Introduction to Deep Learning

4. Linear & Logistic Regression, Basic Optimization

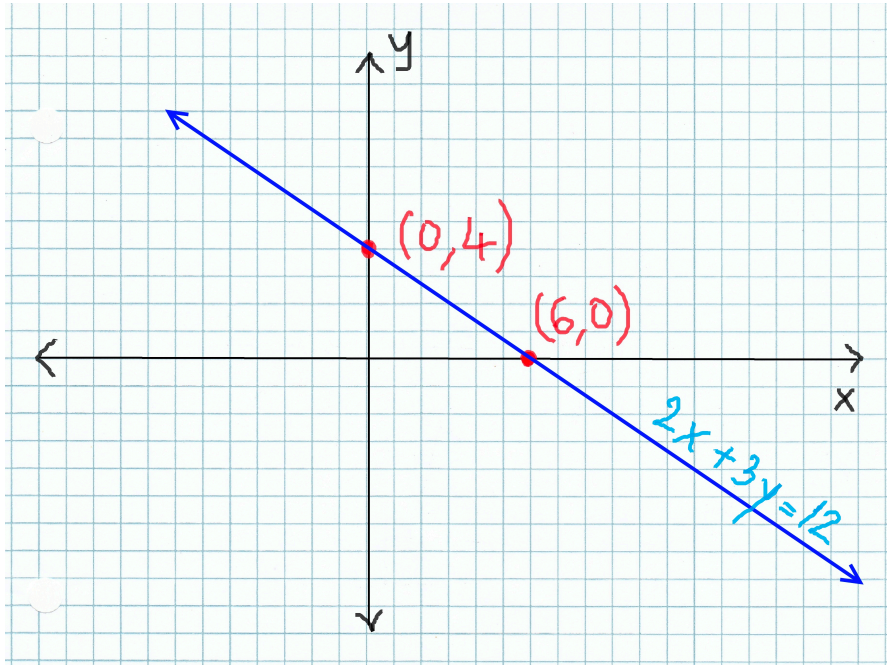
STAT 157, Spring 2019, UC Berkeley

Alex Smola and Mu Li

courses.d2l.ai/berkeley-stat-157

aws

Linear Methods



House Buying 101

- Pick a house, take a tour, and read facts
- Estimate its price, bid it



Listing price from agent

$5,498,000	7	5	4,865 Sq. Ft.
Price	Beds	Baths	$1130 / Sq. Ft.

Redfin Estimate: $5,390,037 On Redfin: 15 days

Predicted sale price

Virtual Tour
- Branded Virtual Tour
- Virtual Tour (External Link)

Parking Information
- Garage (Minimum): 2
- Garage (Maximum): 2
- Parking Description: Attached Garage, On Street
- Garage Spaces: 2

Multi-Unit Information
- # of Stories: 2

School Information
- Elementary School: El Carmelo Eleme
- Elementary School District: Palo Alto
- Middle School: Jane Lathrop Stanford
- High School: Palo Alto High
- High School District: Palo Alto Unified

