

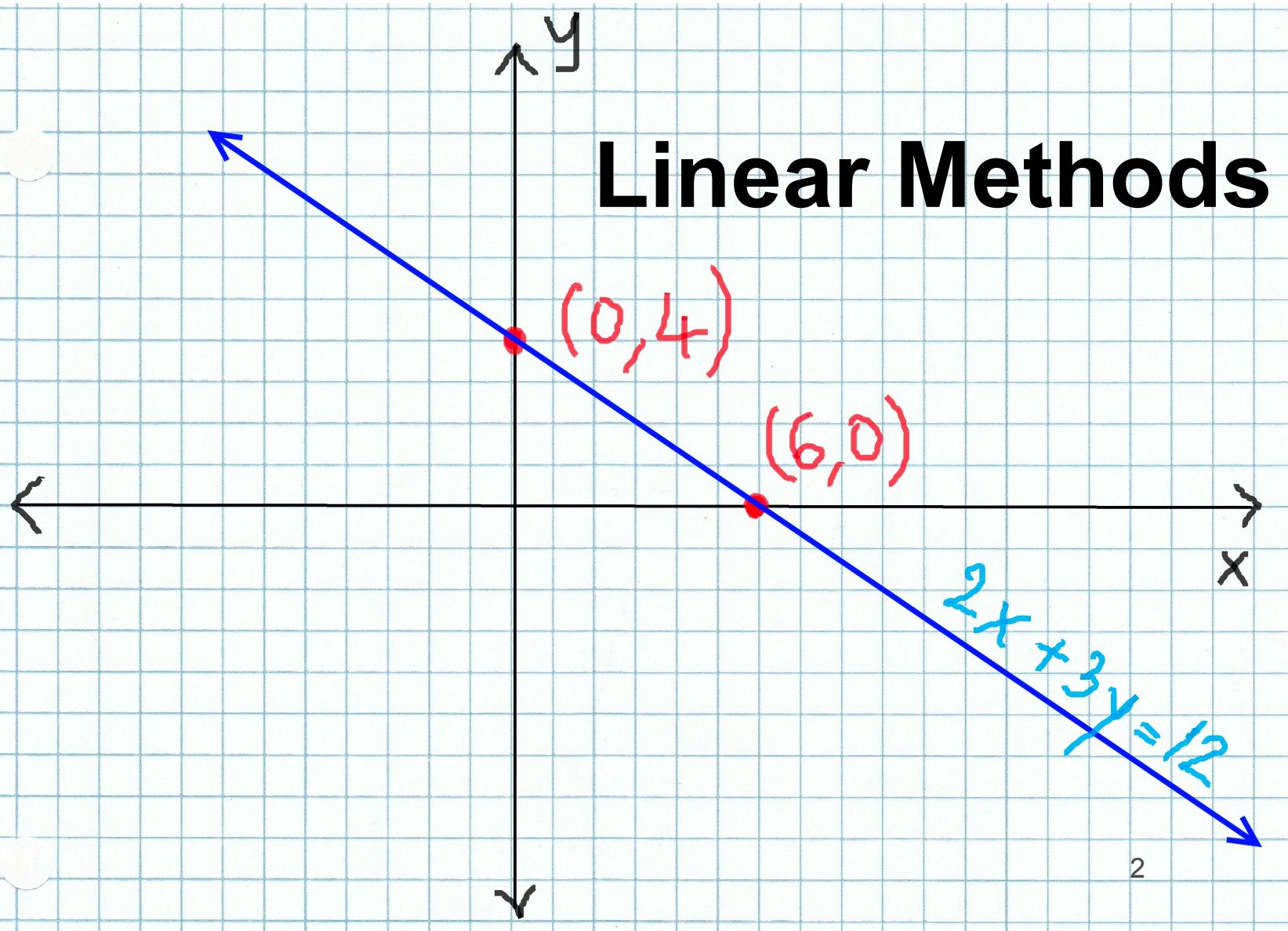
Introduction to Deep Learning

4. Linear Regression, Basic Optimization

MGMT 735 DEEP LEARNING IN FINANCE

SLIDES FROM Alex Smola and Mu Li

Linear Methods



House Buying 101

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

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

Interior Features

