

CS5228: Knowledge Discovery and Data Mining

Lecture 7 — Classification & Regression III

Course Logistics — Update



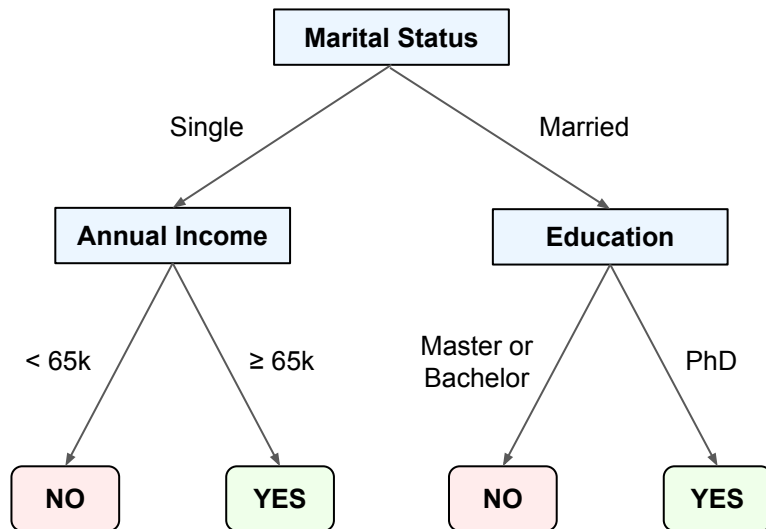
Quick Recap — Decision Trees

- Decision Trees

- Flowchart-like structure mapping input features to output labels or values
- Applicable to classification & regression tasks
- Support for categorical & numerical features
- Typically quite interpretable

- Building Decision Trees

- Greedy algorithm iteratively finding the best splits
- Best split = split that minimizes impurity

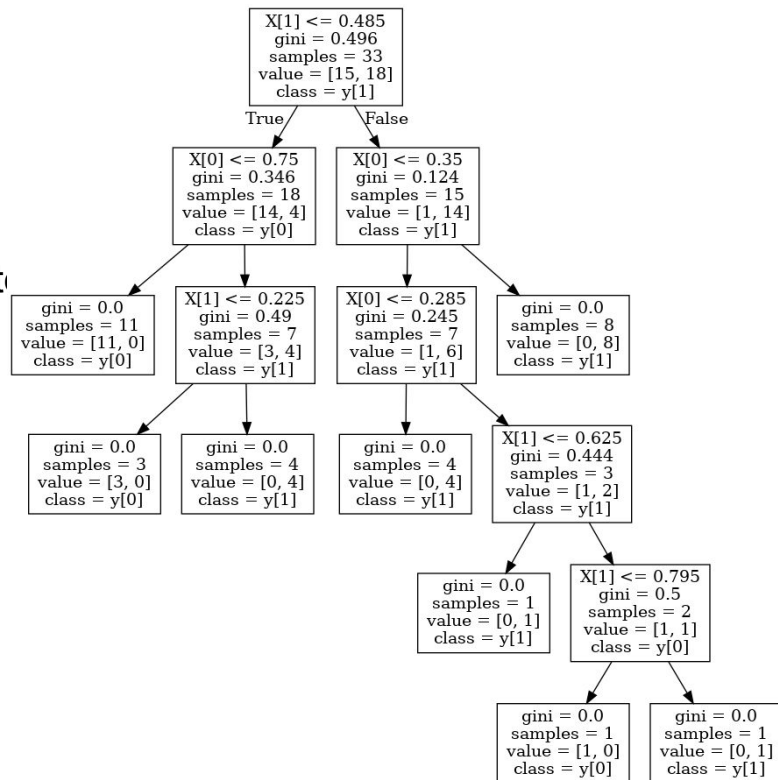


Quick Recap — Decision Trees

- Challenges

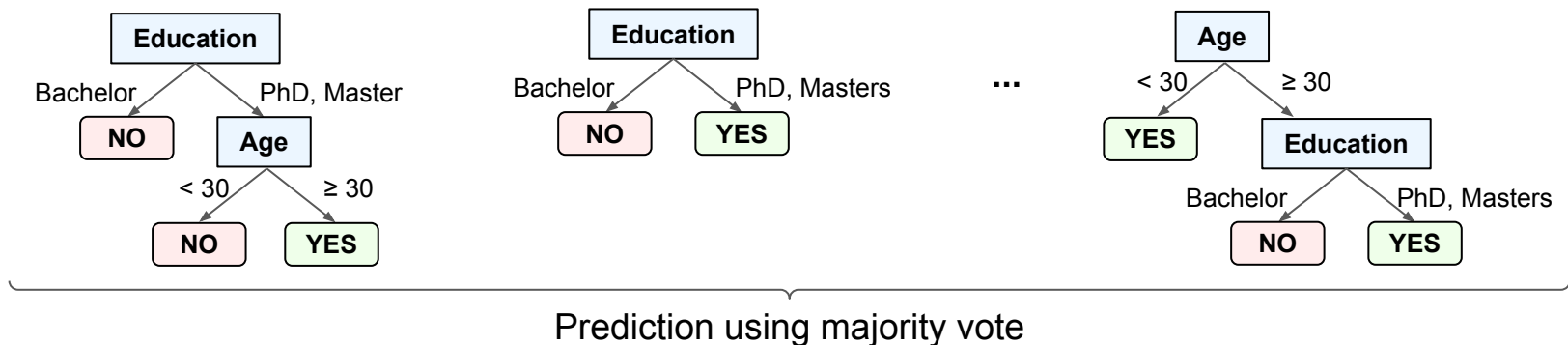
- Sensitive to small changes in training data
→ High variance
- High chance of overfitting in case of maximum/complete
→ Pruning of tree to avoid overfitting
- In practice, Decision Trees do not show state-of-the-art performance

→ Tree Ensemble methods



Quick Recap — Tree Ensembles

- Tree ensembles: Construct many decision trees and combine their predictions
 - Pros: Higher accuracy, lower variance
 - Cons: Lower interpretability, longer training time
- Ensembles of independent models:
 - Bagging — Train decision trees over re-sampled training data (bootstrap sampling)
 - Random Forests — bootstrap sampling + feature sampling



Quick Recap — Tree Ensembles

- Ensembles of dependent models (boosting methods)
 - Sequential training of decision trees
 - Next tree tries to improve errors of previous trees
 - Trees have different amount of say in predictions
- Two introduced boosting methods:
 - AdaBoost — resample training data to favour previously misclassified samples
 - Gradient Boosted Trees — start with initial prediction, improve based on predicted errors

Outline

- **Linear Models**
 - Basic setup
- **Linear Regression**
 - Problem formulation
 - Normal Equation (analytical solution)
 - Gradient Descent (iterative optimization)
 - Polynomial Linear Regression (overfitting, regularization)
 - Interpretation of Coefficients
- **Logistic Regression**
 - Problem formulation
 - Gradient Descent

Linear Models

Age	Education	Marital Status	Income Level	Credit Approval	Credit Limit
23	Masters	Single	Mid	No	\$S5,000
35	College	Married	High	Yes	\$S7,000
26	Masters	Single	High	No	\$S9,000
...

- Basic setup

- Dataset of n samples $\{(x_i, y_i)\}_{i=1}^n$
- Input data with d features $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$

- Assumption

- There exists linear relationship between x_i and dependent variable y_i

$$\hat{y}_i = h_{\theta}(x_i) = f(\theta_0 x_{i0} + \theta_1 x_{i1} + \theta_2 x_{i2} + \dots + \theta_d x_{id})$$

Predicted value which
is hopefully close to y_i

$$\theta = \{\theta_0, \theta_1, \theta_2, \dots, \theta_d\}, \theta_i \in \mathbb{R}$$

Linear Models

- Vector representation

- Introduce constant feature x_{i0}

$$h_{\theta}(x_i) = f(\underbrace{\theta_0 x_{i0}}_{=1} + \theta_1 x_{i1} + \theta_2 x_{i2} + \dots + \theta_d x_{id})$$

- Represent x_i with new constant feature

$$x_i = (1, x_{i1}, x_{i2}, \dots, x_{id})$$

- Rewrite linear relationship using vectors representing x_i and θ

$$h_{\theta}(x_i) = f(\theta^T x_i)$$

Outline

- Linear Models
 - Basic setup
- Linear Regression
 - **Problem formulation**
 - Normal Equation (analytical solution)
 - Gradient Descent (iterative optimization)
 - Polynomial Linear Regression (overfitting, regularization)
 - Interpretation of Coefficients
- Logistic Regression
 - Problem formulation
 - Gradient Descent