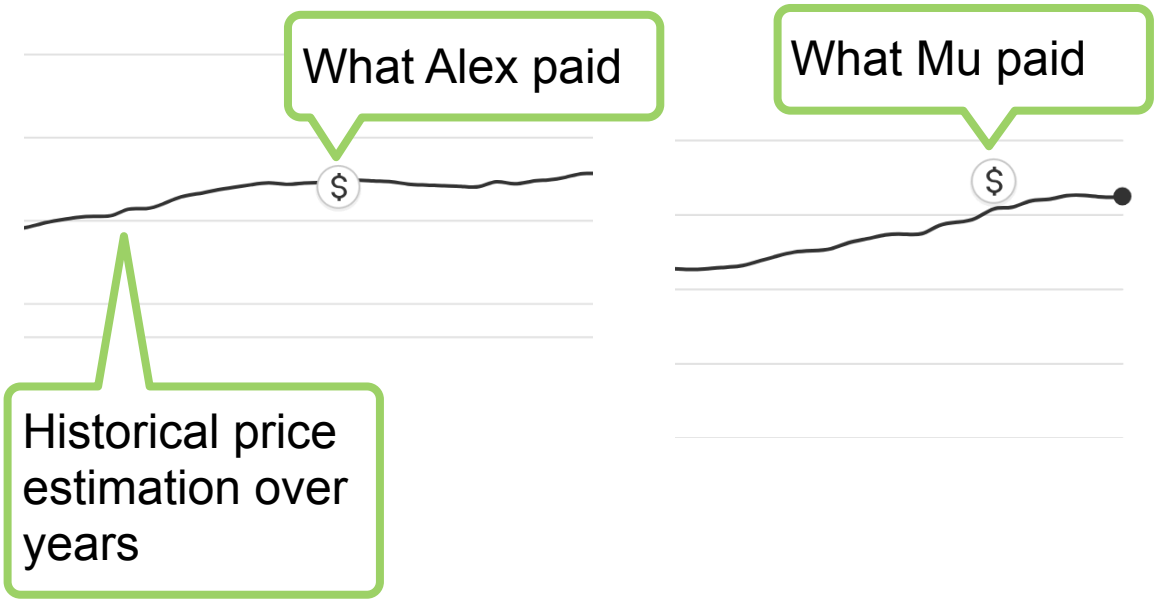
Interior Features

Bedroom Information
- # of Bedrooms (Minimum): 7
- # of Bedrooms (Maximum): 7

- Kitchen Description: Countertop (Gran
  Dishwasher, Garbage Disposal, Hood
  Island with Sink, Microwave, Oven Ran

House Price Predicting

- Very important, that's real money…

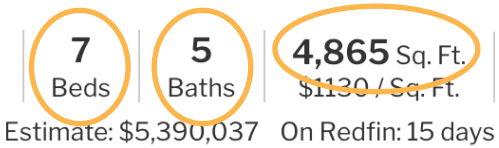


$10K+ gap

Redfin over-estimated the price, and we believed it

A Simplified Model

- Assumption 1: the key factors impacting the prices are
  - #Beds, #Baths, Living Sq. Ft.
  - Denote by x_1, x_2, x_3



- Assumption 2: the sale price is a weighted sum over the key factors

$$y = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

  - Weights and bias are determined later

Linear Model

- Given *n*-dimensional inputs $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$
- Linear model has a *n*-dimensional weight and a bias

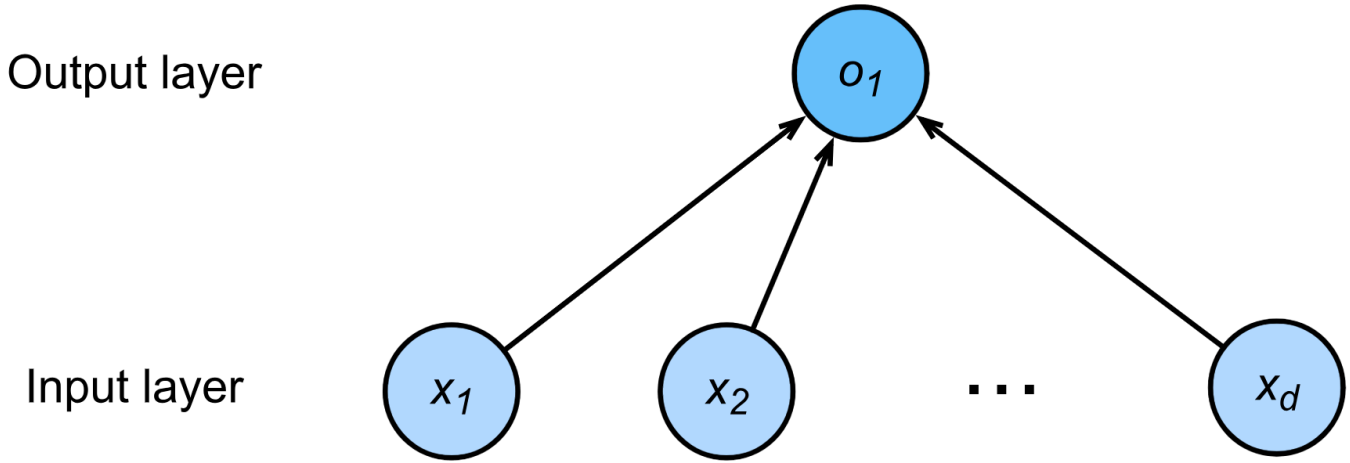
$$\mathbf{w} = [w_1, w_2, \ldots, w_n]^T, \quad b$$