Bedroom Information

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

3

Multi-Unit Information

- # of Stories: 2

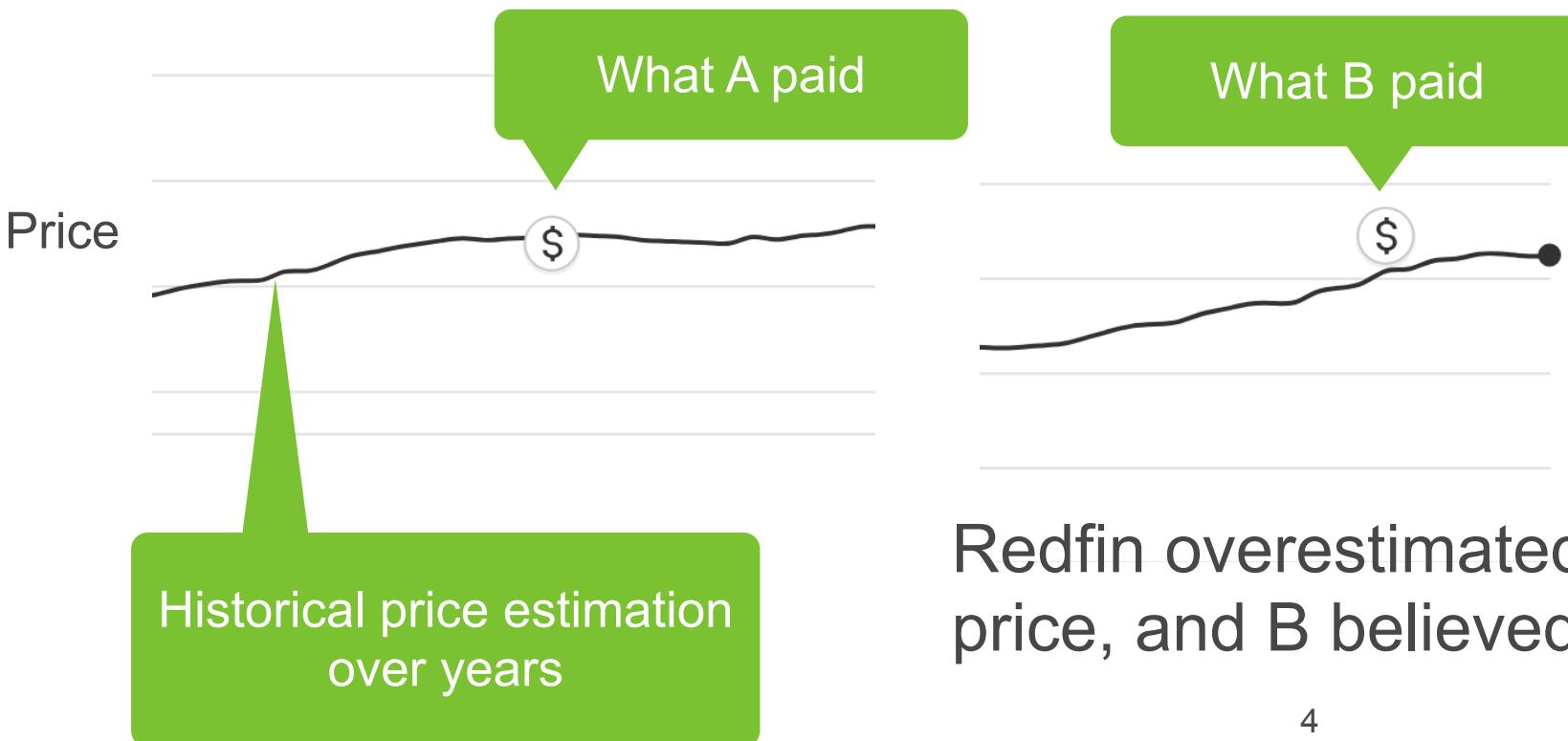
School Information

- Elementary School: El Carmelo El
- Elementary School District: Palo A
- Middle School: Jane Lathrop Stan
- High School: Palo Alto High
- High School District: Palo Alto Un