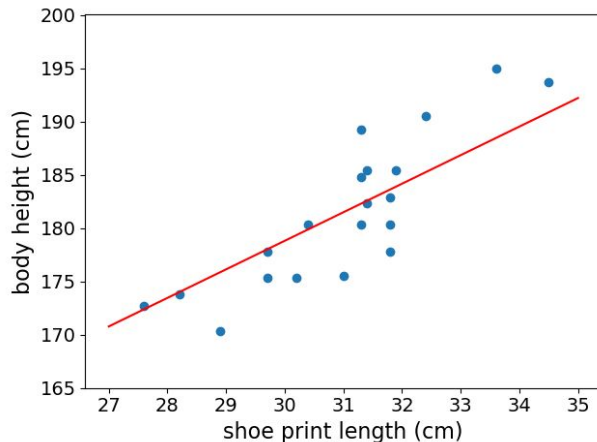
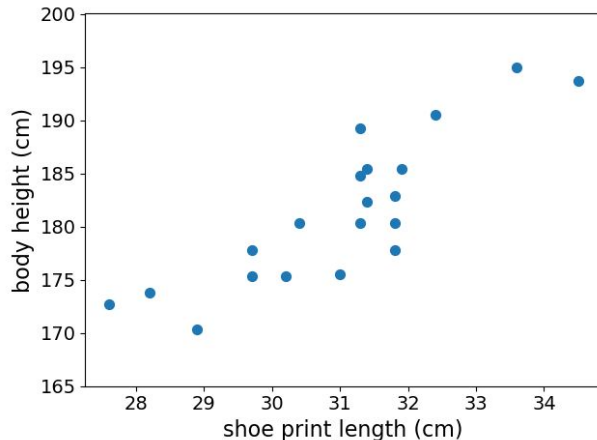
Linear Regression — Simple Example

- Crime scene investigation (CSI)

- Found a shoe print of size 32.2cm
- What is the estimated height of the suspect?

- Approach: Linear Regression

- Collect a dataset of (size, height)-pairs
- Quantify linear relationship between shoe print size and body height $\rightarrow h_{\theta}(\text{size}) = \theta_0 + \theta_1 \text{size}$
- Predict suspect's height via $\hat{y} = h_{\theta}(32.2)$



Linear Regression

- Regression → Real-valued predictions

- Function f is the identity function $f(x) = x$

$$\hat{y}_i = h_{\theta}(x_i) = f(\theta^T x_i) = \theta^T x_i$$

$$\hat{y}_i = \theta^T x_i$$

$$\hat{y} = X\theta$$

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1d} \\ 1 & x_{21} & x_{22} & \dots & x_{2d} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$$

Quick Quiz



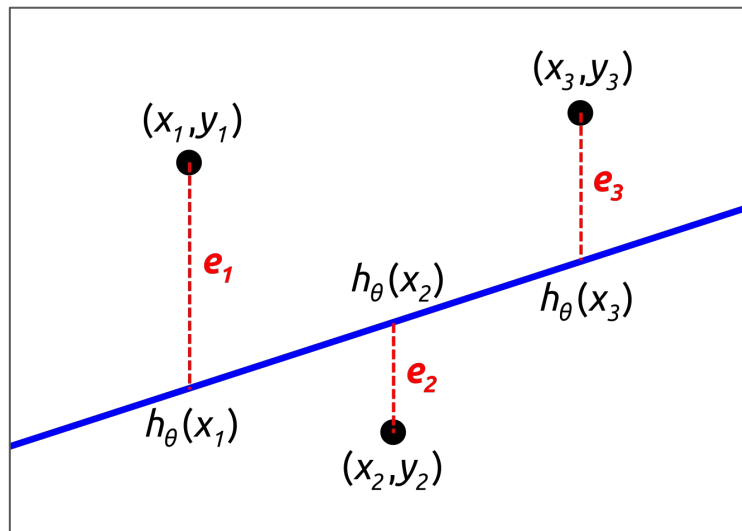
Linear Regression — Loss Function

- **Loss function** (also: cost function, error function)

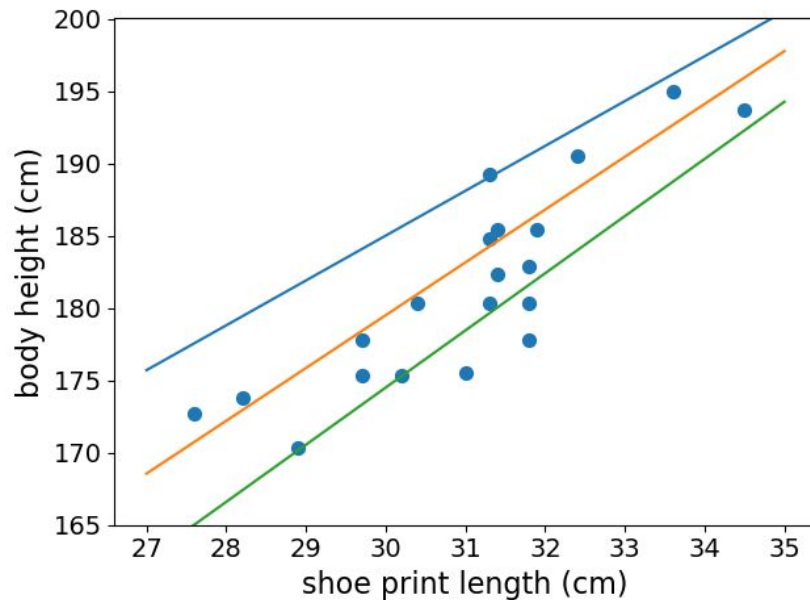
- Quantifies how good or bad a given set of values for θ is?
- Measures the difference between predictions \hat{y} and true values y

- Loss function for Linear Regression:
Mean Squared Error (MSE)

$$L = \frac{1}{n} \sum_{i=1}^n e_i^2 = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2$$



Losses for CSI Example



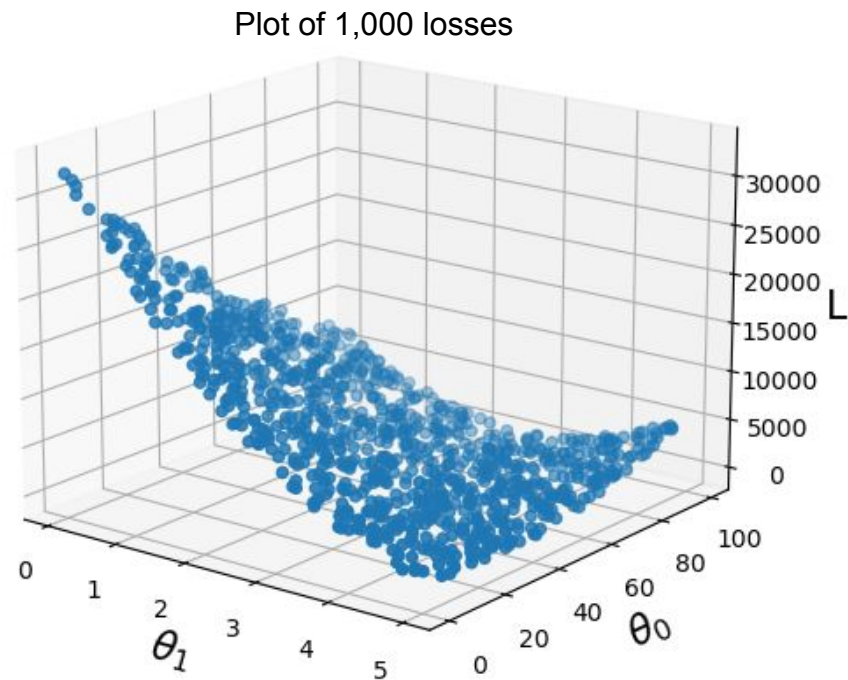
$$\begin{aligned}h_{blue} &= 92 + 3.10 \cdot x \\h_{orange} &= 69 + 3.61 \cdot x \\h_{green} &= 56 + 3.95 \cdot x \\h_{random} &= 100 - 5.00 \cdot x\end{aligned}$$

$$\begin{aligned}L_{blue} &= 57.47 \\L_{orange} &= \mathbf{12.12} \\L_{green} &= 20.83 \\L_{random} &= 56, 129.23\end{aligned}$$

→ How to find the best values for θ ?

