10-301/10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Stochastic Gradient Descent

+

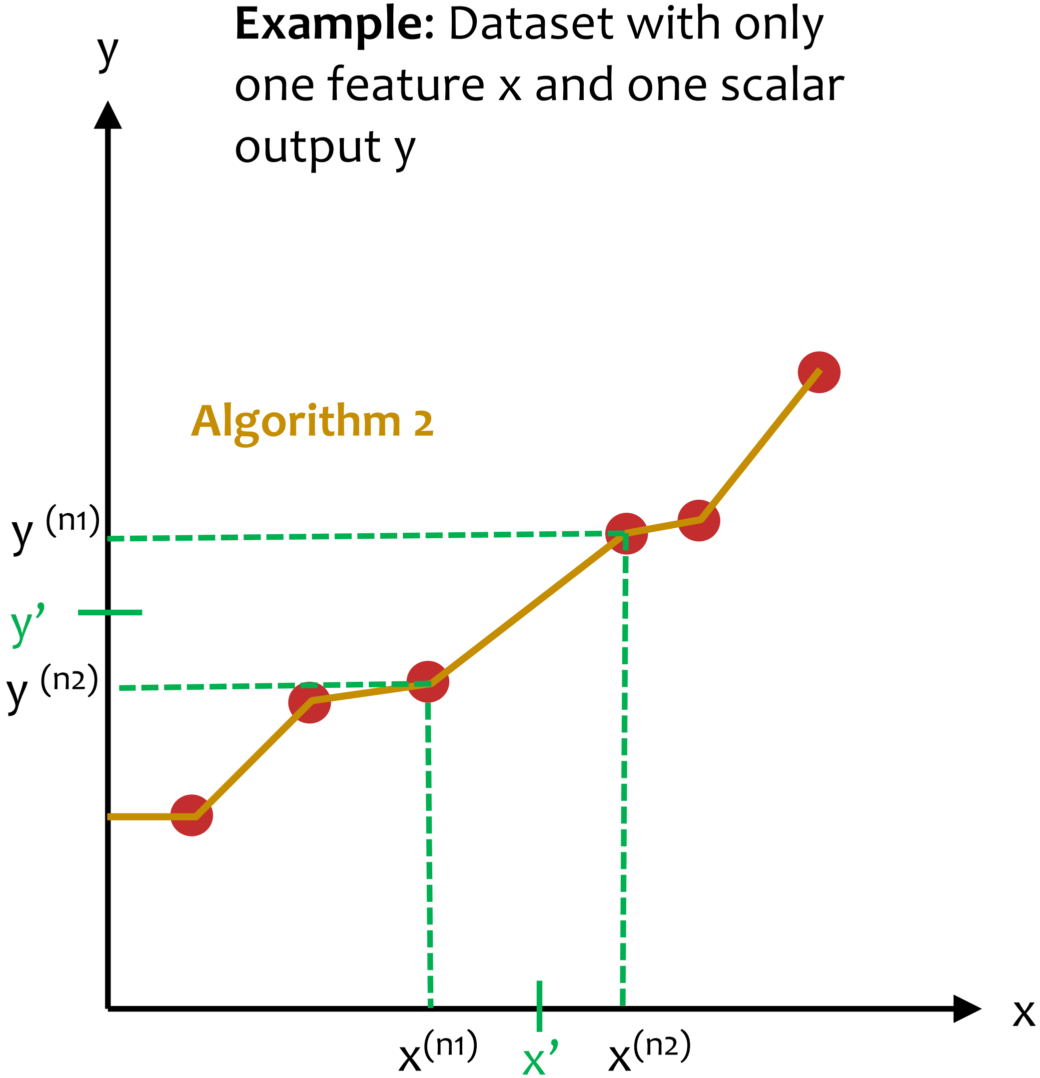
Probabilistic Learning

(Binary Logistic Regression)

Matt Gormley
Lecture 9
Feb. 15, 2023

k-NN Regression

Example: Dataset with only one feature x and one scalar output y



This version is incorrect.

Algorithm 1: k=1 Nearest Neighbor Regression

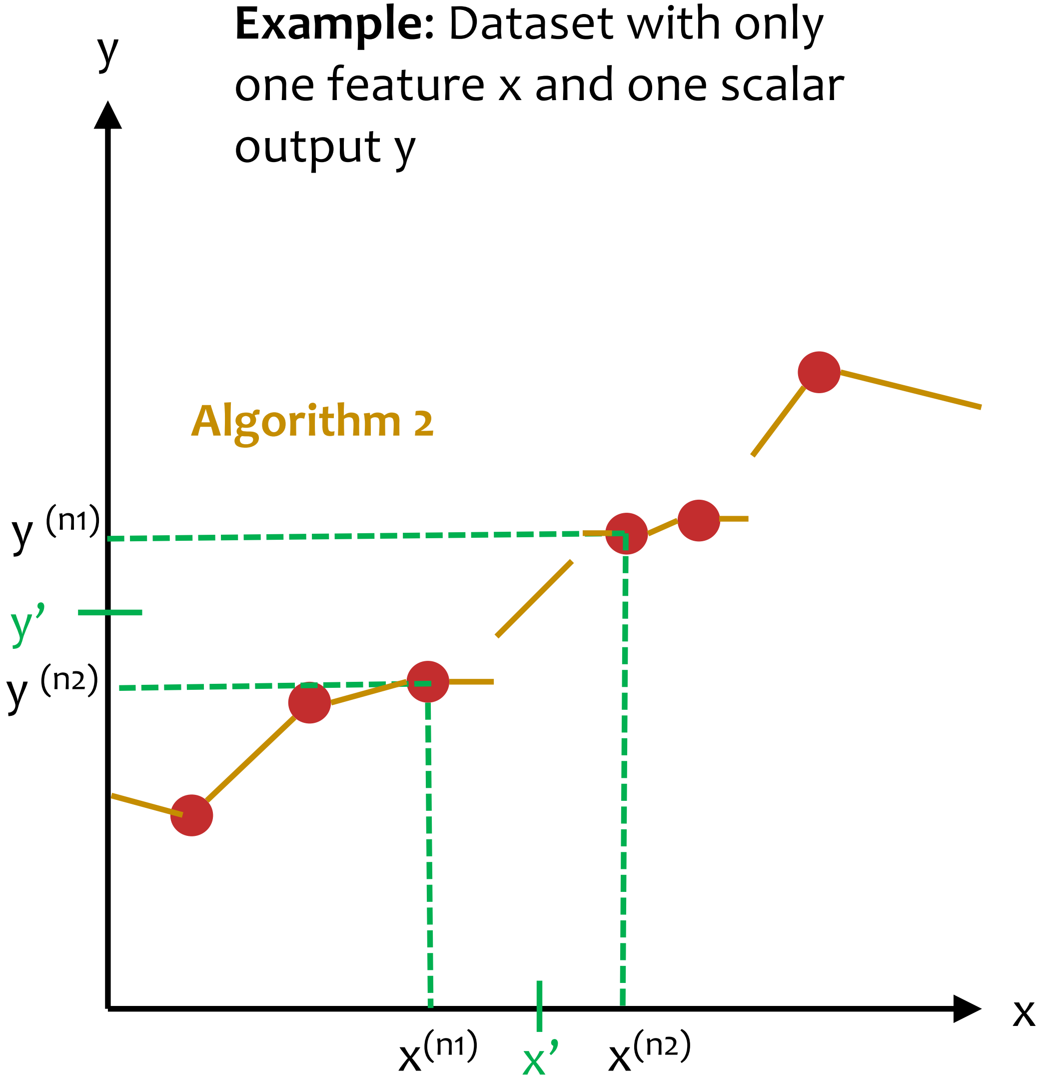
- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest x in training data and return its y