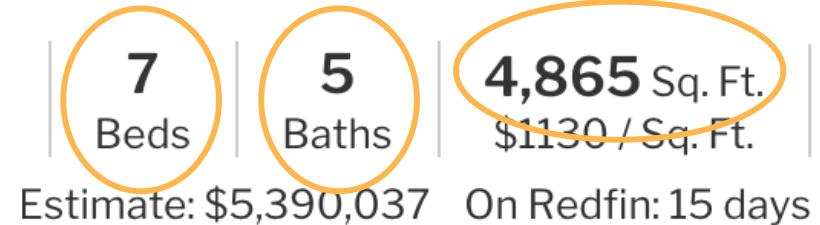
- Kitchen Description: Countertop, Dishwasher, Garbage Disposal, Hood, Island with Sink, Microwave, Over

House Price Prediction

Very important, that's real money...



A Simplified Model



- Assumption 1

The key factors impacting the prices are #Beds, #Baths, Living Sqft, denoted by x_1, x_2, x_3

- Assumption 2

The sale price is a weighted sum over the key factors

$$y = w_1x_1 + w_2x_2 + w_3x_3 + b$$

Weights and bias are determined later

Linear Model

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

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

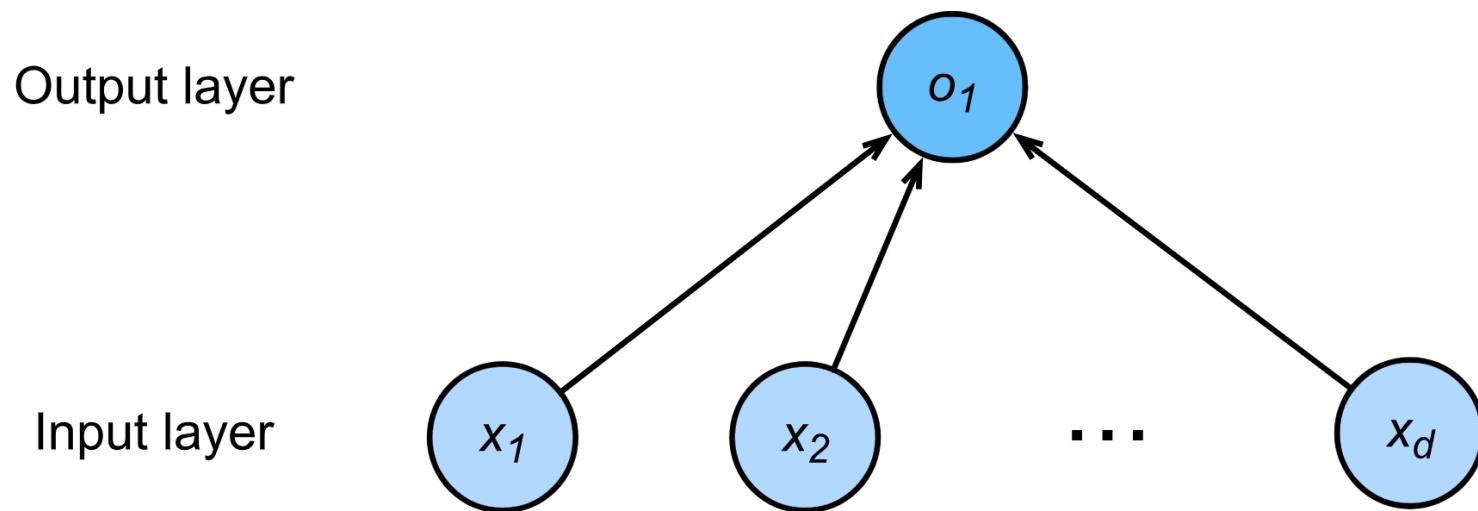
- The output is a weighted sum of the inputs