Method 1: Random Search (the "stupid" way)

- Repeat "enough" times
 - Select random values for $\theta = \{\theta_0, \theta_1, \dots, \theta_d\}$
 - Calculate loss L for current θ
 - Return θ with smallest loss
 - Limitation:
 - Not practical beyond toy examples
- Don't do that! :)



Outline

- Linear Models
 - Basic setup
- Linear Regression
 - Problem formulation
 - **Normal Equation** (analytical solution)
 - Gradient Descent (iterative optimization)
 - Polynomial Linear Regression (overfitting, regularization)
 - Interpretation of Coefficients
- Logistic Regression
 - Problem formulation
 - Gradient Descent

Method 2: Find Minimum of L Analytically (the proper way)

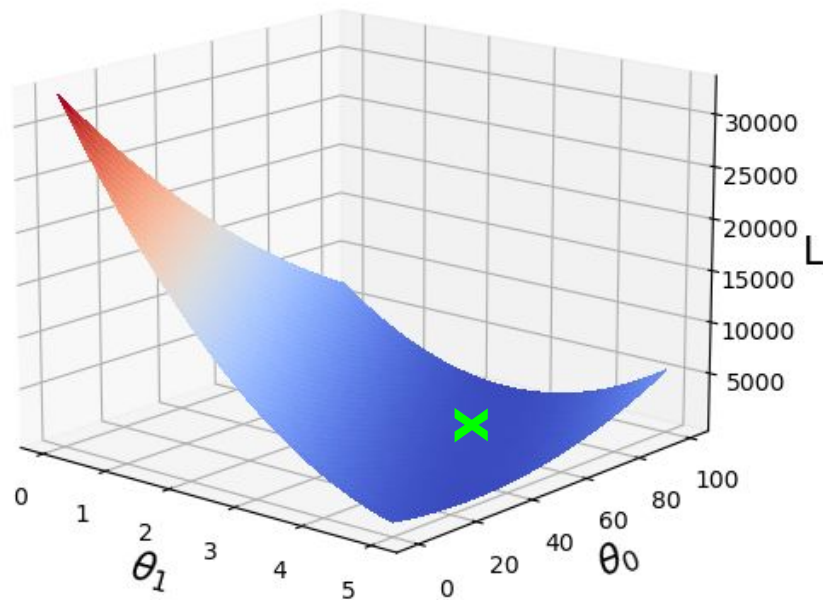
- Minimum of loss function $L \rightarrow$ Calculus to the rescue!

- Partial derivatives w.r.t. to all θ_i are 0

$$\frac{\partial L}{\partial \theta_0} = 0, \frac{\partial L}{\partial \theta_1} = 0, \dots, \frac{\partial L}{\partial \theta_d} = 0$$

- $d+1$ equations with $d+1$ unknowns

- (no need to check if minimum or maximum)



Method 2: Find Minimum of L Analytically (the proper way)

- Rewrite loss function L

- Vector representation mathematically more convenient to handle
- Avoids dealing with d equations

$$\begin{aligned} L &= \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \\ &= \frac{1}{n} \sum_{i=1}^n (\theta^T x_i - y_i)^2 \\ &= \frac{1}{n} \|X\theta - y\|^2 \end{aligned}$$

- Derive L w.r.t. to θ

- Using chain rule

$$\frac{\partial L}{\partial \theta} = \frac{2}{n} X^T (X\theta - y)$$

- Set $\partial L / \partial \theta$ to 0

$$\frac{2}{n} X^T (X\theta - y) \stackrel{!}{=} \vec{0}$$

Linear Regression — Normal Equation

- Solve for θ

$$\frac{2}{n}X^T(X\theta - y) = \vec{0}$$

$$X^T X \theta = X^T y$$

$$(X^T X)^{-1} X^T X \theta = (X^T X)^{-1} X^T y$$

$$\theta = (X^T X)^{-1} X^T y$$

$$\theta = X^\dagger y, \text{ with } X^\dagger = \underbrace{(X^T X)^{-1} X^T}_{\text{"pseudo inverse" of } X}$$

Pseudo Inverse X^\dagger

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

$$\begin{array}{c} X^T \\ (d+1) \times n \end{array} \begin{array}{c} X \\ n \times (d+1) \end{array} = \begin{array}{c} X^T X \\ (d+1) \times (d+1) \end{array} \longrightarrow \underbrace{\begin{array}{c} (X^T X)^{-1} \\ (d+1) \times (d+1) \end{array}^{-1} \begin{array}{c} X^T \\ (d+1) \times n \end{array}}_{(d+1) \times n}$$

- Performance analysis

- Most expensive operation: calculating the inverse of $(X^T X)^{-1}$
- Calculation of inverse depends on number of features d , not on number of data samples n
- Complexity of calculating inverse of a $d \times d$ matrix: $O(d^3)$

Quick Quiz



Quick Quiz



Outline

- Linear Models
 - Basic setup
- Linear Regression
 - Problem formulation
 - Normal Equation (analytical solution)
 - **Gradient Descent** (iterative optimization)
 - Polynomial Linear Regression (overfitting, regularization)
 - Interpretation of Coefficients
- Logistic Regression
 - Problem formulation
 - Gradient Descent

Method 2: Find Minimum of L Analytically (the proper way)

- Algorithm

- Construct matrix X and vector y from data
- Calculate pseudo inverse $X^\dagger = (X^T X)^{-1} X^T$
- Return $\theta = X^\dagger y$

- For the CSI example:

$$\theta = \begin{bmatrix} 20.0 & 620.2 \\ 620.0 & 19284.9 \end{bmatrix}^{-1} X^T y = \begin{bmatrix} 18.4 & -0.6 \\ -0.6 & 0.02 \end{bmatrix} X^T y = \begin{bmatrix} 69.5 \\ 3.6 \end{bmatrix}$$

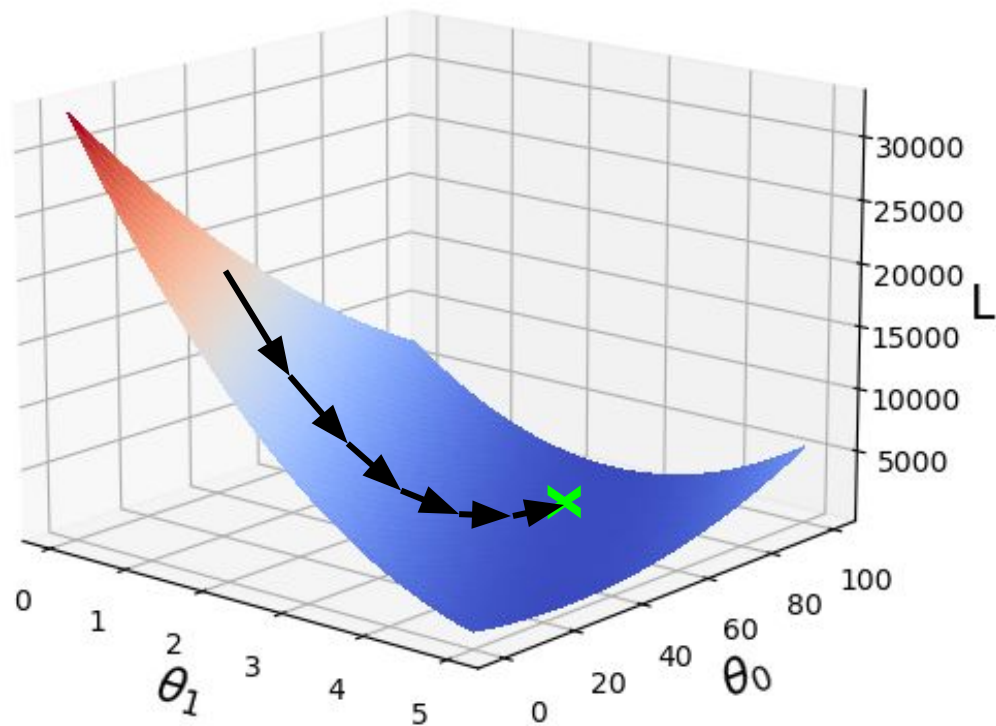
$$h_\theta(\text{size}) = 69.5 + 3.6 \text{size}$$

$$h_\theta(32.2) = 185.4$$

Method 3: Gradient Descent

- Core idea

- Start with a random setting of θ
- Adjust θ iteratively to minimize L



Gradient — Quick Refresher

- Gradient

- Vector of partial derivatives of a multivariable function (e.g., $\theta_0, \theta_1, \dots, \theta_d$)
- Partial derivative: slope w.r.t. to a single variable given a current set of values for all $\theta_0, \theta_1, \dots, \theta_d$
- Points in the direction of the steepest ascent

$$\nabla_{\theta} L = \frac{\partial L}{\partial \theta} = \begin{bmatrix} \frac{\partial L}{\partial \theta_0} \\ \frac{\partial L}{\partial \theta_1} \\ \frac{\partial L}{\partial \theta_2} \\ \vdots \\ \frac{\partial L}{\partial \theta_d} \end{bmatrix}$$

Gradient for CSI Example

Best value: $\theta_0 = 69.5$, $\theta_1 = 3.6$

- Assume $\theta_0 = 60$ and $\theta_1 = 4$

$$\nabla_{\theta} L = \frac{2}{n} X^T (X\theta - y) = \frac{2}{n} X^T \left(X \cdot \begin{bmatrix} 60 \\ 4 \end{bmatrix} - y \right) = \begin{bmatrix} 5.2 \\ 163.9 \end{bmatrix}$$

- Interpretation of $\nabla_{\theta} L = \begin{bmatrix} 5.2 \\ 163.9 \end{bmatrix}$
 - Both values positive: a small increase of θ_0 or θ_1 will increase the loss
 - A small change in θ_1 affects the loss more than the same change in θ_0
 - Absolute values of gradient not a direct indicator of how to update θ

