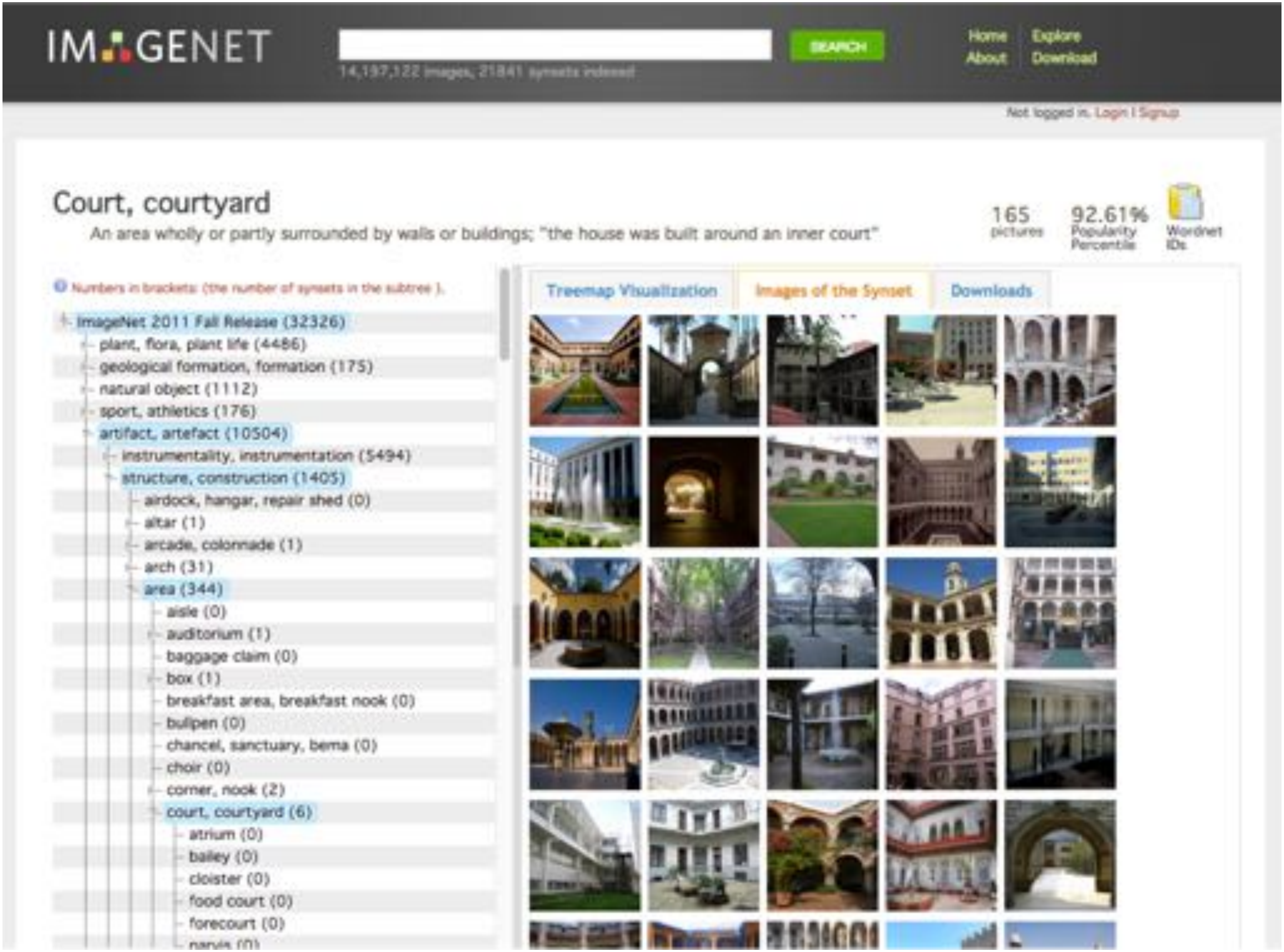
MOTIVATION:
LOGISTIC REGRESSION

Example: Image Classification

- ImageNet LSVRC-2010 contest:
  - **Dataset:** 1.2 million labeled images, 1000 classes
  - **Task**: Given a new image, label it with the correct class
  - **Multiclass** classification problem
- Examples from http://image-net.org/


