Universität Konstanz

Lecture 2 Linear regression & Logistic regression

Dr. Hanhe Lin
Dept. of Computer and Information Science
University of Konstanz

Overview

- Linear regression with one variable

- Optimization

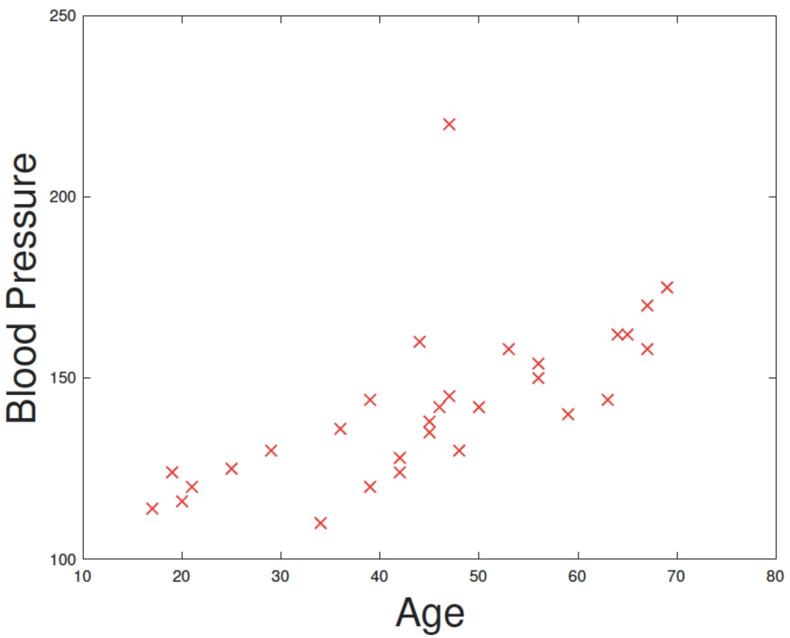
- Linear regression with multiple variables

- Logistic regression

Linear regression with one variable

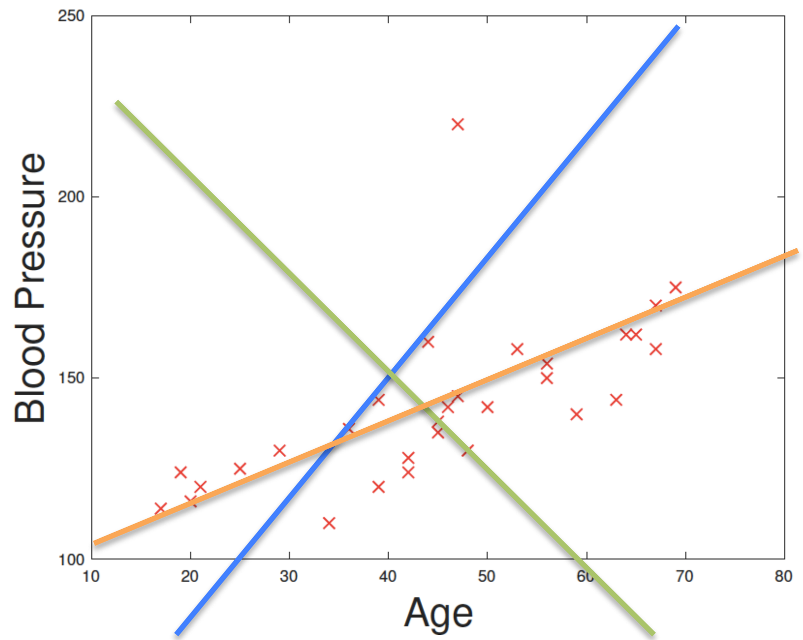
A simple example

- Linear regression is:
  - Supervised learning
  - Regression problem
- Example: predict blood pressure according to age



A simple example

- Linear regression is:
  - Supervised learning
  - Regression problem



Which line is better?

Notations

Age (x)	Blood pressure (y)
39	144
47	220
45	138
47	145
…	…

- m: number of training examples
- x: "input"
- y: "output"
- (x, y): one training example
- $(x^{(i)}, y^{(i)})$: i-th training example

Question

Age (x)	Blood pressure (y)
39	144
47	220
45	138
47	145
…	…

- Consider the training set above, what is $x^{(2)}$ and $y^{(3)}$?