- The output is a weighted sum of the inputs

$$y = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

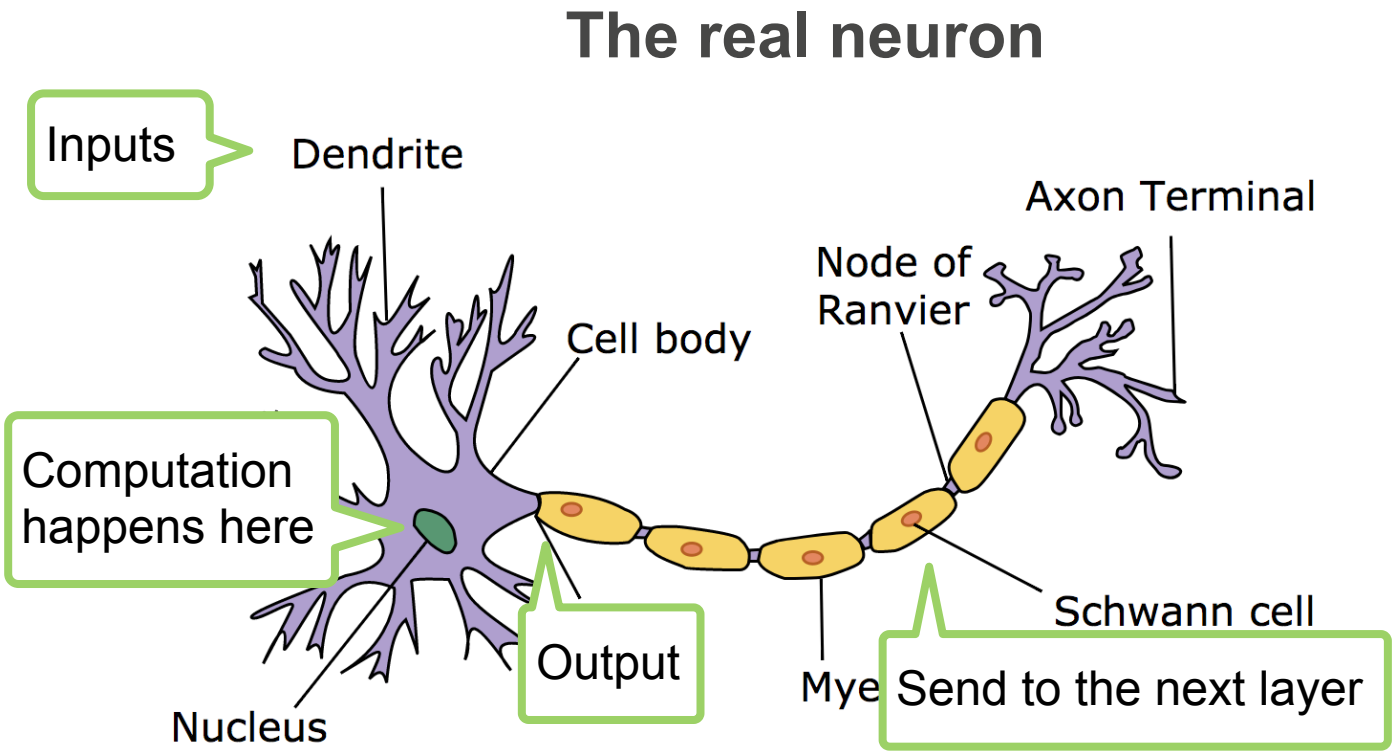
  - Vectorized version $y = \langle \mathbf{w}, \mathbf{x} \rangle + b$

Linear Model as a Single-layer Neural Network



- We can stack multiple layers to get deep neural networks

Neural Networks Derive from Neuroscience



Measure Estimation Quality

- Compare the true value vs the estimated value
  - Real sale price vs estimated house price
- Let y the truth value, and $\hat{y}$ the estimated value, we can compare the loss

$$\ell(y, \hat{y}) = \left(y - \hat{y}\right)^2$$

  - It is called squared loss

Training Data

- Collect multiple data points to fit parameters
  - Houses salad in the last 6 months
- It is called the training data
- The more the better
- Assume *n* examples

$$\mathbf{X} = \left[\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n\right]^T \quad \mathbf{y} = \left[y_0, y_1, \ldots, y_n\right]^T$$