Algorithm 2: k=2 Nearest Neighbors Distance Weighted Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest two instances $x^{(n1)}$ and $x^{(n2)}$ in training data and return the weighted average of their y values

k-NN Regression

Example: Dataset with only one feature x and one scalar output y



Algorithm 1: k=1 Nearest Neighbor Regression

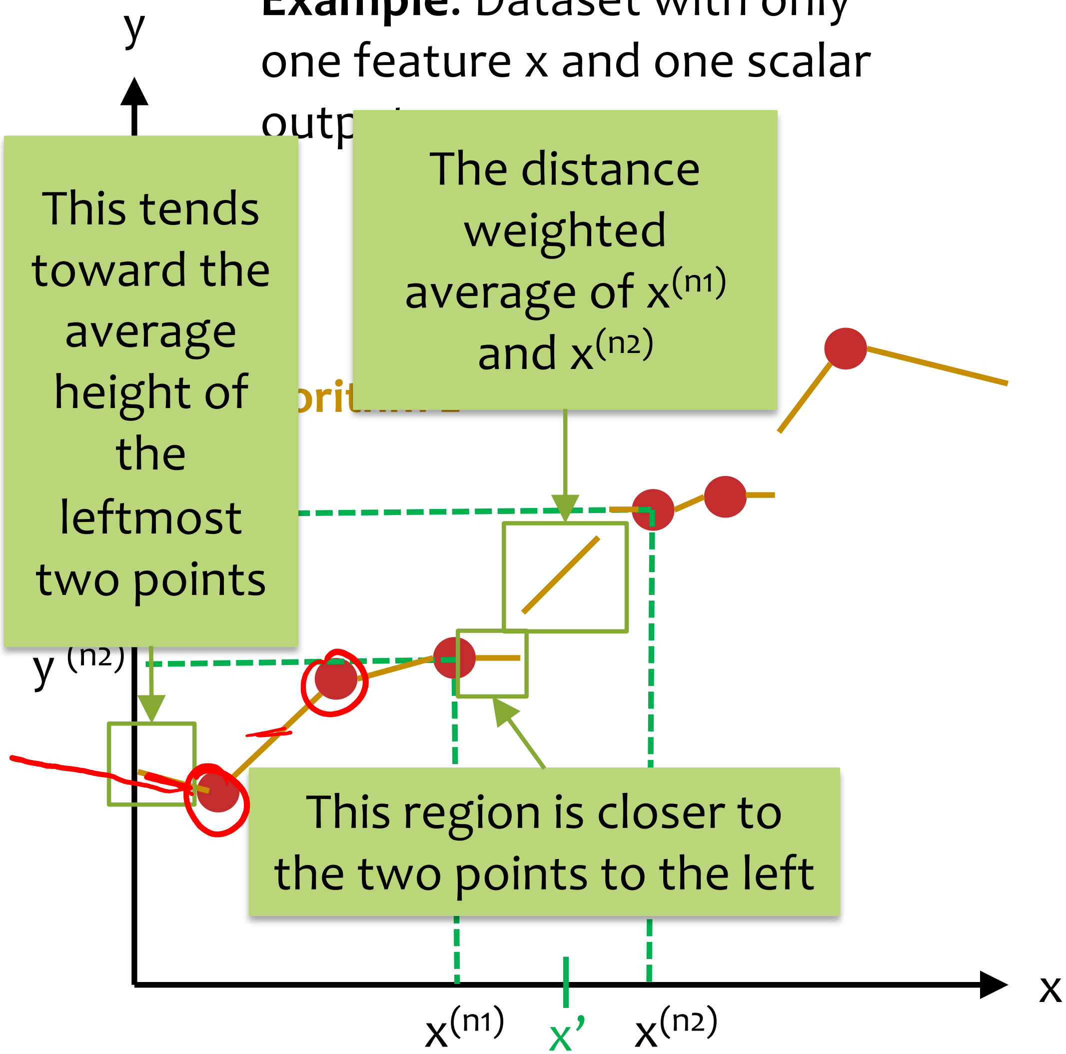
- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest x in training data and return its y

Algorithm 2: k=2 Nearest Neighbors Distance Weighted Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest two instances $x^{(n1)}$ and $x^{(n2)}$ in training data and return the weighted average of their y values

k-NN Regression

Example: Dataset with only one feature x and one scalar outp



Algorithm 1: k=1 Nearest Neighbor Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest x in training data and return its y

Algorithm 2: k=2 Nearest Neighbors Distance Weighted Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest two instances $x^{(n1)}$ and $x^{(n2)}$ in training data and return the weighted average of their y values

Reminders

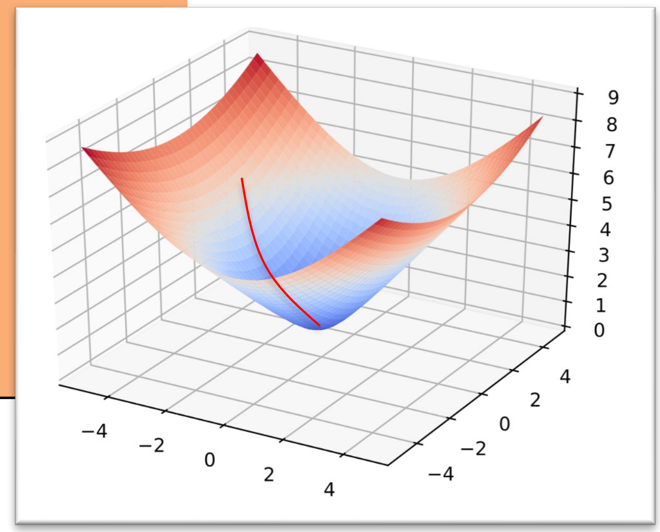
- Practice Problems 1
  - released on course website
- Exam 1: Thu, Feb. 16
  - Time: 6:30 – 8:30pm
  - Location: Your room/seat assignment will be announced on Piazza
- Homework 4: Logistic Regression
  - Out: Fri, Feb 17
  - Due: Sun, Feb. 26 at 11:59pm

OPTIMIZATION METHOD #3: STOCHASTIC GRADIENT DESCENT

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: $\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
5: $\quad$**return** $\boldsymbol{\theta}$

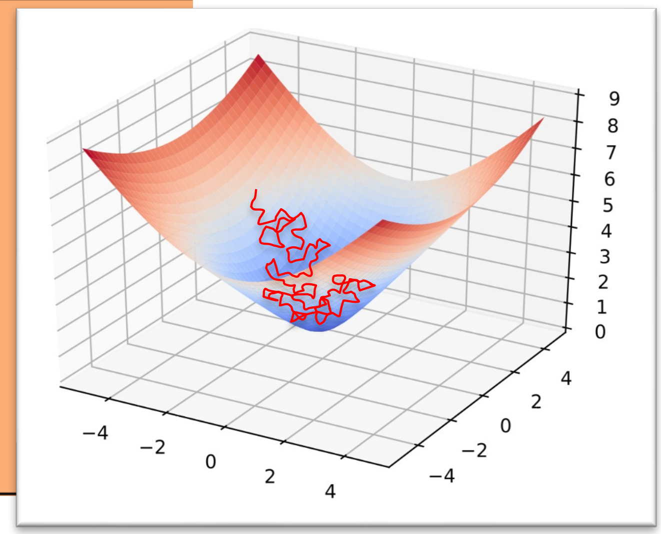


M

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**
    $i \sim \text{Uniform}(\{1, 2, \ldots, N\})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
  return $\boldsymbol{\theta}$



per-example objective:

$$J^{(i)}(\boldsymbol{\theta})$$

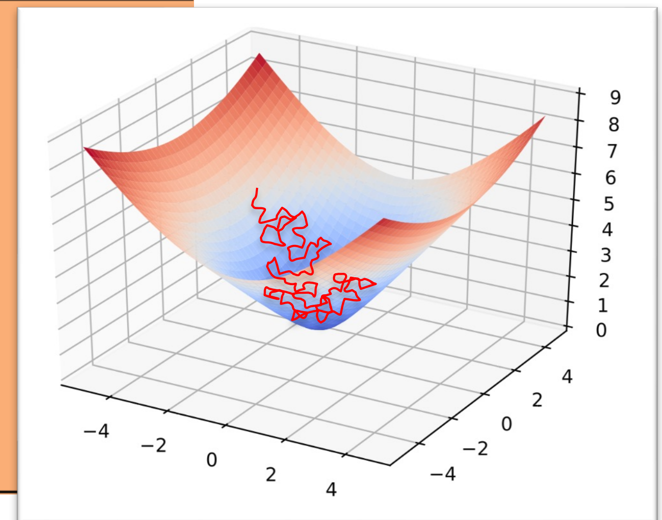
original objective:

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

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**
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
  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}).

per-example objective:

$$J^{(i)}(\boldsymbol{\theta})$$

original objective:

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

Background: Probability

Expectation of a function of a random variable

- For any discrete random variable X

$$E_X[f(X)] = \sum_{x \in \mathcal{X}} P(X = x) f(x)$$

$i \sim I \sim \text{Uniform}(1, \dots, N)$

Why does SGD work?

- If the example is sampled uniformly at random, the expected value of the pointwise gradient is the same as the full gradient!