Gradient Descent Algorithm

- Important concept: learning rate
 - Scaling factor for gradient (typical range: 0.01 - 0.0001)

Input : data (X, y) , loss function L , learning rate η

Initialization : Set θ to random values

while true :

 Calculate gradient $\nabla_{\theta} L$

$\theta \leftarrow \theta - (\eta \cdot \nabla_{\theta} L)$

In practice: stop loop
when θ converges

Gradient Descent for CSI Example

- Input
 - $\eta = 0.0001$
- Initialization
 - $\theta_0 = 100$
 - $\theta_1 = 50$

```
[001] theta0=99.70619, theta1=40.86448, loss=2163825.04200
[002] theta0=99.46909, theta1=33.49256, loss=1409025.31406
[003] theta0=99.27776, theta1=27.54378, loss=917521.53239
[004] theta0=99.12336, theta1=22.74340, loss=597468.46605
[005] theta0=98.99877, theta1=18.86972, loss=389059.15347
[006] theta0=98.89823, theta1=15.74385, loss=253349.02868
[007] theta0=98.81709, theta1=13.22143, loss=164978.51518
[008] theta0=98.75161, theta1=11.18595, loss=107434.18922
[009] theta0=98.69877, theta1=9.54342, loss=69962.98624
[010] theta0=98.65613, theta1=8.21798, loss=45562.82115
[011] theta0=98.62172, theta1=7.14841, loss=29674.13840
[012] theta0=98.59395, theta1=6.28532, loss=19327.88711
[013] theta0=98.57153, theta1=5.58885, loss=12590.70710
[014] theta0=98.55344, theta1=5.02683, loss=8203.65008
[015] theta0=98.53884, theta1=4.57330, loss=5346.92525
[016] theta0=98.52706, theta1=4.20733, loss=3486.70856
[017] theta0=98.51754, theta1=3.91201, loss=2275.38917
[018] theta0=98.50986, theta1=3.67370, loss=1486.61298
[019] theta0=98.50366, theta1=3.48140, loss=972.98470
[020] theta0=98.49866, theta1=3.32622, loss=638.52480
```

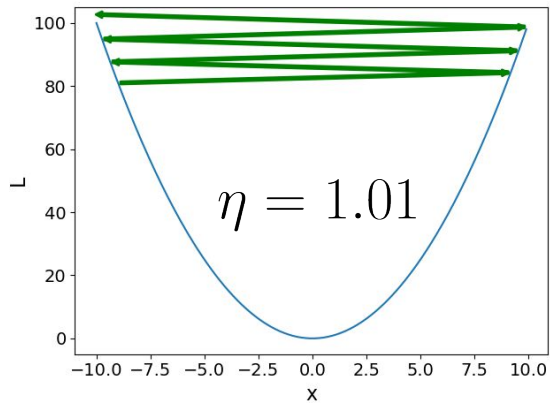
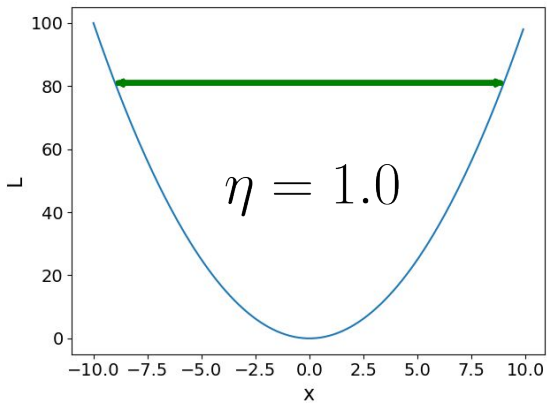
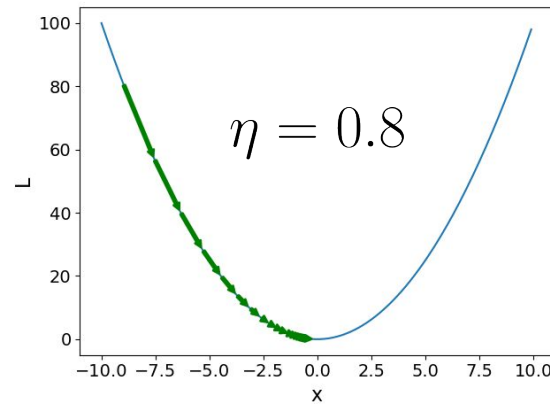
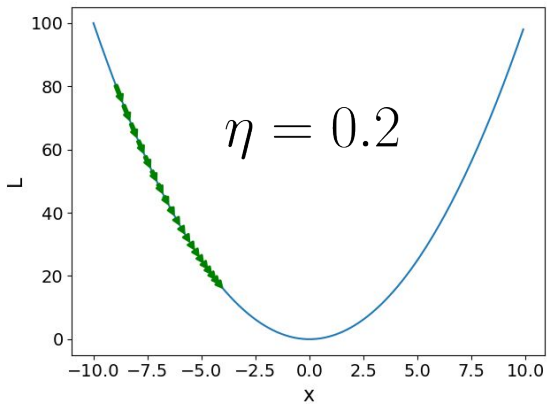
...

```
[098] theta0=98.47656, theta1=2.67760, loss=14.17658
[099] theta0=98.47655, theta1=2.67760, loss=14.17658
[100] theta0=98.47653, theta1=2.67760, loss=14.17658
```

Quick Quiz: Why not just increase the learning rate to speed things up?

Effects of Learning Rate for

$$L = x^2, \quad \frac{\partial L}{\partial x} = 2x, \quad 20 \text{ steps}$$

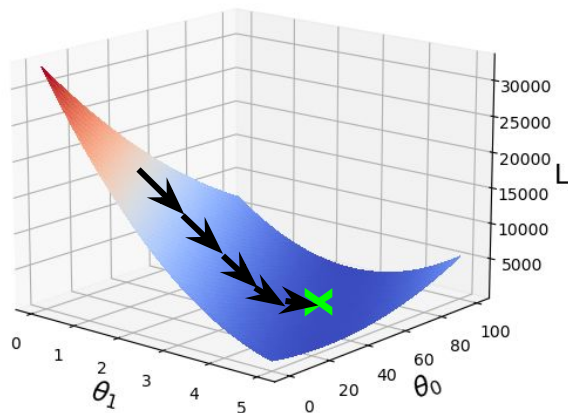


Gradient Descent — Variations

- (Basic) Gradient Descent
 - Calculate gradient und update θ for whole dataset
- Stochastic Gradient Descent (SGD)
 - Calculate gradient und update θ for each data sample
- Mini-batch Gradient Descent
 - Calculate gradient und update θ for batches of sample
 - e.g., batch = 64 data samples
 - In practice often referred to as SGD

Gradient Descent — Variations

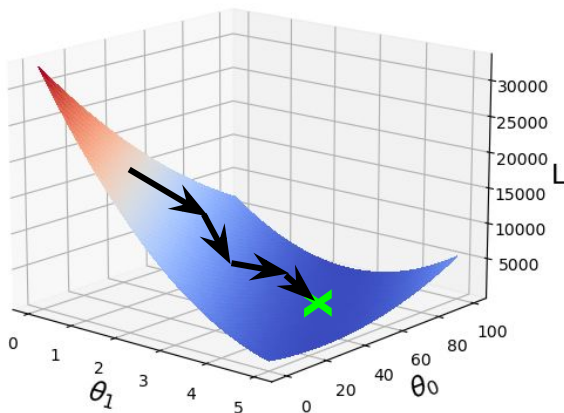
Gradient Descent



Gradient averaged over all data items

- Smooth descent
- Small(er) gradients
- Small(er) update steps

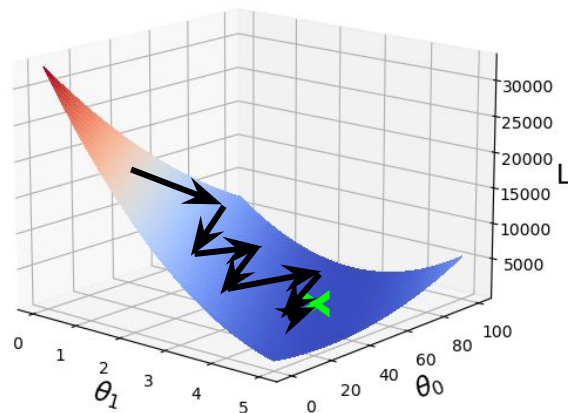
Mini-Batch Gradient Descent



Gradient averaged over some data items

- Well, "somewhere in-between" :)

Stochastic Gradient Descent



Gradient for each data item considered

- Choppy descent
- Large(r) gradients
- Large(r) steps

Normal Equation vs. Gradient Descent

- Gradient Descent

- Works well even if d is large
- Works even if $X^T X$ is non-invertible
- Iterative process; may not find optimal solution in practice
- Learning rate a critical hyperparameter

- Normal Equation

- Finds optimal solutions
- Non-iterative; no need of learning rate
- Calculation of $(X^T X)^{-1}$ in $O(d^3)$