Learn Parameters

- Training loss

$$\ell(\mathbf{X}, \mathbf{y}, \mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \langle \mathbf{x}_i, \mathbf{w} \rangle - b\right)^2 = \frac{1}{n} \left\| \mathbf{y} - \mathbf{X}\mathbf{w} - b \right\|^2$$

- Minimize loss to learn parameters

$$\mathbf{w}^*, \mathbf{b}^* = \arg\min_{\mathbf{w}, b} \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}, b)$$

Closed-form Solution

- Add bias into weights by $\mathbf{X} \leftarrow [\mathbf{X}, \mathbf{1}] \quad \mathbf{w} \leftarrow \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$
- We know

$$\frac{\partial}{\partial \mathbf{w}} \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}) = \frac{2}{n} \left(\mathbf{y} - \mathbf{X}\mathbf{w}\right)^T \mathbf{X}$$

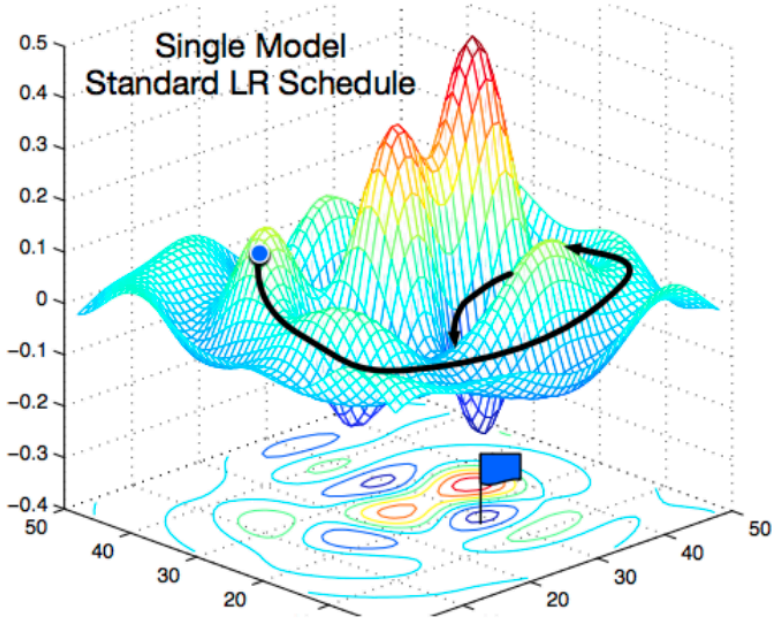
- Loss is convex, so the optimal solutions satisfies

$$\frac{\partial}{\partial \mathbf{w}} \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}) = 0$$

$$\Leftrightarrow \frac{2}{n} \left(\mathbf{y} - \mathbf{X}\mathbf{w}\right)^T \mathbf{X} = 0$$

$$\Leftrightarrow \mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}\mathbf{y}$$

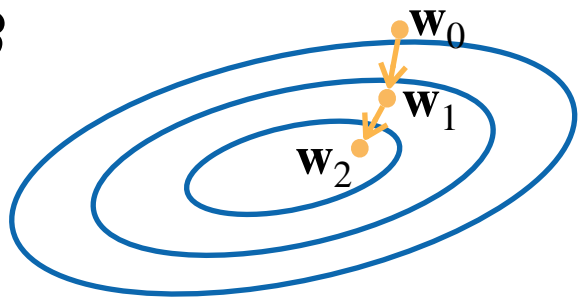
Basic Optimization



Gradient Descent

- Choose a staring point $\mathbf{w}_0$
- Repeat to update the weight *t=1,2,3*

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \frac{\partial \ell}{\partial \mathbf{w}}$$



  - Gradient: a direction that increases the value
  - Learning rate: a hyper-parameter specifies the step length

Mini-batch Stochastic Gradient Descent (SGD)