$E_I[f(i)]$ $\quad P(I=i)$ $\quad f(i)$

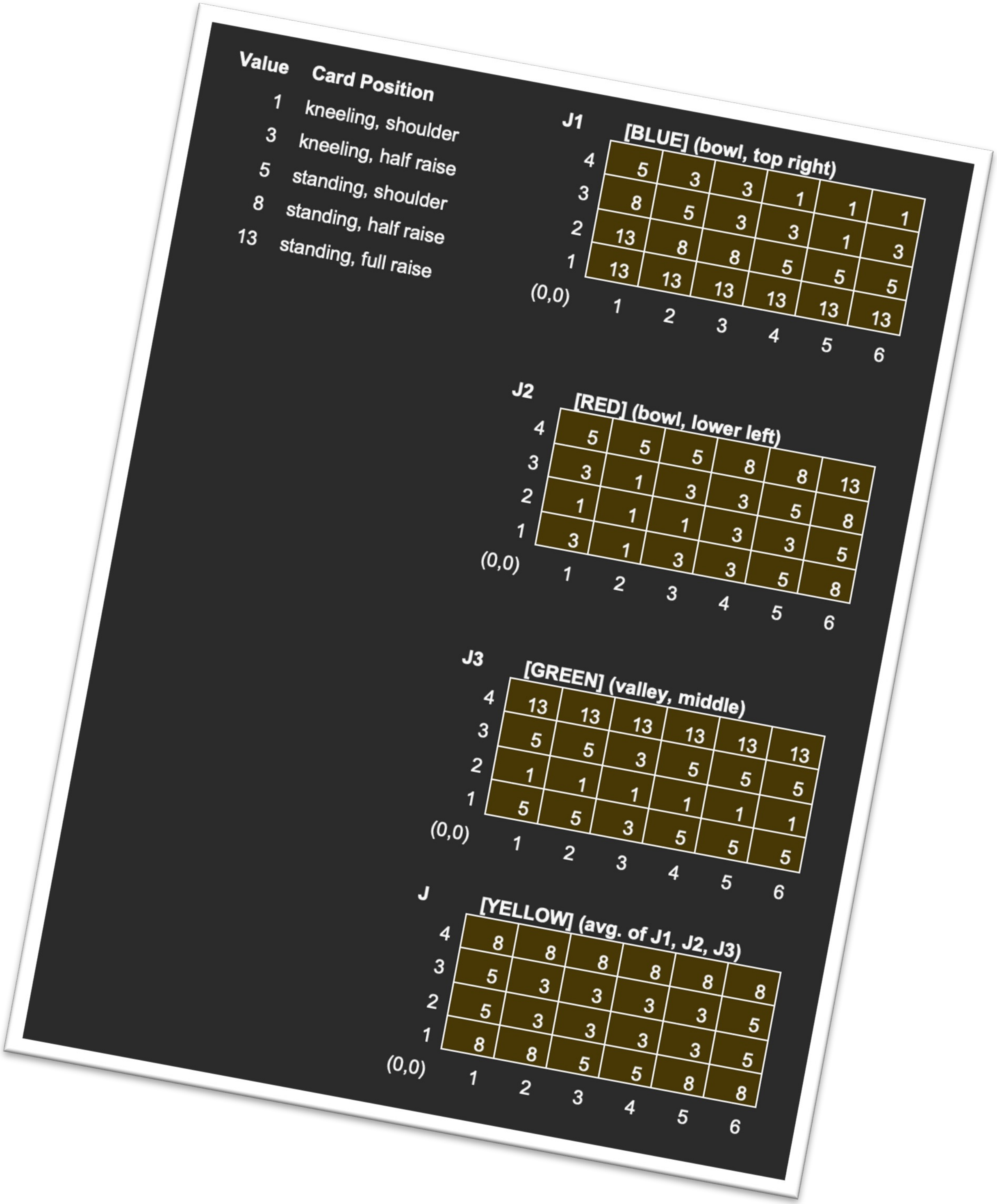
$$E\left[\nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})\right] = \sum_{i=1}^{N} \left(\text{probability of selecting } \boldsymbol{x}^{(i)}, y^{(i)}\right) \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$$

$$= \sum_{i=1}^{N} \left(\frac{1}{N}\right) \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$$

$$= \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$$

$$= \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

- In practice, the data set is randomly shuffled then looped through so that each data point is used equally often



Value	Card Position
1	kneeling, shoulder
3	kneeling, half raise
5	standing, shoulder
8	standing, half raise
13	standing, full raise

J1 [BLUE] (bowl, top right)

	1	2	3	4	5	6
4	5	3	3	1	1	1
3	8	5	3	3	1	3
2	13	8	8	5	5	5
1	13	13	13	13	13	13
(0,0)						

J2 [RED] (bowl, lower left)

	1	2	3	4	5	6
4	5	5	5	8	8	13
3	3	1	3	3	5	8
2	1	1	1	3	3	5
1	3	1	3	3	5	8
(0,0)						

J3 [GREEN] (valley, middle)

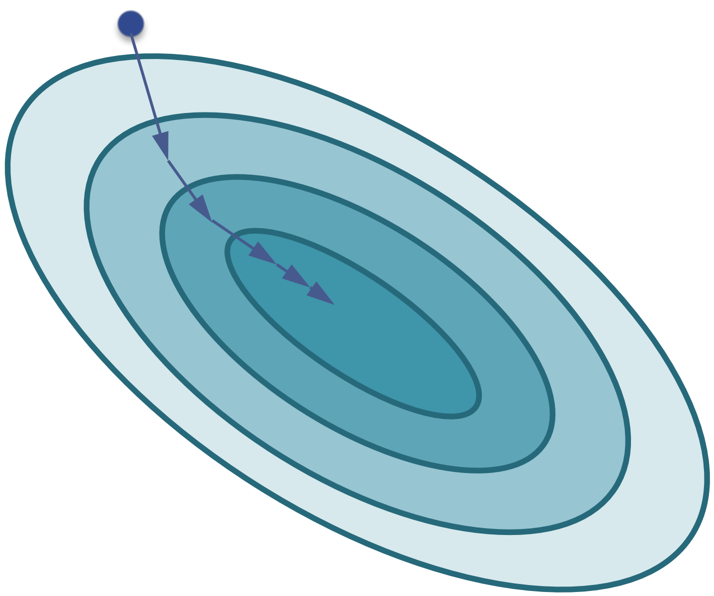
	1	2	3	4	5	6
4	13	13	13	13	13	13
3	5	5	3	5	5	5
2	1	1	1	1	1	1
1	5	5	3	5	5	5
(0,0)						

J [YELLOW] (avg. of J1, J2, J3)

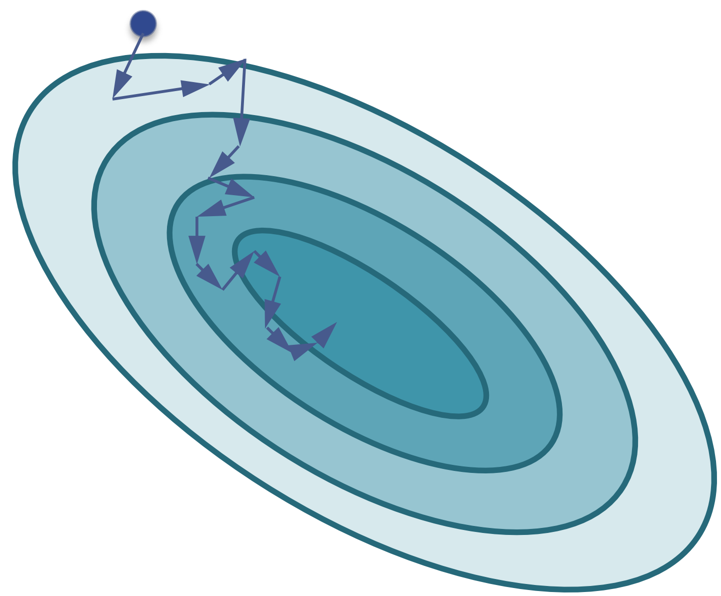
	1	2	3	4	5	6
4	8	8	8	8	8	8
3	5	3	3	3	3	5
2	5	3	3	3	3	5
1	8	8	5	5	8	8
(0,0)						