$$y = w_1x_1 + w_2x_2 + \dots + w_nx_n + b$$

Vectorized version

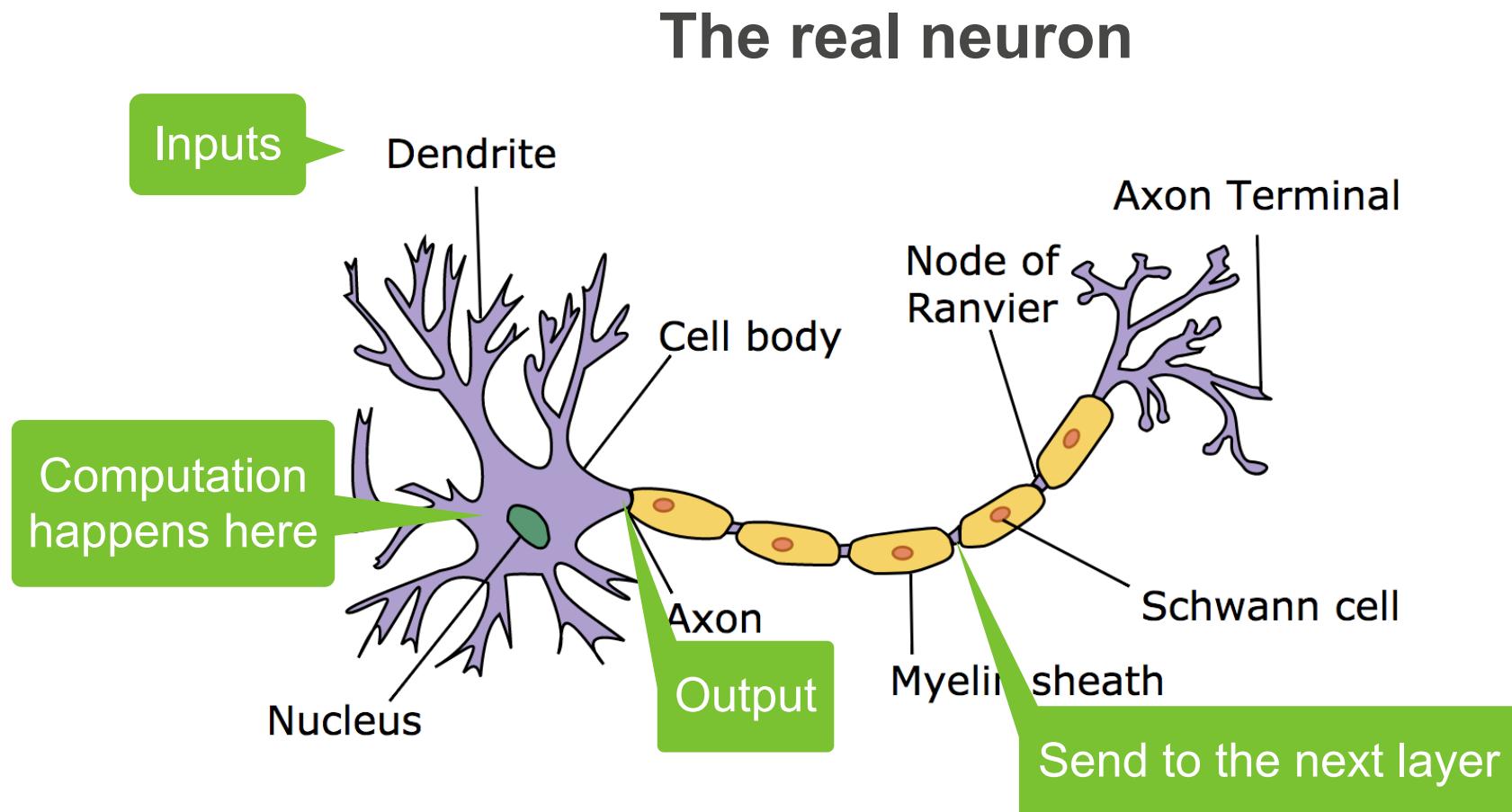
$$y = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