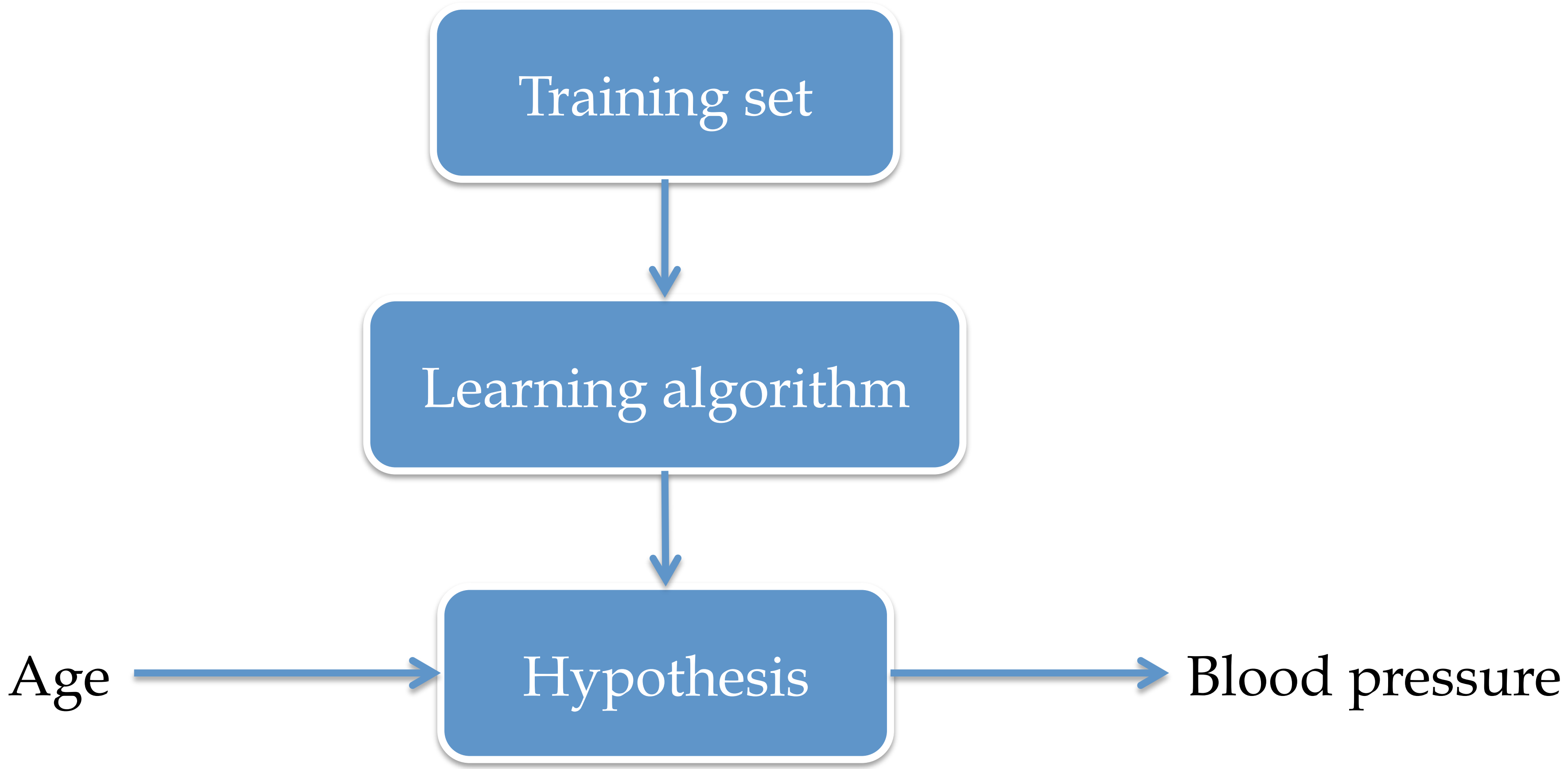
Question

Age (x)	Blood pressure (y)
39	144
47	220
45	138
47	145
…	…

- Consider the training set above, what is $x^{(2)}$ and $y^{(3)}$?

47 and 138

Model representation



Hypothesis

- How do we represent the hypothesis of linear regression model?
  - $h_{w,b}(x) = wx + b$
  - Linear regression with one variable
  - Now the question becomes how to choose w, b?

Loss function

- Loss/cost function
- Idea: choose w, b so that $h_{w,b}(x)$ is close to y for train examples (x, y)
- Loss function:

$$L(w,b) = \frac{1}{m}\sum_{i=1}^{m}\left(h_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