SGD VS. GRADIENT DESCENT

SGD vs. Gradient Descent



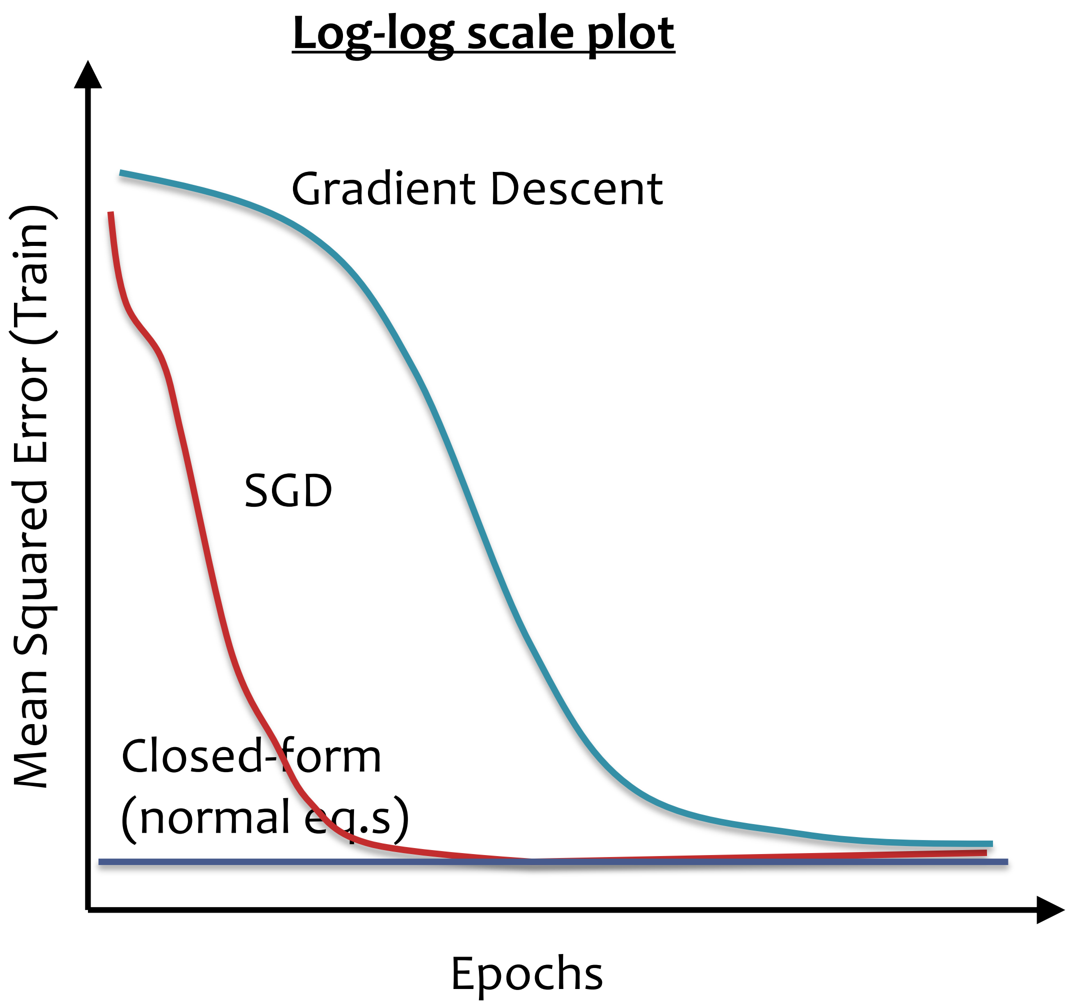
Gradient Descent



Stochastic Gradient Descent

SGD vs. Gradient Descent

- Empirical comparison:



- *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

SGD vs. Gradient Descent

- Theoretical comparison:

argmin

Define convergence to be when $J(\boldsymbol{\theta}^{(t)}) - J(\boldsymbol{\theta}^{*}) < \epsilon$

Method	Steps to Convergence	Computation per Step
Gradient descent	$O(\log {}^{1}/_{\epsilon})$	$O(NM)$
SGD	$O({}^{1}/_{\epsilon})$	$O(M)$

(with high probability under certain assumptions)

Main Takeaway: SGD has much slower asymptotic convergence (i.e. it's slower in theory), but is often much faster in practice.

SGD FOR LINEAR REGRESSION

Linear Regression as Function Approximation

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

1. Assume $\mathcal{D}$ generated as:

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

2. Choose hypothesis space, $\mathcal{H}$:
all linear functions in M-dimensional space

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}, \boldsymbol{\theta} \in \mathbb{R}^M\}$$

3. Choose an objective function:
mean squared error (MSE)

$$\begin{aligned} J(\boldsymbol{\theta}) &= \frac{1}{N} \sum_{i=1}^N e_i^2 \\ &= \frac{1}{N} \sum_{i=1}^N \left(y^{(i)} - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\right)^2 \\ &= \frac{1}{N} \sum_{i=1}^N \left(y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)^2 \end{aligned}$$

4. Solve the unconstrained optimization problem via favorite method:
   - *gradient descent*
   - *closed form*
   - *stochastic gradient descent*
   - ...

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$$

5. Test time: given a new $\mathbf{x}$, make prediction $\hat{y}$

$$\hat{y} = h_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) = \hat{\boldsymbol{\theta}}^T \mathbf{x}$$

Gradient Calculation for Linear Regression

Derivative of $J^{(i)}(\boldsymbol{\theta})$:

$$\begin{aligned}
\frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta}) &= \frac{d}{d\theta_k} \frac{1}{2}(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 \\
&= \frac{1}{2} \frac{d}{d\theta_k} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 \\
&= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \frac{d}{d\theta_k} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \\
&= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \frac{d}{d\theta_k} \left(\sum_{j=1}^{K} \theta_j x_j^{(i)} - y^{(i)} \right) \\
&= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_k^{(i)}
\end{aligned}$$

Derivative of $J(\boldsymbol{\theta})$:

$$\begin{aligned}
\frac{d}{d\theta_k} J(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta}) \\
&= \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_k^{(i)}
\end{aligned}$$

$\boxed{\text{Gradient of } J^{(i)}(\boldsymbol{\theta})}$ [used by SGD]

$$\nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J^{(i)}(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J^{(i)}(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J^{(i)}(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_1^{(i)} \\ (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_2^{(i)} \\ \vdots \\ (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_N^{(i)} \end{bmatrix}$$
$$= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$$

Gradient of $J(\boldsymbol{\theta})$ [used by Gradient Descent]

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_1^{(i)} \\ \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_2^{(i)} \\ \vdots \\ \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_N^{(i)} \end{bmatrix}$$
$$= \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$$

SGD for Linear Regression

SGD applied to Linear Regression is called the "Least Mean Squares" algorithm

Algorithm 1 Least Mean Squares (LMS)

procedure LMS($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$ ▷ Initialize parameters
  while not converged **do**
    for $i \in$ shuffle($\{1, 2, \ldots, N\}$) **do**
      $\mathbf{g} \leftarrow (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})\mathbf{x}^{(i)}$ ▷ Compute gradient
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \mathbf{g}$ ▷ Update parameters
  return $\boldsymbol{\theta}$

GD for Linear Regression

Gradient Descent for Linear Regression repeatedly takes steps opposite the gradient of the objective function

Algorithm 1 GD for Linear Regression

1: **procedure** GDLR($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
2: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$ ▷ Initialize parameters
3: **while** not converged **do**
4: $\mathbf{g} \leftarrow \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$ ▷ Compute gradient
5: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \mathbf{g}$ ▷ Update parameters
6: **return** $\boldsymbol{\theta}$

Solving Linear Regression

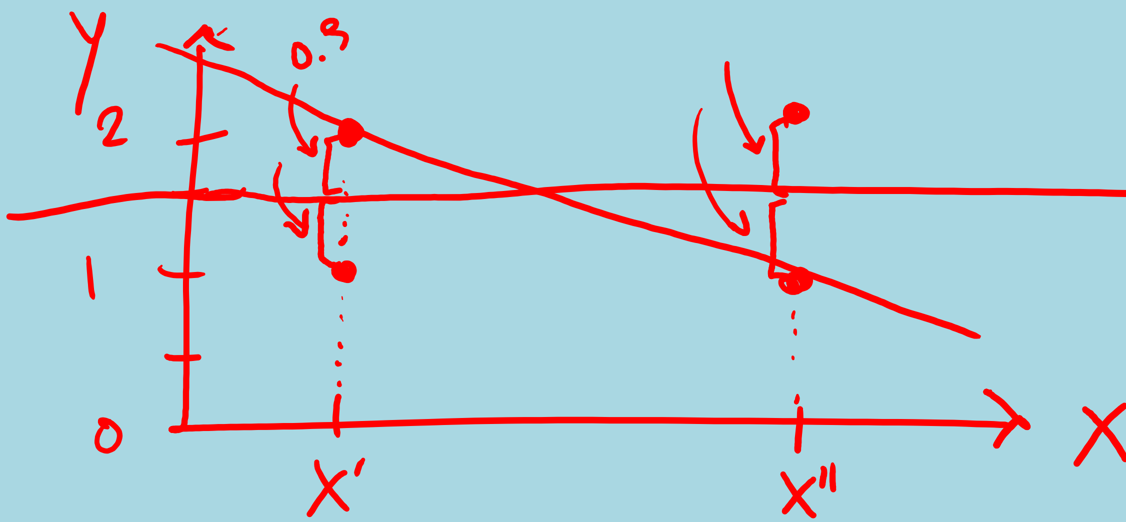
54%

1% 49% 50%

Question: Q1 A = toxic B = True C = False

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:



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

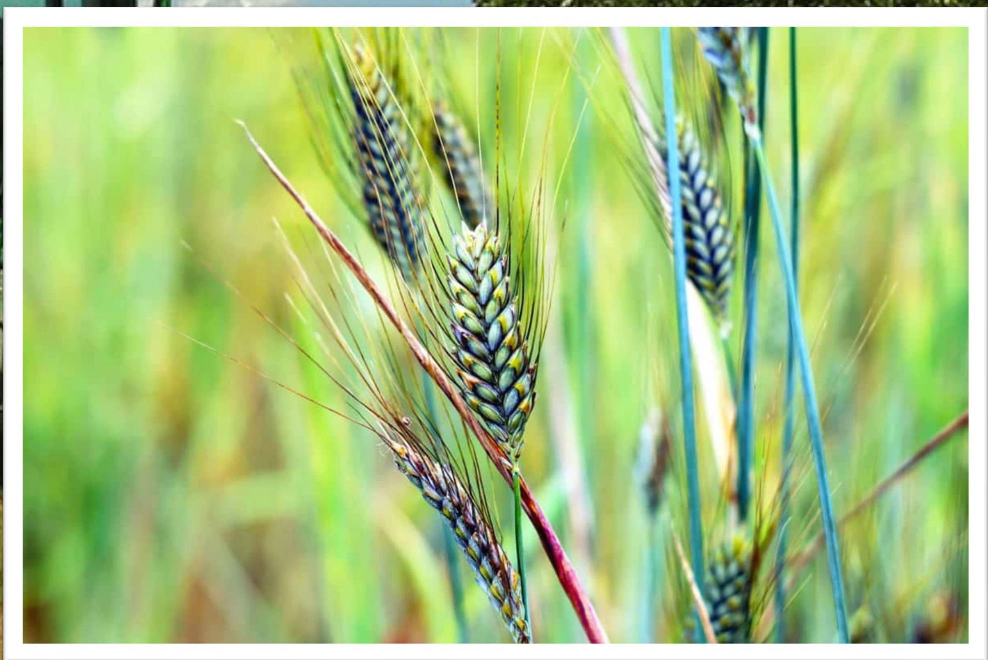
Today, we assume that our output is **sampled** from a conditional **probability distribution:**

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

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?



MAXIMUM LIKELIHOOD ESTIMATION

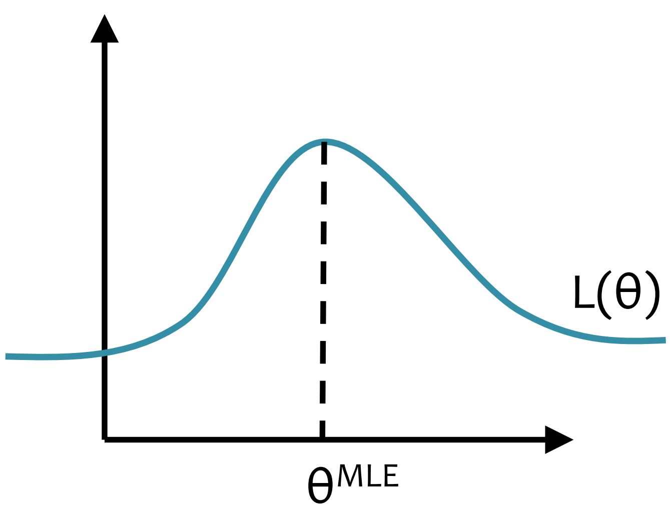
MLE

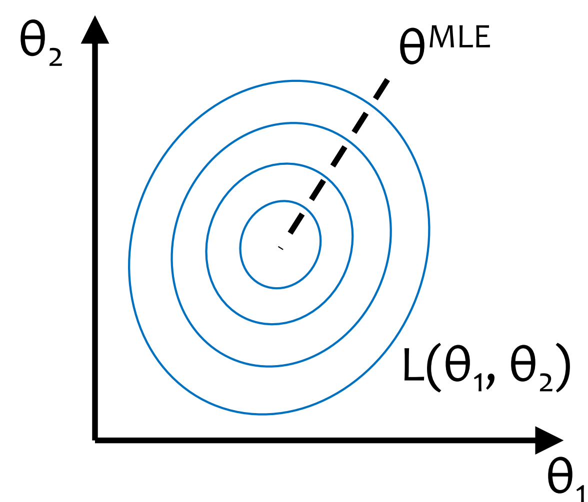
Suppose we have data $\mathcal{D} = \{x^{(i)}\}_{i=1}^N$

Principle of Maximum Likelihood Estimation:
Choose the parameters that maximize the likelihood of the data.

$$\boldsymbol{\theta}^{\text{MLE}} = \underset{\boldsymbol{\theta}}{\text{argmax}} \underbrace{\prod_{i=1}^{N} p(\mathbf{x}^{(i)} | \boldsymbol{\theta})}_{\text{likelihood}}$$

Maximum Likelihood Estimate (MLE)





MLE

What does maximizing likelihood accomplish?

- There is only a finite amount of probability mass (i.e. sum-to-one constraint) $\sum_{x \in \mathcal{X}} P(X=x) = 1$
- MLE tries to allocate **as much** probability mass **as possible** to the things we have observed…

  … **at the expense** of the things we have **not** observed

Maximum Likelihood Estimation

The principle of Maximum likelihood estimator (MLE):

Choose parameters that make the data "most likely".

Assumptions: Data generated iid from distribution $p^*(x \mid \vec{\theta}^*)$ and comes from a family of distn parameterized $\theta \in \Theta$

independent and identically distributed

set of possible parameters

Formally:

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

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

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

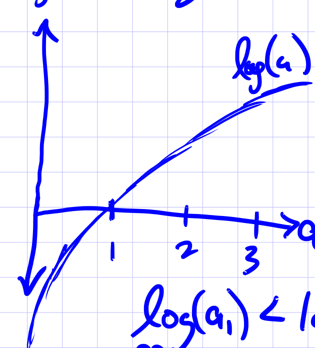
usually a continuous optimization

where $\ell(\theta) \triangleq \log p(D \mid \theta)$

"log-likelihood"