- Computing the gradient over the whole training data is too expensive
  - Takes minutes to hours for DNN models
- Randomly sample *b* examples $i_1, i_2, \ldots, i_b$ to approximate the gradient

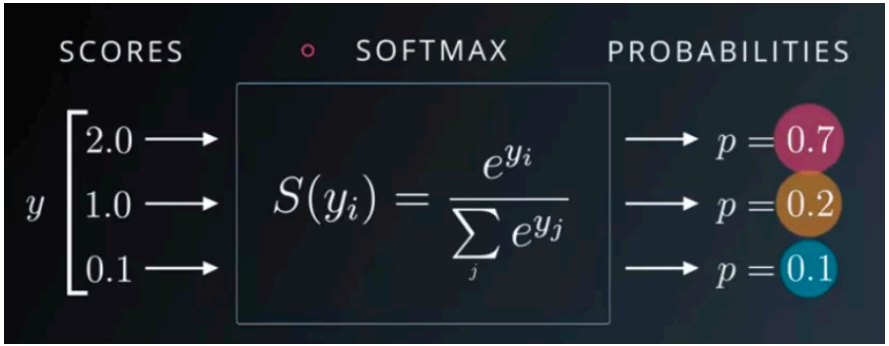
$$\frac{1}{b} \sum_{i \in I_b} \ell(\mathbf{x}_i, y_i, \mathbf{w})$$

  - *b* is the batch size, another important hyper-parameters

Summarize

- Problem: estimate a real value
- Model: $y = \langle \mathbf{w}, \mathbf{x} \rangle + b$
- Loss: square loss $\ell(y, \hat{y}) = \left(y - \hat{y}\right)^2$
- Learn by mini-batch SGD
  - Choose a starting point
  - Repeat
    - Compute gradient
    - Update parameters

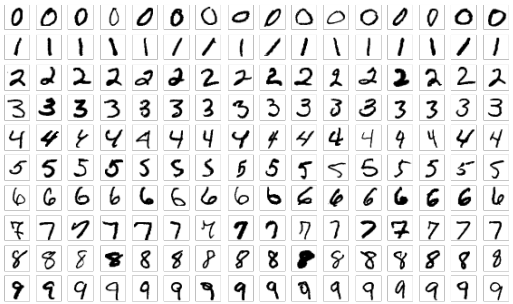
Softmax Regression



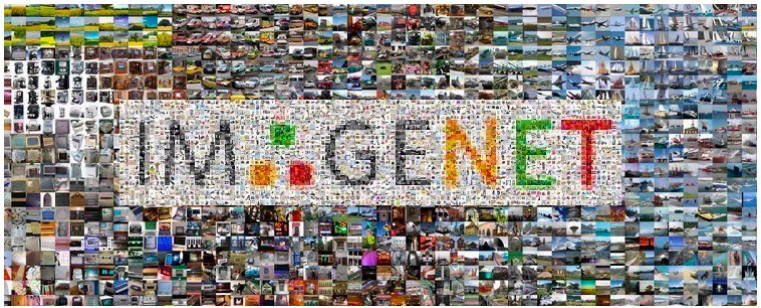
Regression vs Classification

- Regression estimates a continuous value
- Classification predicts a discrete category

MNIST: classify hand-written digits (10 classes)

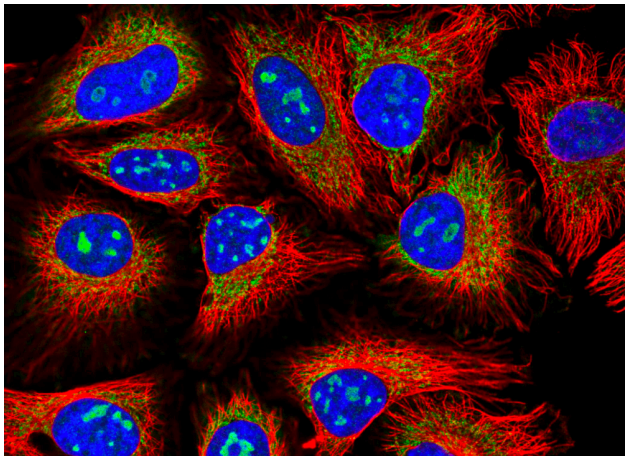


ImageNet: classify nature objects (1000 classes)