Quick Quiz



Quick Quiz



Outline

- Linear Models
 - Basic setup
- Linear Regression
 - Problem formulation
 - Normal Equation (analytical solution)
 - Gradient Descent (iterative optimization)
 - **Polynomial Linear Regression** (overfitting, regularization)
 - Interpretation of Coefficients
- Logistic Regression
 - Problem formulation
 - Gradient Descent

Polynomial Linear Regression

- Linear Regression ➔ line / plane / hyperplane
- Polynomial Linear Regression
 - Allows to capture nonlinear relationships between X and y
 - Polynomial regression model for 1 input feature

Still linear in θ !

$$\hat{y}_i = \theta_0 1 + \theta_1 x_i + \theta_2 x_i^2 + \dots + \theta_p x_i^p$$

- Matrix representation (again, 1 input feature!)

$$X^{(1)} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

$$X^{(2)} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix}$$

$$X^{(3)} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}$$

$$X^{(p)} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & \dots & x_1^p \\ 1 & x_2 & x_2^2 & x_2^3 & \dots & x_2^p \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & \dots & x_n^p \end{bmatrix}$$

Polynomial Linear Regression

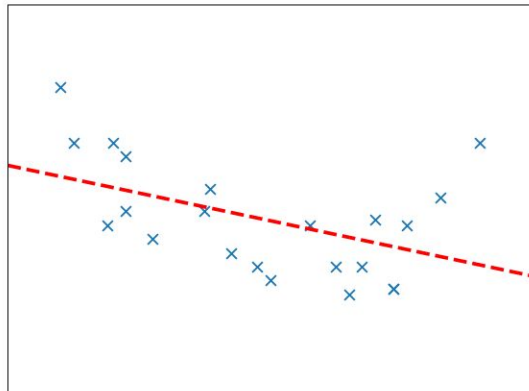
- Example: 1 input feature, 3 data samples

$$X^{(1)} = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} \quad X^{(2)} = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 1 & 1 \end{bmatrix} \quad X^{(3)} = \begin{bmatrix} 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

- Polynomial terms "look the same" as additional features
- Finding best θ using the Normal Equation or Gradient Descent

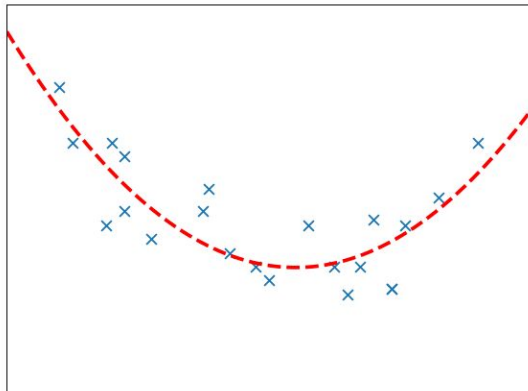
Polynomial Linear Regression — Example

$p = 1$



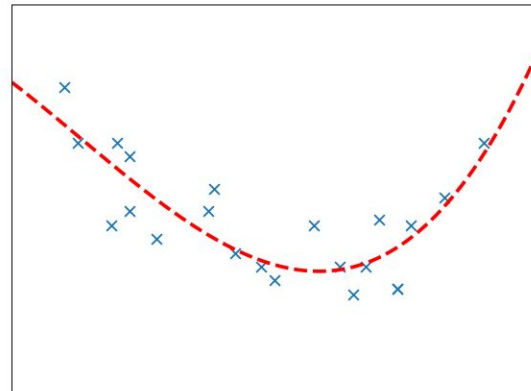
Underfitting

$p = 2$



Good fit

$p = 3$



Overfitting?

Polynomial Linear Regression — Overfitting

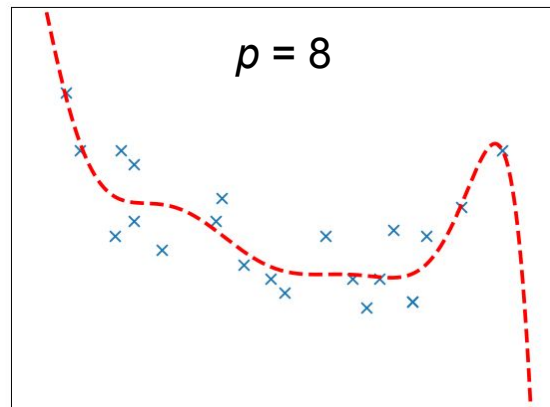
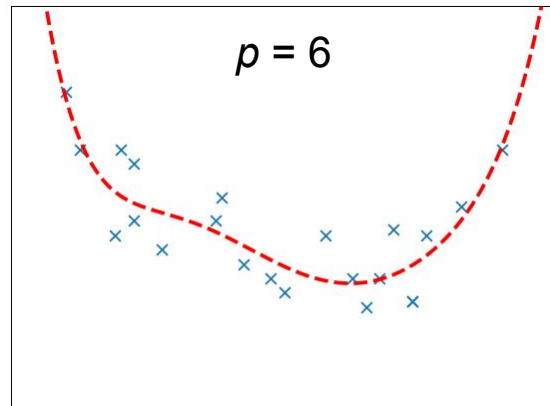
- Increasing degree of polynomial p
 - More capacity to capture nonlinear relationships
 - Much higher sensitivity to noise and outliers
- Countermeasure: **Regularization**
 - Extend loss function to "punish" large values of θ

$$L = \frac{1}{n} \|X\theta - y\|^2 + \lambda \frac{1}{n} \|\theta\|_2^2$$

Regularization parameter

$$\|\theta\|_2^2 = \sum_{i=1}^d \theta_i^2$$

Note: excludes θ_0 !



Polynomial Linear Regression — Minimizing Loss L

- Normal Equation

$$\theta = (X^T X + \lambda \overset{(d+1) \times (d+1)}{\begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}})^{-1} X^T y$$

- Gradient Descent

$$\nabla_{\theta} L = \frac{2}{n} X^T (X\theta - y) + \lambda \frac{2}{n} \theta$$

Polynomial Linear Regression — More than 1 Feature

- Polynomial of degree $p=2$ and two input features ($d=2$)

$$\hat{y}_i = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \theta_3 x_{i1}^2 + \theta_4 x_{i2}^2 + \underbrace{\theta_5 x_{i1} x_{i2}}$$

interaction terms (cross terms)

- Polynomial of degree $p=3$ and two input features ($d=2$)

$$\hat{y}_i = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \theta_3 x_{i1}^2 + \theta_4 x_{i2}^2 + \theta_5 x_{i1}^3 + \theta_6 x_{i2}^3 + \\ \theta_7 x_{i2}^2 x_{i1} + \theta_8 x_{i2} x_{i1}^2 + \theta_9 x_{i2} x_{i1}$$

- Polynomial of degree $p=2$ and two input features ($d=3$)

$$\hat{y}_i = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \theta_3 x_{i3} + \theta_4 x_{i1}^2 + \theta_5 x_{i2}^2 + \theta_6 x_{i3}^2 \\ \theta_7 x_{i2} x_{i1} + \theta_8 x_{i3} x_{i1} + \theta_9 x_{i3} x_{i2}$$

Polynomial Linear Regression — More than 1 Feature

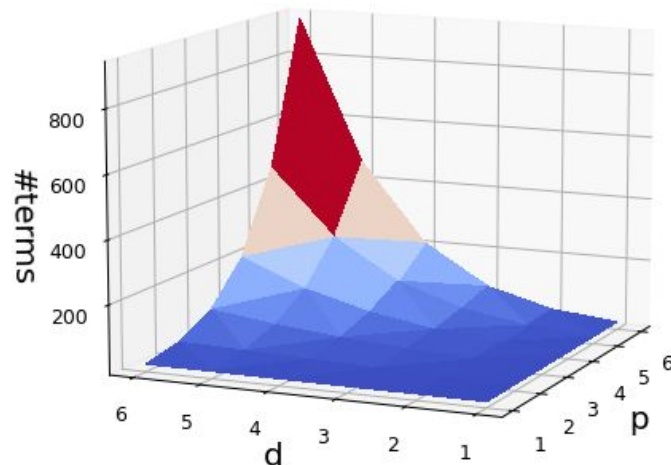
- Number of terms in multivariate polynomial given given p , d

$$\theta_i, \quad 0 \leq i \leq M, \quad \text{with } M = \binom{p+d}{p}$$

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

- **Practical considerations**

- Limited to small number of features and small polynomial degrees
- In principle, terms can be dropped (e.g., all interaction terms)



} **risk of overfitting + low interpretability**

} **justification typically not obvious**

Outline

- Linear Models
 - Basic setup
- Linear Regression
 - Problem formulation
 - Normal Equation (analytical solution)
 - Gradient Descent (iterative optimization)
 - Polynomial Linear Regression (overfitting, regularization)
 - **Interpretation of Coefficients**
- Logistic Regression
 - Problem formulation
 - Gradient Descent

Linear Regression — Interpretation of θ_i

- Example: prediction of house prices

$$\left. \begin{aligned} price &= \theta_0 + \theta_1(\#rooms) + \theta_2(area) + \theta_3(floor) + \dots \\ price' &= \theta_0 + \theta_1(\#rooms + 1) + \theta_2(area) + \theta_3(floor) + \dots \end{aligned} \right\} \Delta(price) = price' - price = \theta_1$$

- Interpretation

- Change of value of feature i by 1 unit \rightarrow change of output value by θ_i
- Assumption: all other features values remain the same

\rightarrow What about normalizing the data?