treat as function of θ where D is constant

since log is monotonic



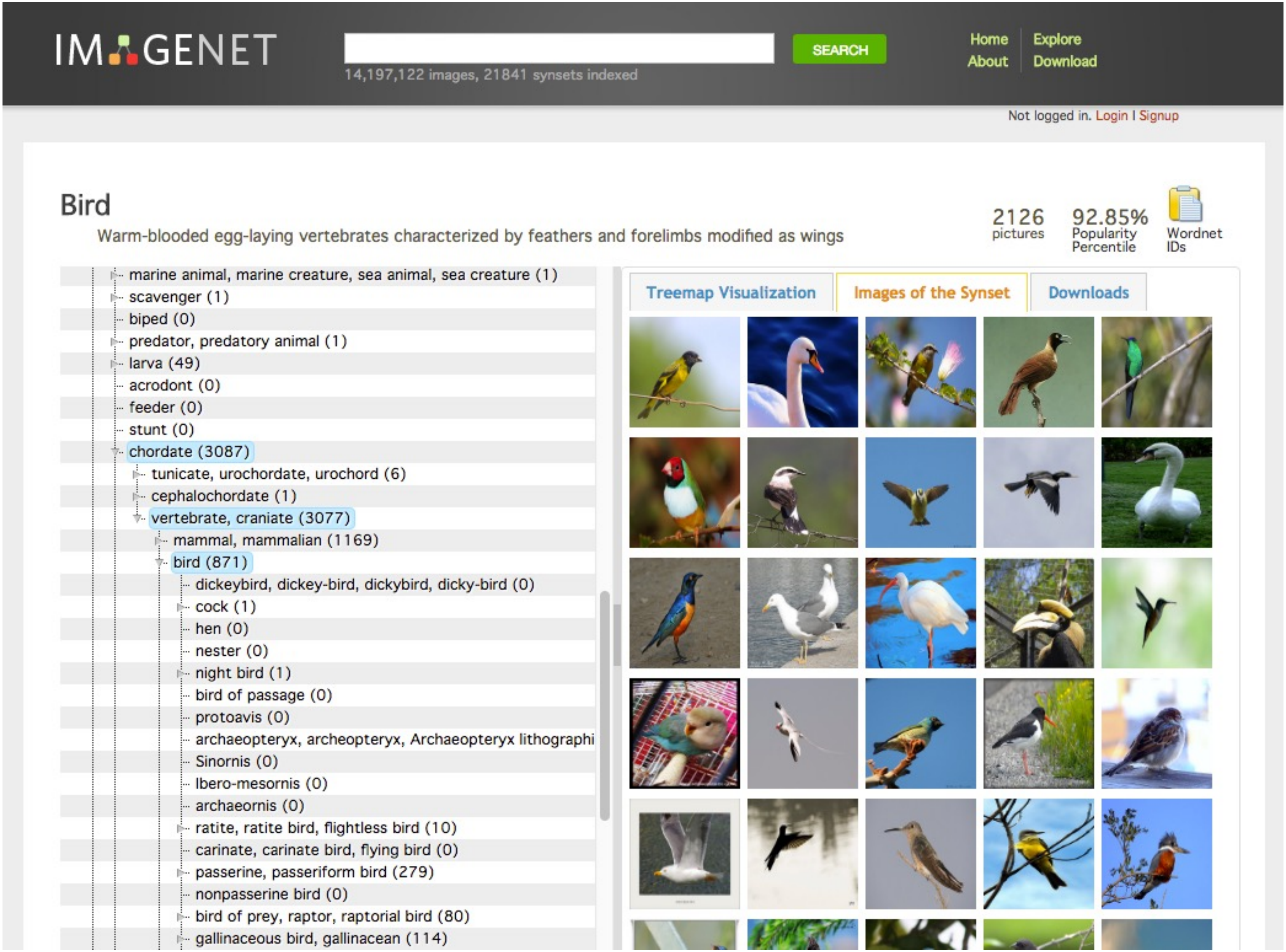
$\log(a_1) < \log(a_2)$ iff $a_1 < a_2$

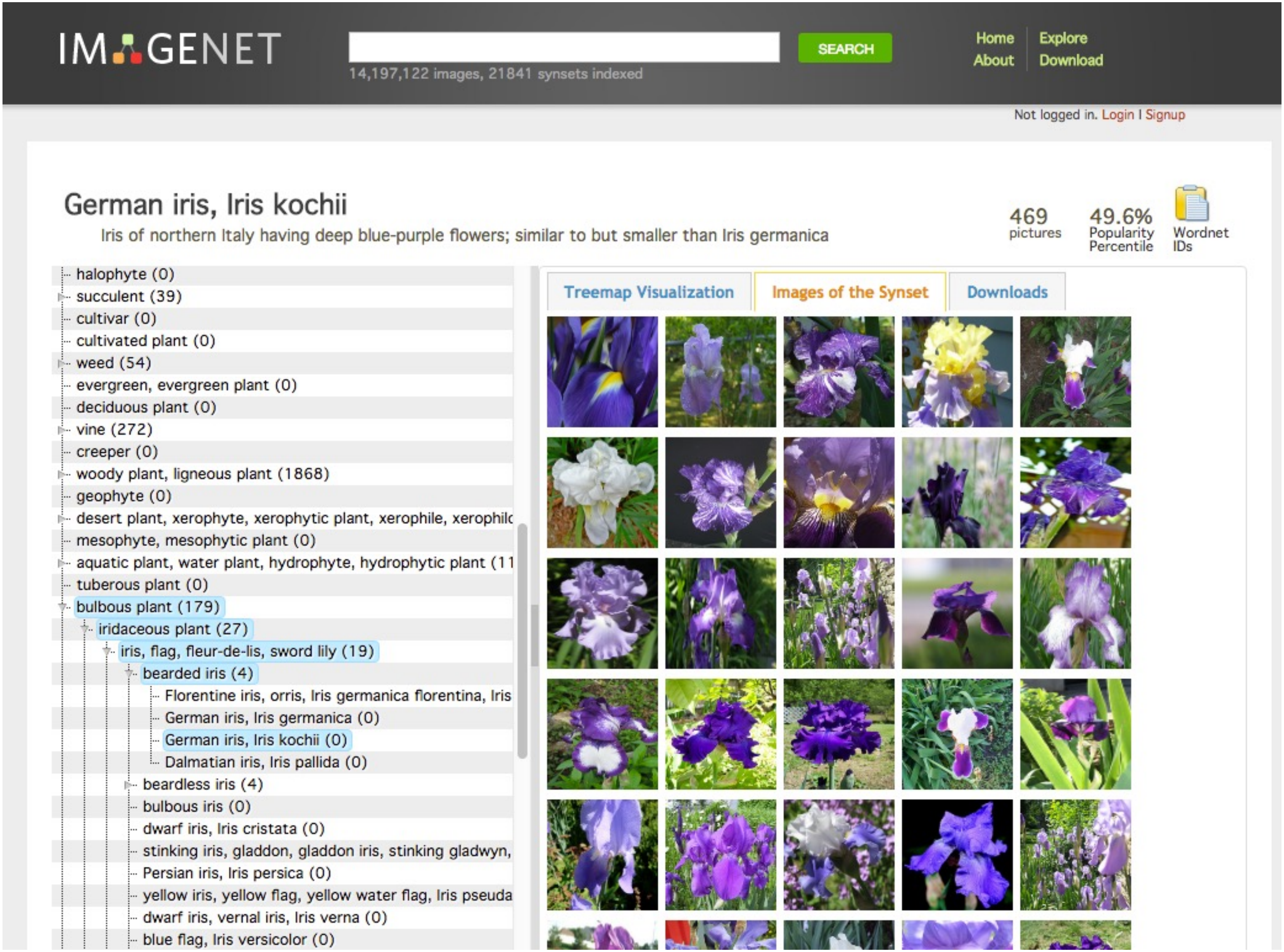
$\Rightarrow \log(f(a_1)) < \log(f(a_2))$ iff $f(a_1) < f(a_2)$

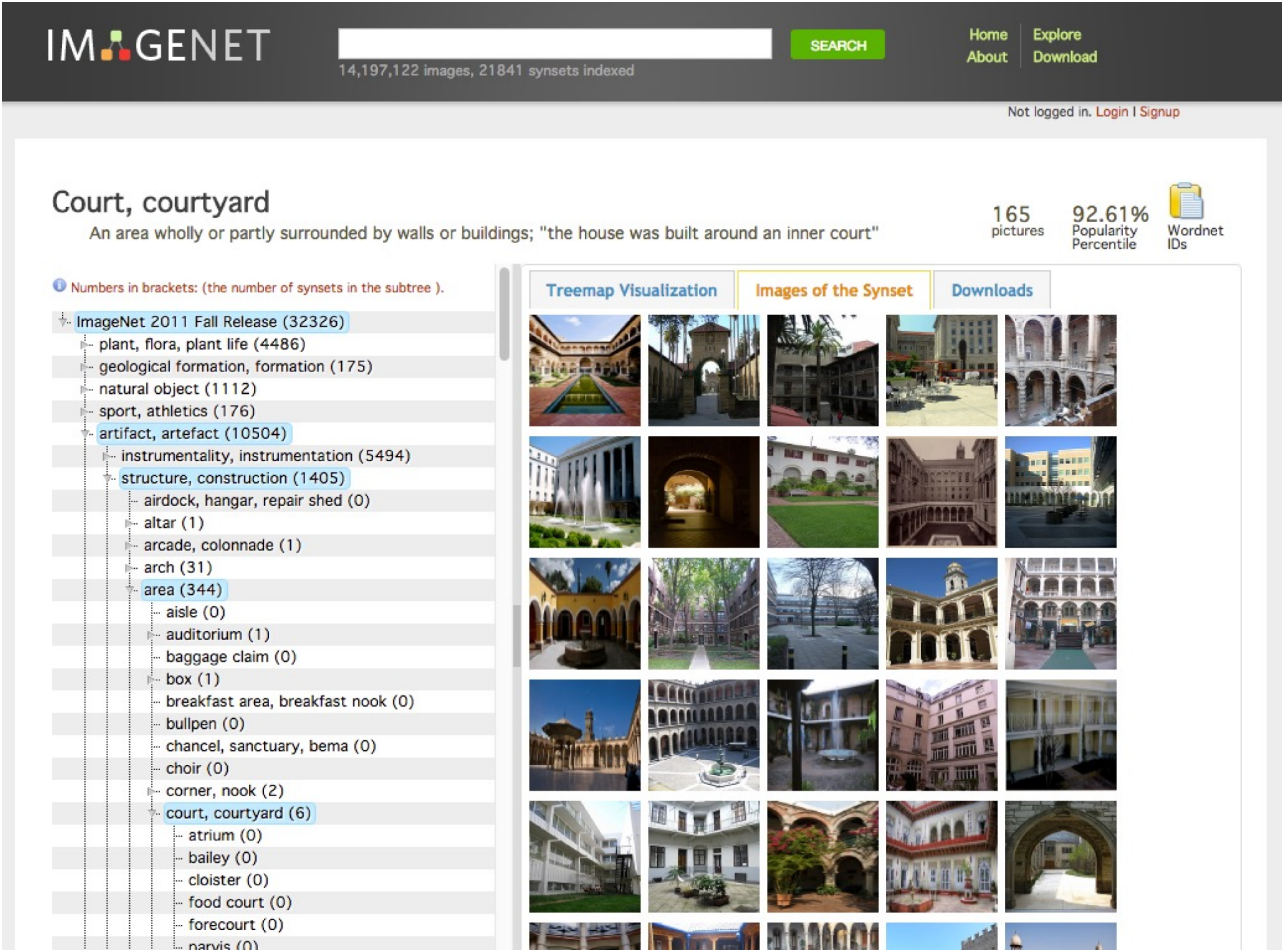
MOTIVATION:
LOGISTIC REGRESSION

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


IM GENET
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, xerophilo
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
About
Explore
Download
Not logged in. Login I 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