Various Classification Tasks at Kaggle

- Classify human protein microscope images into 28 categories



```
0.  Nucleoplasm
1.  Nuclear membrane
2.  Nucleoli
3.  Nucleoli fibrillar
4.  Nuclear speckles
5.  Nuclear bodies
6.  Endoplasmic reticu
7.  Golgolgi apparatus
8.  Peroxisomes
9.  Endosomes
10.  Lysosomes
11.  Intermediate fila
12.  Actin filaments
13.  Focal adhesion si
14.  Microtubules
15.  Microtubule ends
16.  Cytokinetic bridg
```

https://www.kaggle.com/c/human-protein-atlas-image-classification

Various Classification Tasks at Kaggle

- Classify malware into 9 categories

https://www.kaggle.com/c/malware-classification

Various Classification Tasks at Kaggle

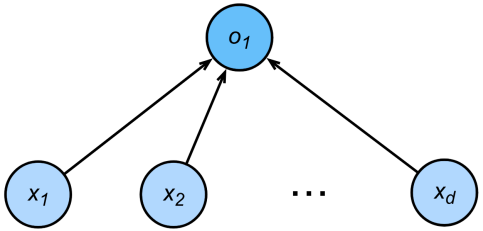
- Classify toxic Wikipedia comments into 7 categories

comment_text	toxic	severe_toxic	obsc
Explanation\nWhy the edits made under my usern...	0	0	0
D'aww! He matches this background colour I'm s...	0	0	0
Hey man, I'm really not trying to edit war. It...	0	0	0
"\nMore\nI can't make any real suggestions on ...	0	0	0
You, sir, are my hero. Any chance you remember...	0	0	0

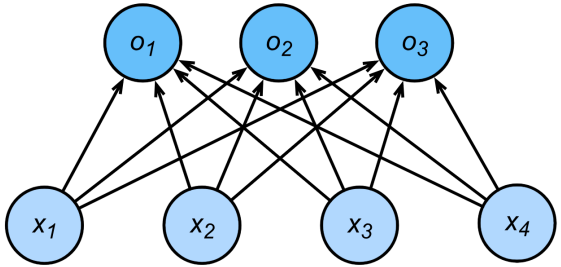
https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

From Regression to Multi-class Classification

- Single continuous output



- Output a confidence score for each category



- Predict the category

$$\arg\max_i (o_1, o_2, o_3)$$

  - *max* is not differentiable
  - define the loss function

Softmax

$$\text{softmax}([x_1, x_2, \ldots, x_n]^T) = \left[\frac{e^{x_1}}{\sum_{i=1}^{n} e^{x_i}}, \frac{e^{x_2}}{\sum_{i=1}^{n} e^{x_i}}, \ldots, \frac{e^{x_n}}{\sum_{i=1}^{n} e^{x_i}} \right]$$

- Use exp to get positive numbers
- Divide by the sum to obtain a probability distribution

Example: given scores [1, -1, 2]

softmax: [0.26, 0.04, 0.7]

Use Square Loss?

- One-hot encoding of label *y*

$$\mathbf{y} = [y_1, y_2, \ldots, y_n]^T, \quad y_i = \begin{cases} 1 & \text{if } i = y \\ 0 & \text{otherwise} \end{cases}$$

- We want the softmax output close to $\mathbf{y}$
- Use square loss for y = [0, 0, 1]

Softmax output	Loss
[0.3, 0, 0.7]	0.18
[0.17, 0.17, 0.66]	0.173

Cross Entry Loss

- Both one-hot encoding and softmax output are probability distribution
- Cross entropy is commonly used to compare distributions

$$H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$

  - If $y = i$, then $H(\mathbf{y}, \hat{\mathbf{y}}) = -\log(\hat{y}_i)$

Linear Model vs Logistic Regression

Problem	single value regression	k-class classification
Model	$\langle \mathbf{w}, \mathbf{x} \rangle + b$ $\mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}$	$\text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})$ $\mathbf{W} \in \mathbb{R}^{k \times n}, \mathbf{b} \in \mathbb{R}^k$
Loss	Squared loss	Cross-entropy loss