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login I Signup
Bird
Warm-blooded egg-laying vertebrates characterized by feathers and forelimbs modified as wings
2126
pictures
92.85%
Popularity Percentile
Wordnet IDs
marine animal, marine creature, sea animal, sea creature (1)
scavenger (1)
biped (0)
predator, predatory animal (1)
larva (49)
acrodont (0)
feeder (0)
stunt (0)
chordate (3087)
tunicate, urochordate, urochord (6)
cephalochordate (1)
vertebrate, craniate (3077)
mammal, mammalian (1169)
bird (871)
dickeybird, dickey-bird, dickybird, dicky-bird (0)
cock (1)
hen (0)
nester (0)
night bird (1)
bird of passage (0)
protoavis (0)
archaeopteryx, archeopteryx, Archaeopteryx lithographi
Sinornis (0)
Ibero-mesornis (0)
archaeornis (0)
ratite, ratite bird, flightless bird (10)
carinate, carinate bird, flying bird (0)
passerine, passeriform bird (279)
nonpasserine bird (0)
bird of prey, raptor, raptorial bird (80)
gallinaceous bird, gallinacean (114)
Treemap Visualization
Images of the Synset
Downloads


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login I Signup
German iris, Iris kochii
Iris of northern Italy having deep blue-purple flowers; similar to but smaller than Iris germanica
469
pictures
49.6%
Popularity Percentile
Wordnet IDs
halophyte (0)
succulent (39)
cultivar (0)
cultivated plant (0)
weed (54)
evergreen, evergreen plant (0)
deciduous plant (0)
vine (272)
creeper (0)
woody plant, ligneous plant (1868)
geophyte (0)
desert plant, xerophyte, xerophytic plant, xerophile, xerophic
mesophyte, mesophytic plant (0)
aquatic plant, water plant, hydrophyte, hydrophytic plant (11
tuberous plant (0)
bulbous plant (179)
iridaceous plant (27)
iris, flag, fleur-de-lis, sword lily (19)
bearded iris (4)
Florentine iris, orris, Iris germanica florentina, Iris
German iris, Iris germanica (0)
German iris, Iris kochii (0)
Dalmatian iris, Iris pallida (0)
beardless iris (4)
bulbous iris (0)
dwarf iris, Iris cristata (0)
stinking iris, gladdon, gladdon iris, stinking gladwyn,
Persian iris, Iris persica (0)
yellow iris, yellow flag, yellow water flag, Iris pseuda
dwarf iris, vernal iris, Iris verna (0)
blue flag, Iris versicolor (0)
Treemap Visualization
Images of the Synset
Downloads


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
Explore
About
Download
Not logged in. Login | Signup
Court, courtyard
An area wholly or partly surrounded by walls or buildings; "the house was built around an inner court"
165 pictures
92.61% Popularity Percentile
Wordnet IDs
Numbers in brackets: (the number of synsets in the subtree).
ImageNet 2011 Fall Release (32326)
plant, flora, plant life (4486)
geological formation, formation (175)
natural object (1112)
sport, athletics (176)
artifact, artefact (10504)
instrumentality, instrumentation (5494)
structure, construction (1405)
airdock, hangar, repair shed (0)
altar (1)
arcade, colonnade (1)
arch (31)
area (344)
aisle (0)
auditorium (1)
baggage claim (0)
box (1)
breakfast area, breakfast nook (0)
bullpen (0)
chancel, sanctuary, bema (0)
choir (0)
corner, nook (2)
court, courtyard (6)
atrium (0)
bailey (0)
cloister (0)
food court (0)
forecourt (0)
Treemap Visualization
Images of the Synset
Downloads