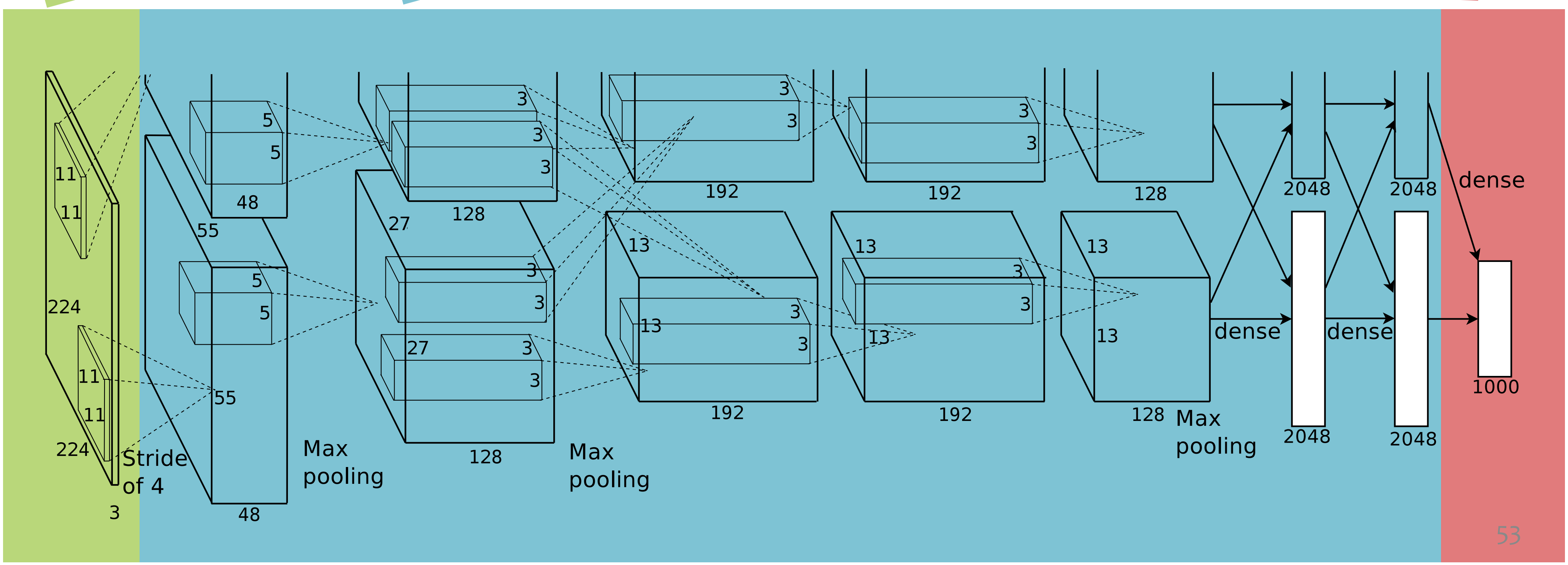
Example: Image Classification

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



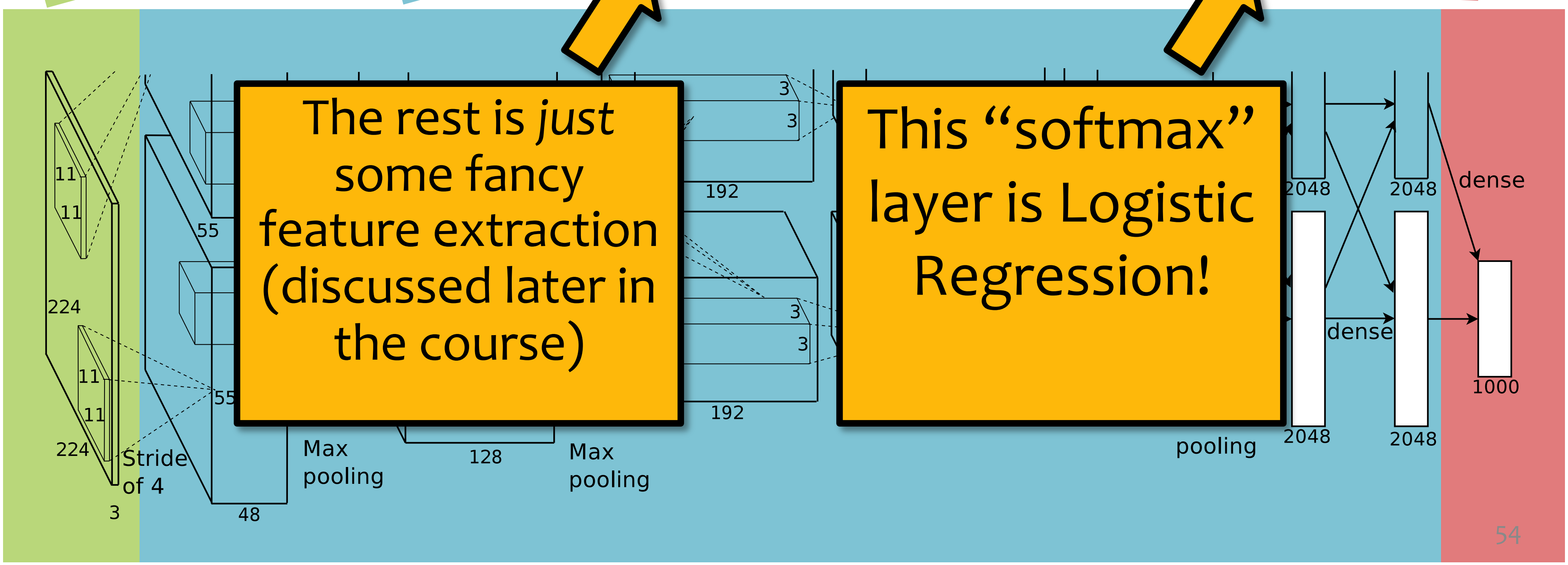
Example: Image Classification

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.

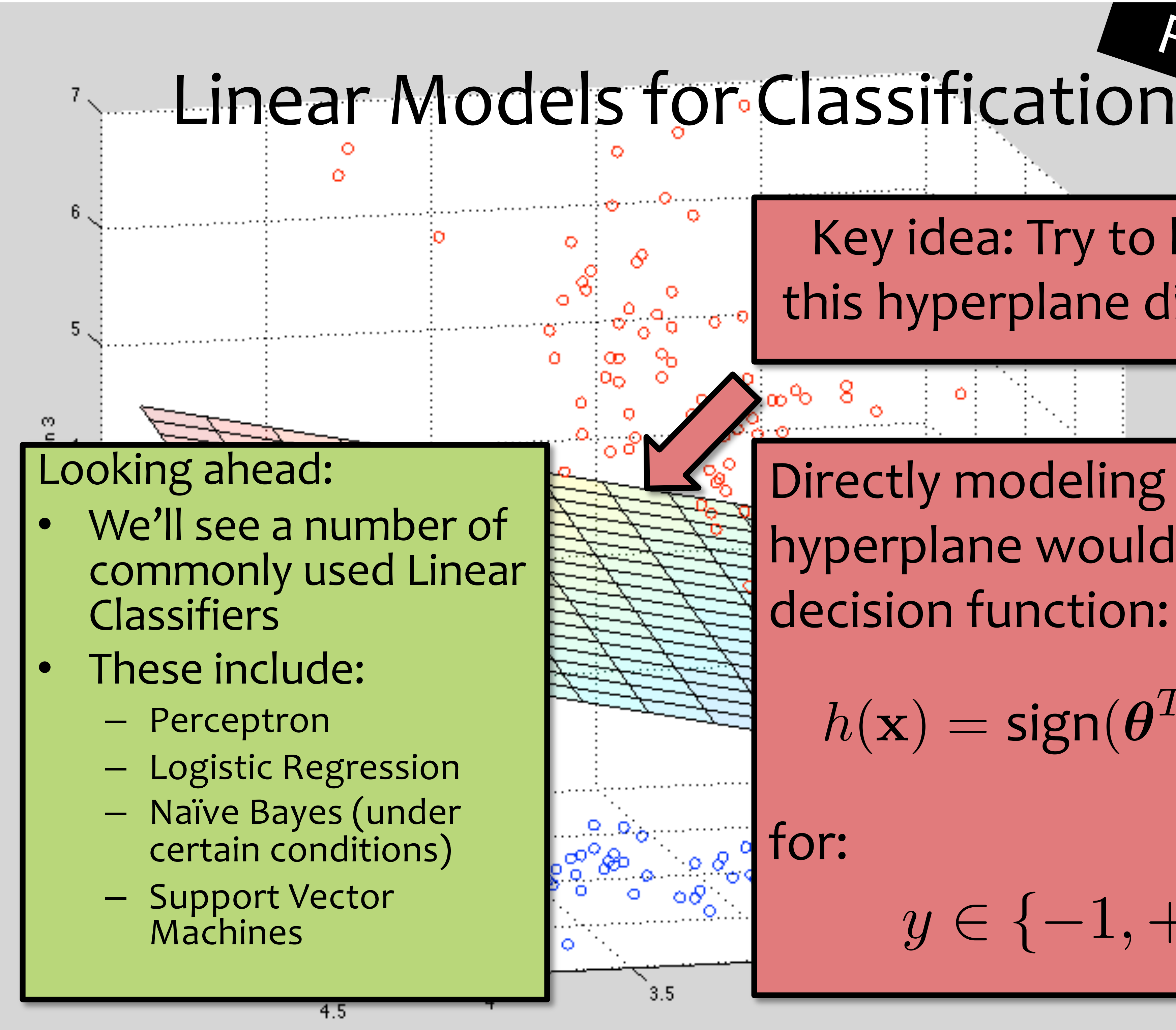
$$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N} \text{ where } \mathbf{x} \in \mathbb{R}^M \text{ and } y \in \{0, 1\}$$

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 to get **x'**!