Data Normalization — Yes or No? (standardization, min-max scaling)

- Data normalization does not affect model performance

(assuming basic Linear Regression without regularization)

- In favor of "No"

- Preserves unit of feature $i \rightarrow$ direct interpretation of θ_i
- Better for comparing θ_i for the same features across different datasets

- In favour of "Yes"

- When using regularization
- When using Polynomial Linear Regression
- Better for comparing θ_i within a model (e.g., $\theta_i > \theta_j \rightarrow$ feature i more important than feature j)

Quick Quiz



Overview

- Linear Models
 - Basic setup
- Linear Regression
 - Problem formulation
 - Normal Equation (analytical solution)
 - Gradient Descent (iterative optimization)
 - Polynomial Linear Regression (overfitting, regularization)
 - Interpretation of Coefficients
- Logistic Regression
 - **Problem formulation**
 - Gradient Descent

Logistic Regression — Simple Example

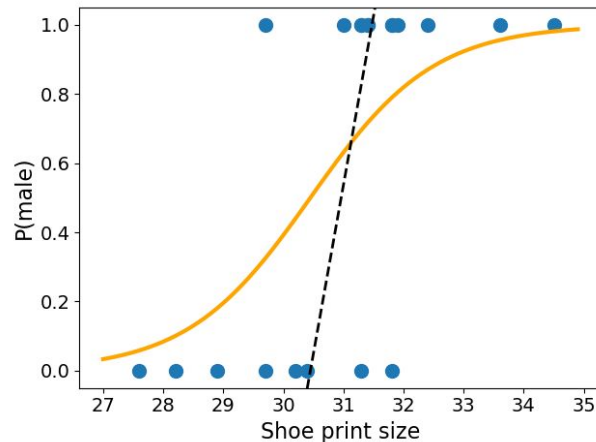
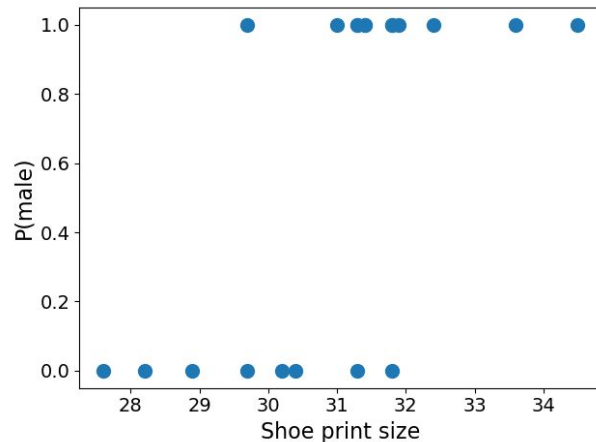
- Crime scene investigation (CSI)

- Found a shoe print of size 32.2cm
- Is the suspect a male or not?

- Approach: Logistic Regression for binary classification $y_i \in \{0, 1\}$

- Collect a dataset of (size, sex)-pairs
- Train linear classifier such that

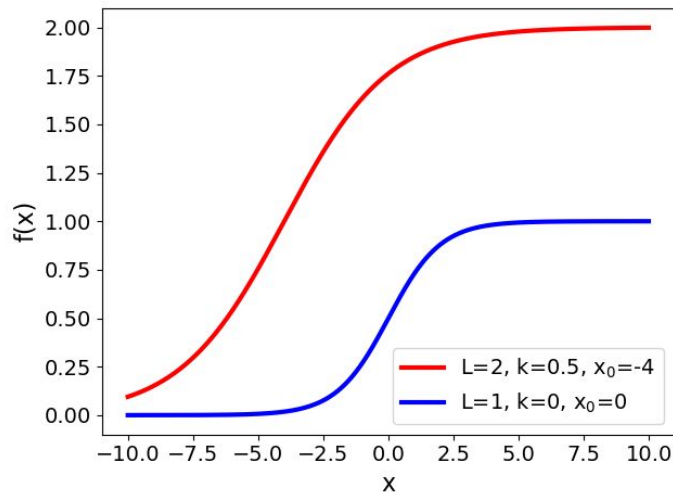
$$0 \leq h_{\theta}(x_i) \leq 1$$



Logistic Regression

- Logistic Regression → Real-valued predictions interpreted as probability
 - Function f is the standard **Logistic Function** (Sigmoid function)

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \xrightarrow{L=1, k=1, x_0=0} f(x) = \frac{1}{1 + e^{-x}}$$



Logistic Regression — Probabilistic Interpretation

- \hat{y} interpreted as a probability

$$\hat{y} = h_{\theta}(x) = f(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \quad \text{with } \hat{y} \in [0, 1]$$

→ $\hat{y} = h_{\theta}(x)$ is the estimated probability that $y_i = 1$ (male) given x and θ

$$\hat{y} = P(y = 1|x, \theta)$$

→ Given only discrete 2 outcomes: $P(y = 1|x, \theta) + P(y = 0|x, \theta) = 1$

$$\hat{y} = 1 - P(y = 0|x, \theta)$$

Logistic Regression — Probabilistic Interpretation

$$\hat{y} = P(y = 1|x, \theta) = 1 - P(y = 0|x, \theta)$$

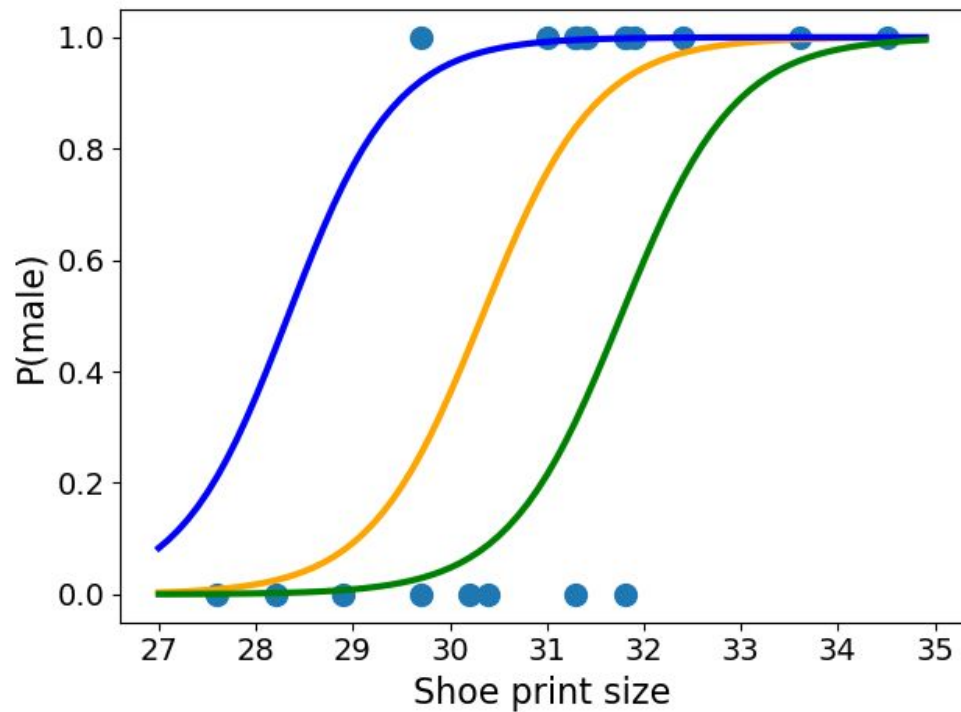
→ $P(y|x)$ is a Bernoulli distribution

$$P(y|x) = \begin{cases} \hat{y} & , y = 1 \\ 1 - \hat{y} & , y = 0 \end{cases}$$

$$P(y|x) = \hat{y}^y (1 - \hat{y})^{1-y}$$

$$\hat{y} = \frac{1}{1 + e^{-\theta^T x}}$$

Logistic Regression



Which $h_{\theta}(x)$ is the best?

How to find it θ ?