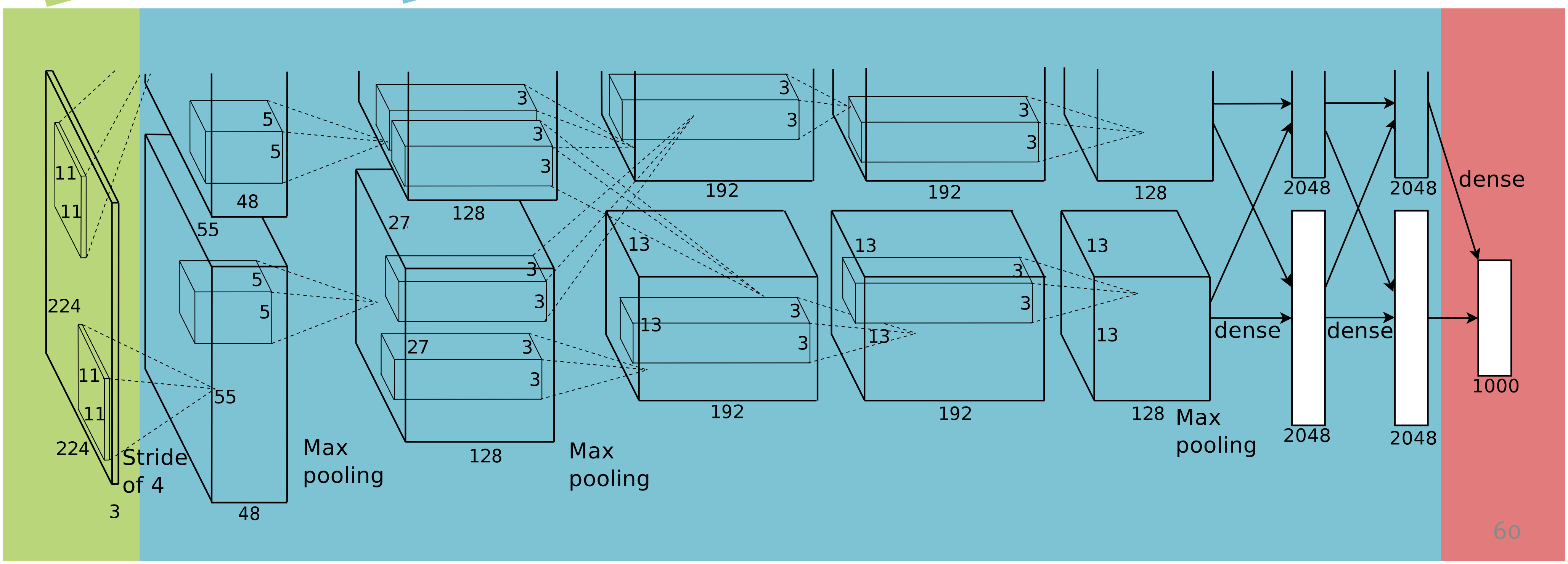
Example: Image Classification

CNN for Image Classification
(Krizhevsky, Sutskever & Hinton, 2011)
17.5% error on ImageNet LSVRC-2010 contest

Input image (pixels)

- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax



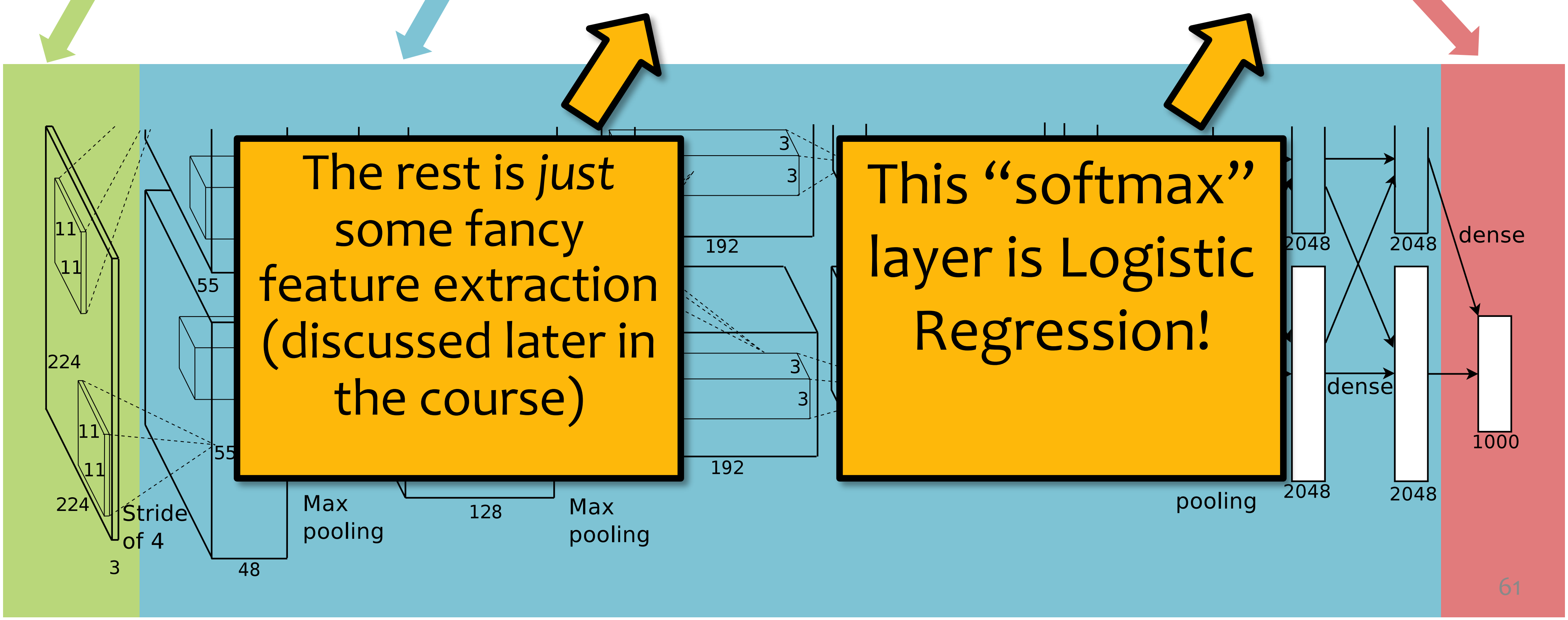
Example: Image Classification

CNN for Image Classification
(Krizhevsky, Sutskever & Hinton, 2011)
17.5% error on ImageNet LSVRC-2010 contest

Input image (pixels)

- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax



LOGISTIC REGRESSION

Logistic Regression

Data: Inputs are continuous vectors of length M. Outputs are discrete.