Hyperplane (Definition 1):

$$\mathcal{H} = \{\mathbf{x} : \mathbf{w}^T \mathbf{x} + b = 0\}$$

Hyperplane (Definition 2):

$$\mathcal{H} = \{\mathbf{x}' : \boxed{\boldsymbol{\theta}^T \mathbf{x}' = 0}$$

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

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

$$\mathbf{x}' = [1, x_1, \ldots, x_M]^T$$

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

Optimization for Linear Classifiers

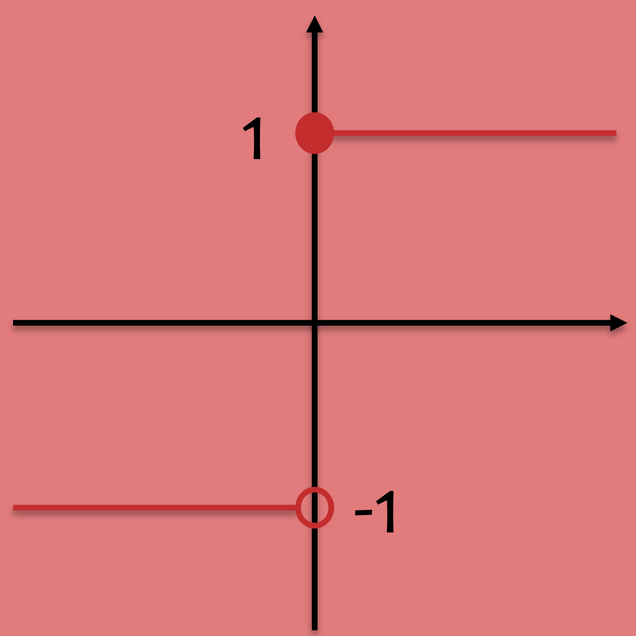
Whiteboard

- Strawman: Mean squared error for Perceptron!
- What does $\boldsymbol{\theta}^T \mathbf{x}$ tell us about **x**?

Using gradient ascent for linear classifiers

Suppose we wanted to learn a linear classifier, but instead of predicting y ∈ {-1,+1} we wanted to predict y ∈ {0,1}

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

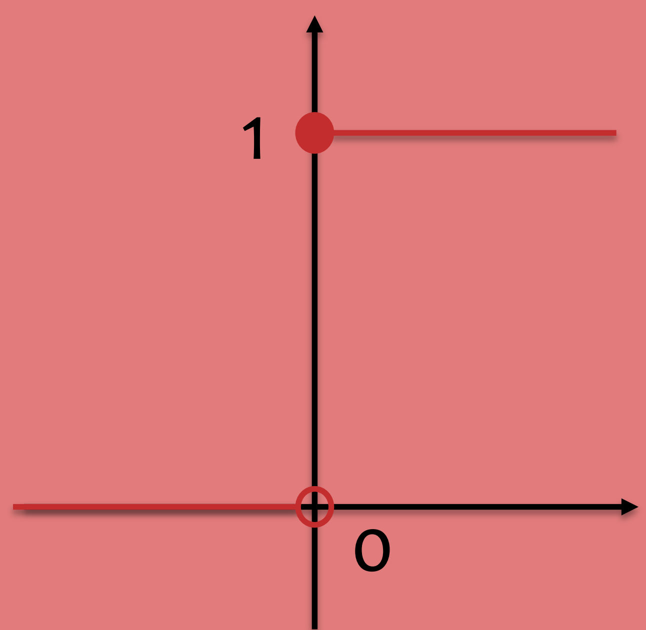


sign(u)

Using gradient ascent for linear classifiers

Suppose we wanted to learn a linear classifier, but instead of predicting y ∈ {-1,+1} we wanted to predict y ∈ {0,1}

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



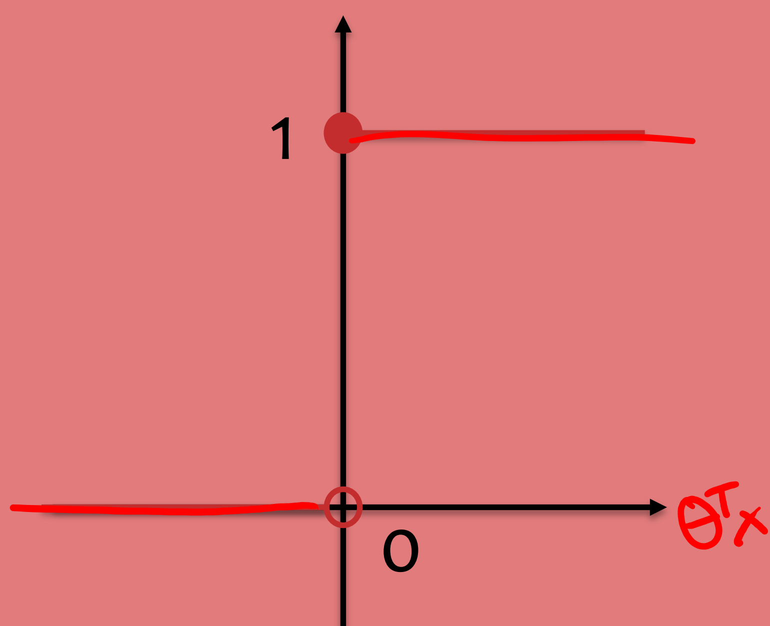
“sign”(u)

Goal: Learn a linear classifier with Gradient Descent

Using gradient ascent for linear classifiers

But this decision function isn't differentiable…

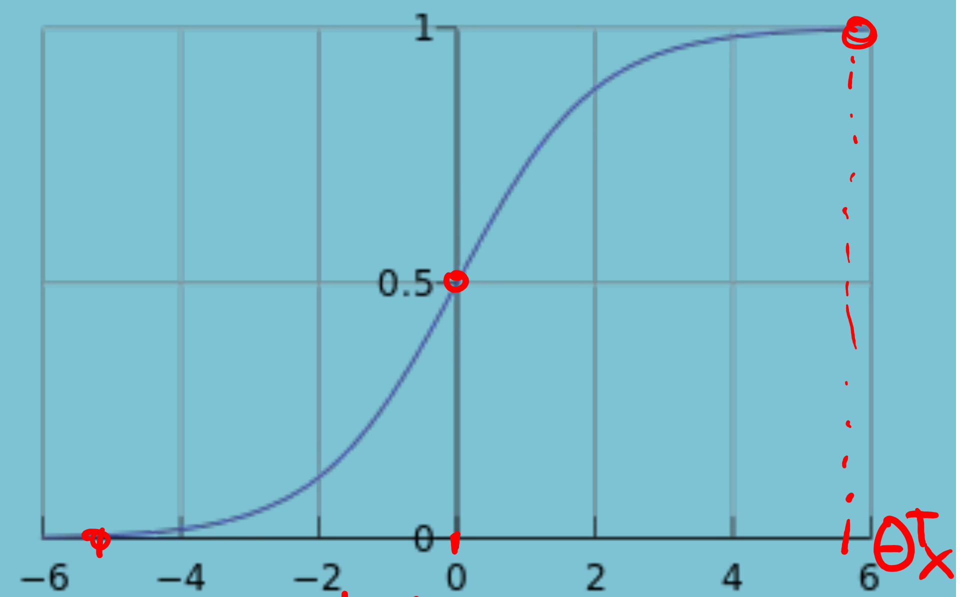
$$h(\mathbf{x}) = \text{“sign”}(\boldsymbol{\theta}^T \mathbf{x})$$



“sign”(u)

Use a differentiable function instead!

$$p_{\boldsymbol{\theta}}(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x})}$$



$$\text{sigmoid}(u) = \text{logistic}(u) \equiv \frac{1}{1 + e^{-u}}$$

Logistic Regression

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

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

Model: Logistic function applied to dot product of parameters with input vector.

$$p_{\boldsymbol{\theta}}(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x})}$$

Learning: finds the parameters that minimize some objective function.

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Prediction: Output is the most probable class.

$$\hat{y} = \operatorname*{argmax}_{y \in \{0,1\}} p_{\boldsymbol{\theta}}(y|\mathbf{x})$$

Logistic Regression

Whiteboard

- Logistic Regression Model
- Partial derivative for logistic regression
- Gradient for logistic regression
- Decision boundary