Logistic Regression — Loss Function

$$\hat{y} = \frac{1}{1 + e^{-\theta^T x}}$$

- Goal: Maximize probability of true y label given training sample x

- Find θ that **maximizes**

$$P(y|x) = \hat{y}^y (1 - \hat{y})^{1-y}$$

$$\log P(y|x) = \log [\hat{y}^y (1 - \hat{y})^{1-y}]$$

$$= y \log \hat{y} + (1 - y) \log (1 - \hat{y})$$

- Find θ that **minimizes**

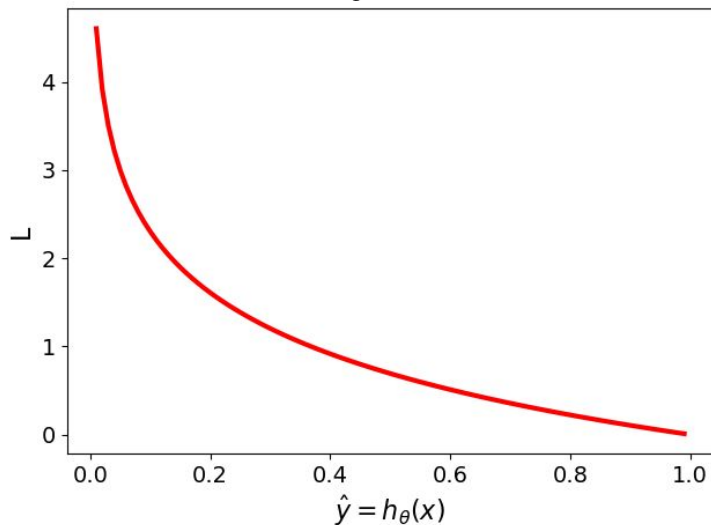
$$L = -P(y|x) = - [y \log \hat{y} + (1 - y) \log (1 - \hat{y})]$$

Cross-Entropy Loss

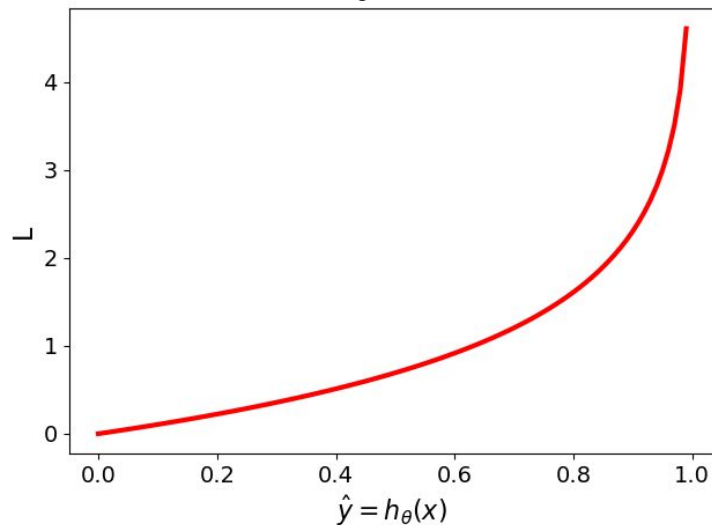
Cross-Entropy Loss — Visualization

$$L = -[y \log \hat{y} + (1 - y) \log (1 - \hat{y})]$$

if $y = 1$



if $y = 0$



Quick Quiz



Overview

- Linear Models
 - Basic setup
- Linear Regression
 - Problem formulation
 - Normal Equation (analytical solution)
 - Gradient Descent (iterative optimization)
 - Polynomial Linear Regression (overfitting, regularization)
 - Interpretation of Coefficients
- Logistic Regression
 - Problem formulation
 - **Gradient Descent**

Logistic Regression — Loss Function

$$\hat{y} = \frac{1}{1 + e^{-\theta^T x}}$$

- Loss for all training samples

$$\begin{aligned} L &= -\frac{1}{n} \sum_{i=1}^n [y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)] \\ &= -\frac{1}{n} \sum_{i=1}^n [y_i \log h_{\theta}(x_i) + (1 - y_i) \log (1 - h_{\theta}(x_i))] \\ &= -\frac{1}{n} \sum_{i=1}^n \left[y_i \log \frac{1}{1 + e^{\theta^T x_i}} + (1 - y_i) \log \left(1 - \frac{1}{1 + e^{\theta^T x_i}} \right) \right] \end{aligned}$$

- Required for minimizing the loss: $\nabla_{\theta} L = \frac{\partial L}{\partial \theta} = ???$

Logistic Regression — Loss Function

- After lots of tedious math...

$$\frac{\partial L}{\partial \theta_j} = \frac{1}{n} \sum_{i=1}^n [h_{\theta}(x_i) - y_i] x_{ij}$$

$$\nabla_{\theta} L = \frac{1}{n} X^T (h_{\theta}(X) - y)$$

- Problem: $\frac{1}{n} X^T (h_{\theta}(X) - y) \stackrel{!}{=} 0$ has no closed-form solution for θ

→ Gradient Descent!

Logistic Regression in Practice (CSI Example)

Vanilla implementation

```
Start training for 1,000,000 iterations...
Loss: 0.6931471805599453      0.0%
Loss: 0.765069661957756      10.0%
Loss: 0.6093119537276577      20.0%
Loss: 0.5066673325457598      30.0%
Loss: 0.4551164039897514      40.0%
Loss: 0.4280635319707213      50.0%
Loss: 0.4154389042684111      60.0%
Loss: 0.4152887492109594      70.0%
Loss: 0.4151849953246913      80.0%
Loss: 0.41511286000667447      90.0%
loss: 0.41506245481850296      100.0%
Training finished in 0:00:09.617970
```

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} -43.45 \\ 1.42 \end{bmatrix}$$

sklearn.linear_model.LogisticRegression (with default L-BFGS-B solver)

```
Start training...
Training finished in 0:00:00.000035 (#iterations: 21)
```

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} -44.91 \\ 1.47 \end{bmatrix}$$

Logistic Regression in Practice (CSI Example)

Vanilla implementation

```
Start training for 78,000 iterations...
Loss: 0.6931471805599453      0.0%
Loss: 0.6565544350291496     10.0%
Loss: 0.6475478825116517     20.0%
Loss: 0.6389819932273285     30.0%
Loss: 0.6308342158505089     40.0%
Loss: 0.6230827400881452     50.0%
Loss: 0.6157065622831859     60.0%
Loss: 0.6086855316895313     70.0%
Loss: 0.6020003798552925     80.0%
Loss: 0.5956327355164103     90.0%
loss: 0.5895658866765062     100.0%
Training finished in 0:00:00.784118
```

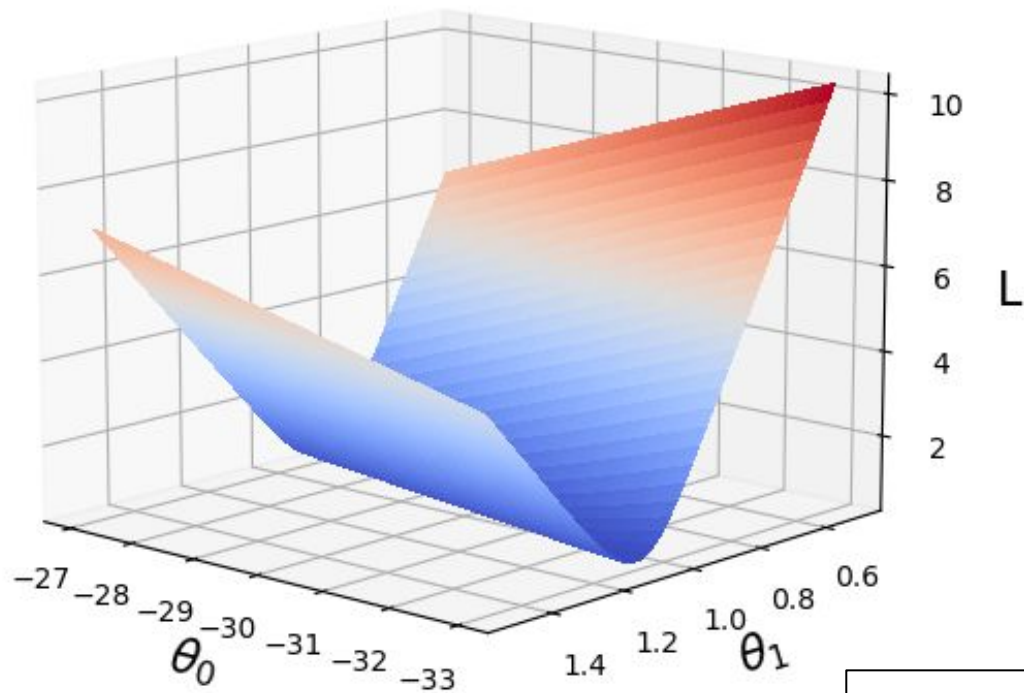
$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} -5.44 \\ 0.19 \end{bmatrix}$$

sklearn.linear_model.LogisticRegression
(with SAG solver)

```
Start training...
Training finished in 0:00:00.007022 (#iterations: 5820)
```

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} -5.45 \\ 0.19 \end{bmatrix}$$

Logistic Regression in Practice (CSI Example)



Region of "near-plateau":

- Gradient $\nabla_{\theta} L$ very small
- Step $\eta \nabla_{\theta} L$ extremely small
- Very slow convergence

Note: The Cross-Entropy loss of Logistic Regression is convex
→ There always exists exactly one minimum (global minimum)!