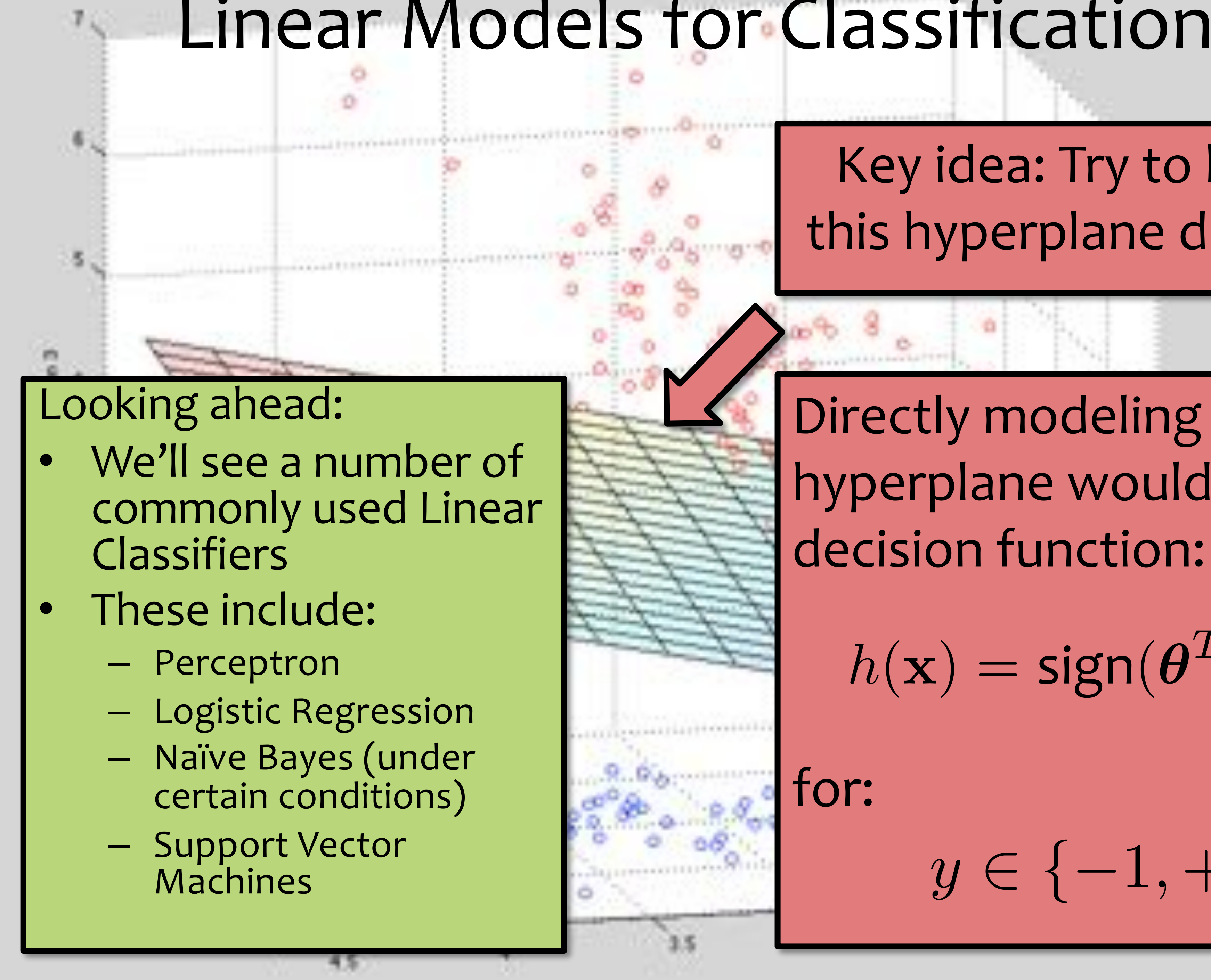
$$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N} \text{ where } \mathbf{x} \in \mathbb{R}^M \text{ and } y \in \{0, 1\}$$

We are back to classification.

Despite the name logistic **regression.**

Linear Models for Classification

Recall...



Key idea: Try to learn this hyperplane directly

Looking ahead:

- We'll see a number of commonly used Linear Classifiers
- These include:
  - Perceptron
  - Logistic Regression
  - Naïve Bayes (under certain conditions)
  - Support Vector Machines

Directly modeling the hyperplane would use a decision function:

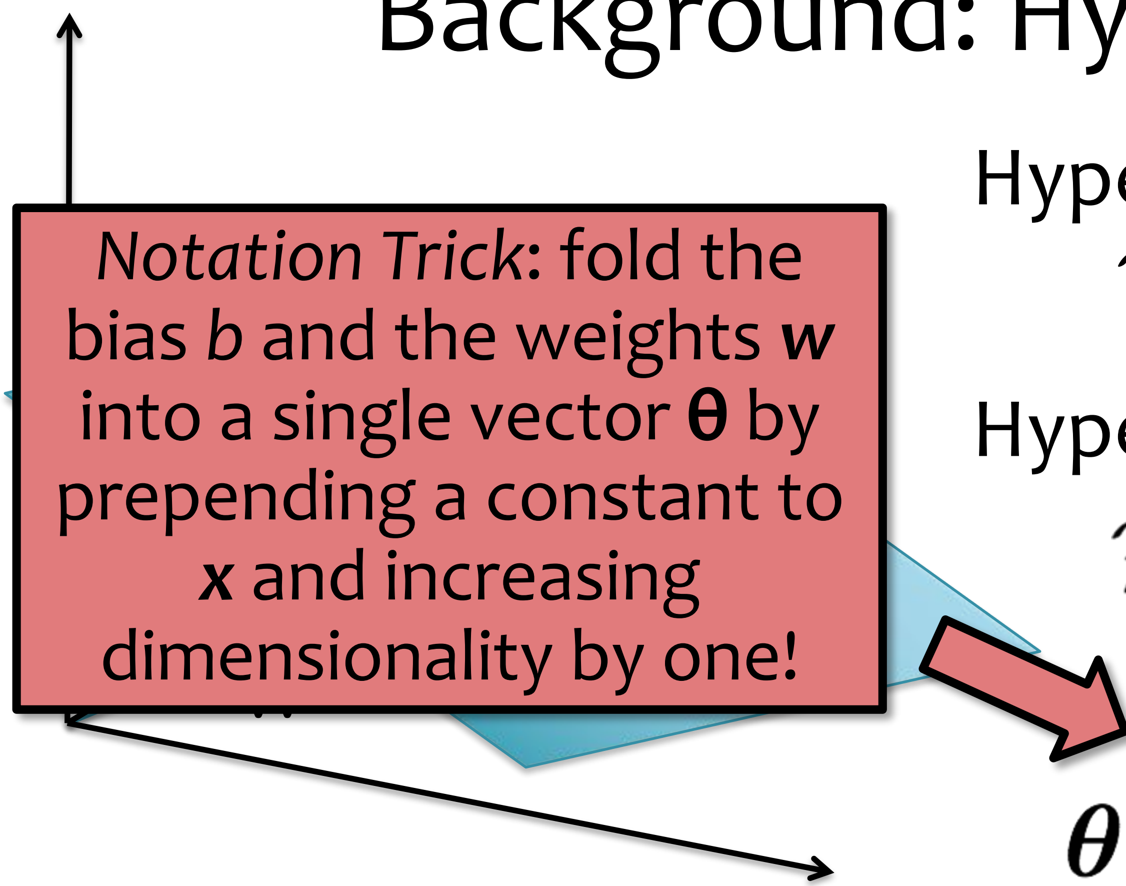
$$h(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$

for:

$$y \in \{-1, +1\}$$

Recall…

Background: Hyperplanes



Notation Trick: fold the bias *b* and the weights ***w*** into a single vector **θ** by prepending a constant to ***x*** and increasing dimensionality by one!

Hyperplane (Definition 1):

$$\mathcal{H} = \{\mathbf{x} : \mathbf{w}^T \mathbf{x} = b\}$$

Hyperplane (Definition 2):

$$\mathcal{H} = \{\mathbf{x} : \boldsymbol{\theta}^T \mathbf{x} = 0$$

$$\text{and } x_0 = 1\}$$

$$\boldsymbol{\theta} = [b, w_1, \ldots, w_M]^T$$

Half-spaces:

$$\mathcal{H}^+ = \{\mathbf{x} : \boldsymbol{\theta}^T \mathbf{x} > 0 \text{ and } x_0 = 1\}$$

$$\mathcal{H}^- = \{\mathbf{x} : \boldsymbol{\theta}^T \mathbf{x} < 0 \text{ and } x_0 = 1\}$$

Using gradient ascent for linear classifiers

Key idea behind today's lecture:

1. Define a linear classifier (logistic regression)
2. Define an objective function (likelihood)
3. Optimize it with gradient descent to learn parameters
4. Predict the class with highest probability under the model