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 true value, and \hat{y} the estimated value, we can compare the loss

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

It is called squared loss

Training Data

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

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

Learn Parameters

- Training loss

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

- 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

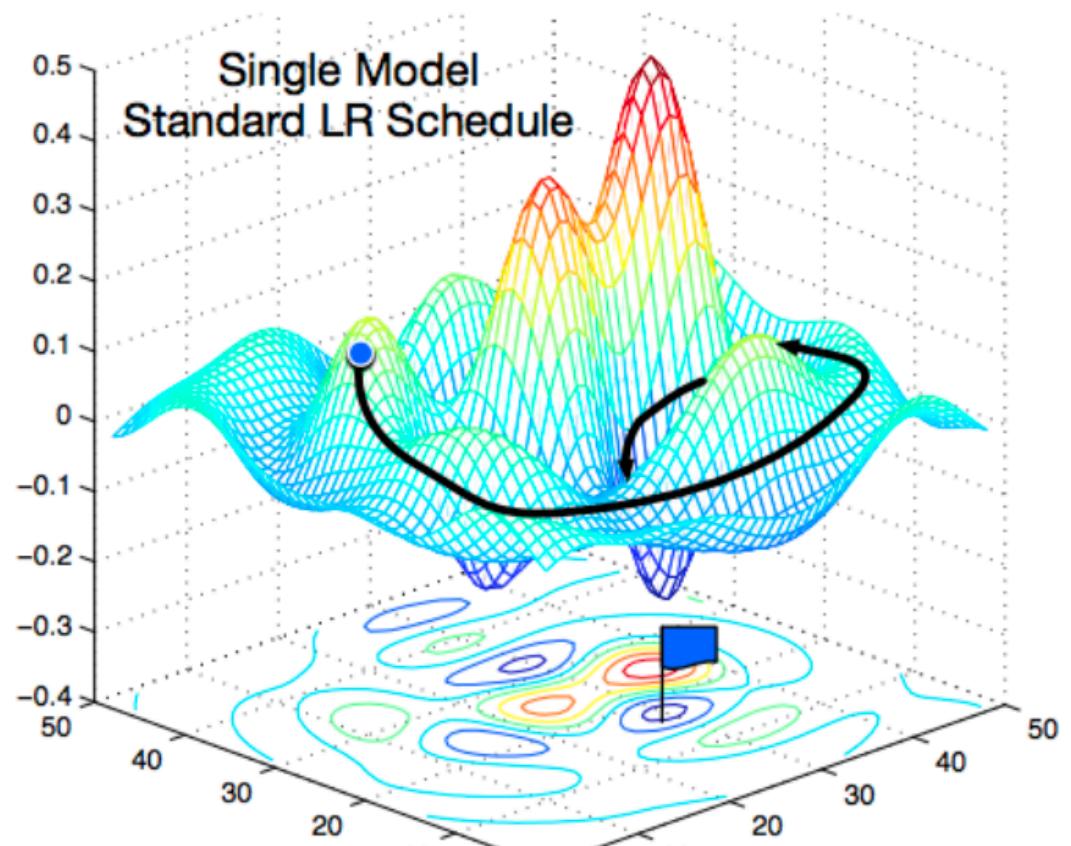
- Add bias into weights by $\mathbf{X} \leftarrow [\mathbf{X}, \mathbf{1}] \quad \mathbf{w} \leftarrow \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$

$$\ell(\mathbf{X}, \mathbf{y}, \mathbf{w}) = \frac{1}{n} \| \mathbf{y} - \mathbf{X}\mathbf{w} \|^2 \quad \frac{\partial}{\partial \mathbf{w}} \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}) = \frac{2}{n} (\mathbf{y} - \mathbf{X}\mathbf{w})^T \mathbf{X}$$