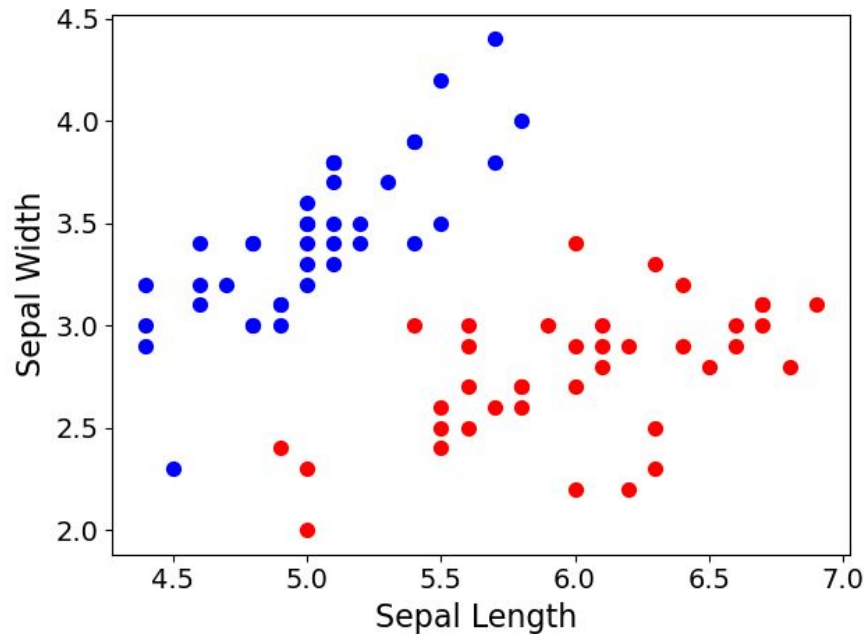
Logistic Regression in Practice

- Logistic Regression with vanilla Gradient Descent (also Linear Regression)
 - Works in principle — the math is sound!
 - Often poor performance compared to more sophisticated implementations
- Many techniques to boost performance
 - Smart(er) initialization of θ
 - Adaptive learning rates
 - Extensions to Gradient Descent
 - Regularization
 - ...and others

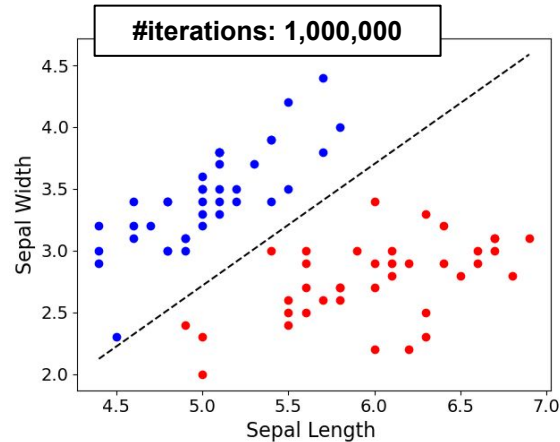
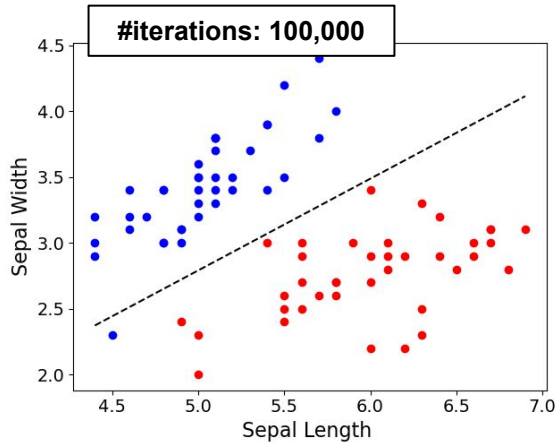
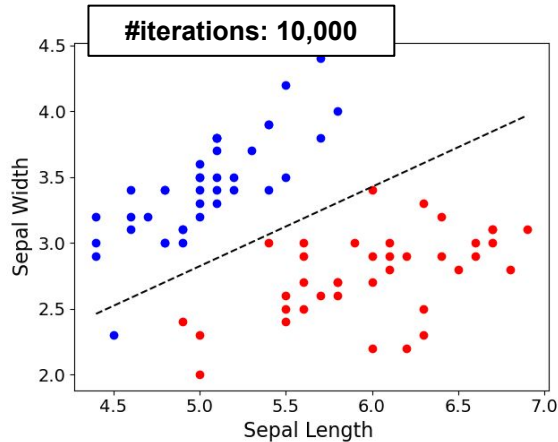
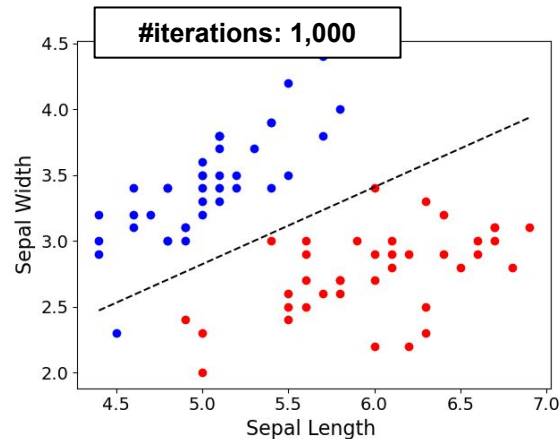
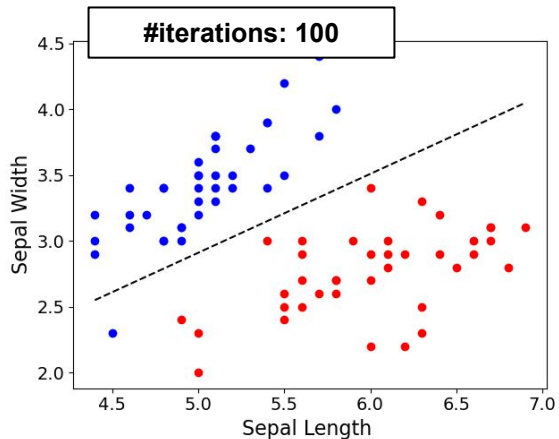
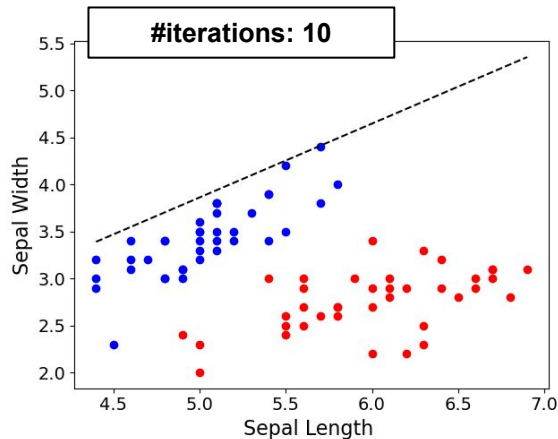
Logistic Regression — 2D Example

- IRIS dataset

- 3 classes of Iris plants
(only 2 considered)
- 50 samples per class
- 4 continuous features
(only sepal length/width considered)



Logistic Regression — 2D Example (Vanilla Gradient Descent)



Polynomial Logistic Regression

- Analogous to Polynomial Linear Regression

- Allows to capture nonlinear relationships between X and y
- Polynomial Logistic Regression model for 1 input feature

$$\hat{y}_i = \frac{1}{1 + e^{-\theta^T x_i}}, \quad \text{with } \theta^T x_i = \theta_0 + \theta_1 x_i + \theta_2 x_i^2 + \dots + \theta_p x_i^p$$

- Identical practical considerations

- Solve using Gradient Descent as usual
(optionally with regularization to avoid overfitting)
- Limited to small number of features
and small polynomial degrees

$$\nabla_{\theta} L = \frac{1}{n} X^T (h_{\theta}(X) - y) + \lambda \frac{2}{n} \theta$$

Overview

- Linear Models
 - Basic setup
- Linear Regression
 - Problem formulation
 - Normal Equation (analytical solution)
 - Gradient Descent (iterative optimization)
 - Polynomial Linear Regression (overfitting, regularization)
 - Interpretation of Coefficients
- Logistic Regression
 - Problem formulation
 - Gradient Descent

Summary

- Linear Models

- Assume linear relationship between input features and output
(i.e., $\text{output} = \text{sum of weighted feature values}$)
- However, regression line / decision boundary not always a line/plane/hyperplane
(data transformation to add polynomial terms based on input features)
- Generally good interpretability

- Linear Regression & Logistic Regression

- Very fundamental and popular regression and classification methods
- Gradient Descent to solve both tasks + Normal Equation to solve Linear Regression
- Effects of data normalization mainly(!) on explainability / interpretability of results
- Straightforward extension of Logistic Regression beyond 2 classes (not covered here)

Solutions to Quick Quizzes