- Loss is convex, so the optimal solutions satisfies

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}} \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}) &= 0 \\ \Leftrightarrow \frac{2}{n} (\mathbf{y} - \mathbf{X}\mathbf{w})^T \mathbf{X} &= 0 \\ \Leftrightarrow \mathbf{w}^* &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X} \mathbf{y} \end{aligned}$$

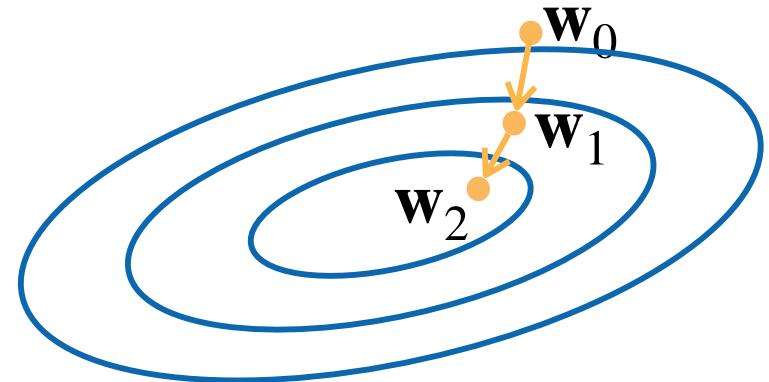
Basic Optimization



Gradient Descent

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

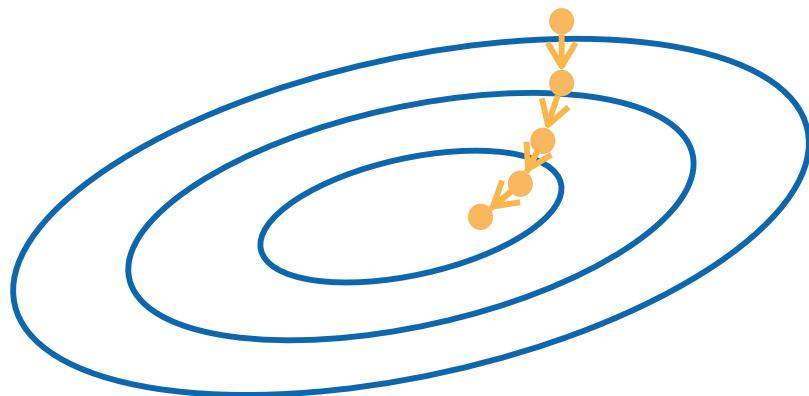
$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \frac{\partial \ell}{\partial \mathbf{w}_{t-1}}$$



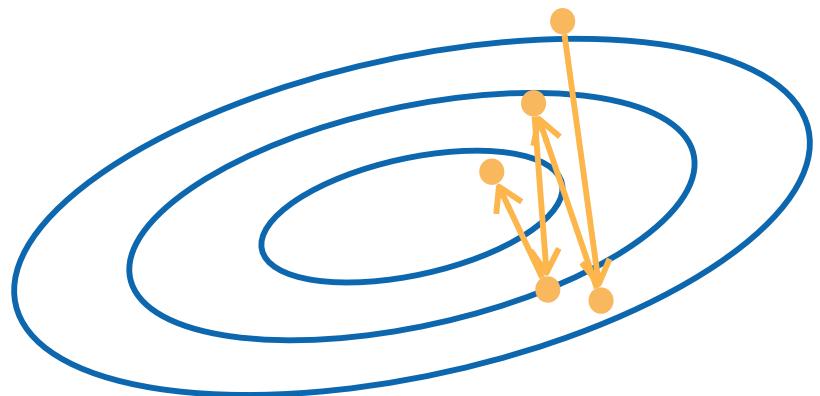
- Gradient: a direction that increases the value
- Learning rate: a hyper-parameter specifies the step length

Choose a Learning Rate

Not too small



Not too big



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, \dots, i_b to approximate the loss

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

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

Choose a Batch Size

Not too small

Workload is too small,
hard to fully utilize
computation resources

Not too big

Memory issue
Waste computation, e.g.
when all x_i are identical

Summary

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