Loss function

- Loss/cost function
- Idea: choose w, b so that $h_{w,b}(x)$ is close to y for train examples (x, y)
- Loss function:

$$L(w,b) = \frac{1}{m}\sum_{i=1}^{m}\left(h_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

$$L(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

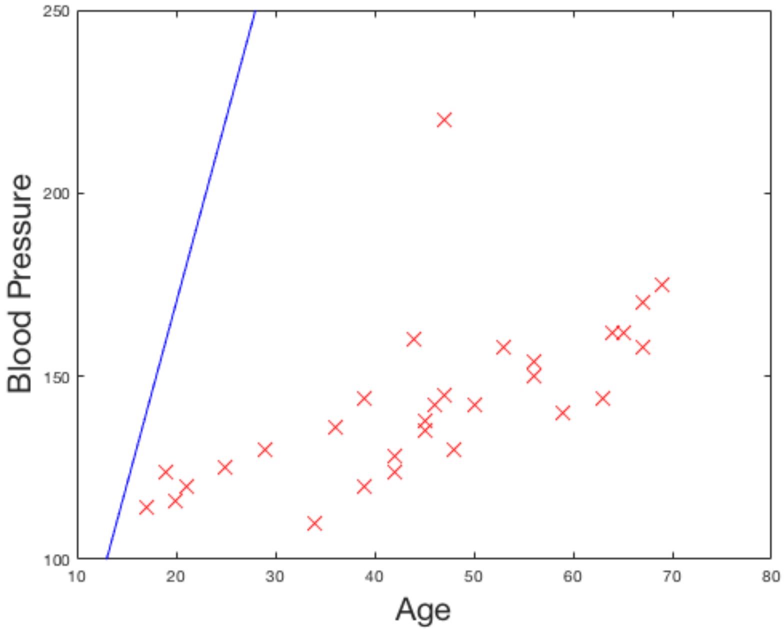
Linear regression with one variable (Overview)

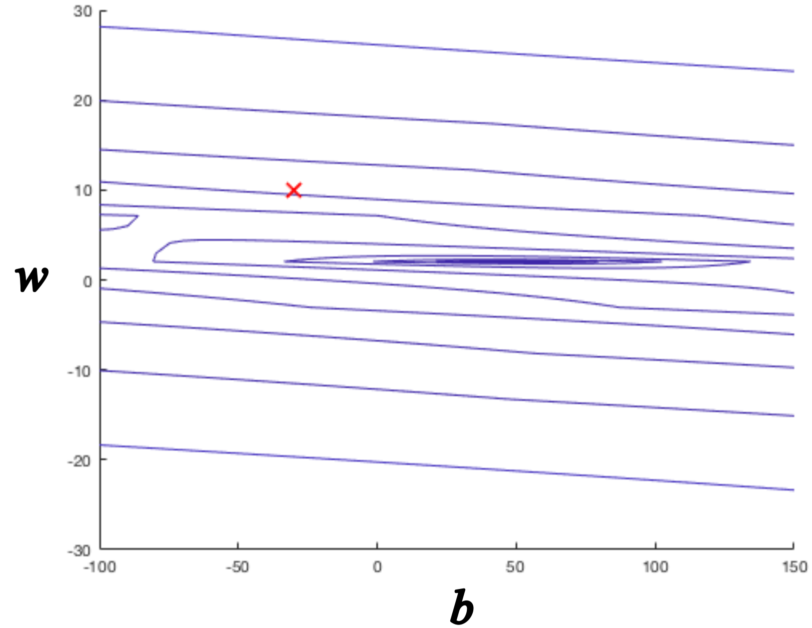
- Hypothesis: $h_{w,b}(x) = wx + b$
- Parameters: w, b
- Loss function:

$$L(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

- Goal: $\min_{w,b} L(w,b)$

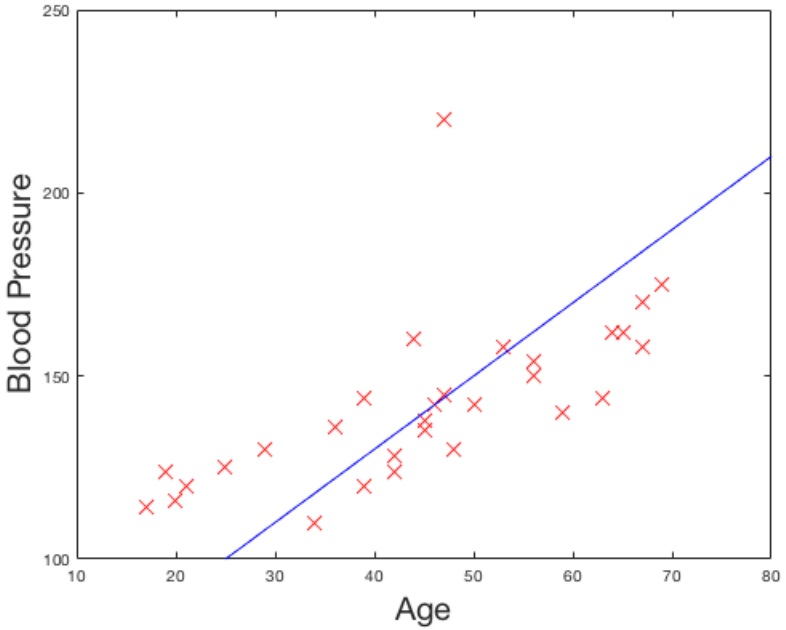
Visualization

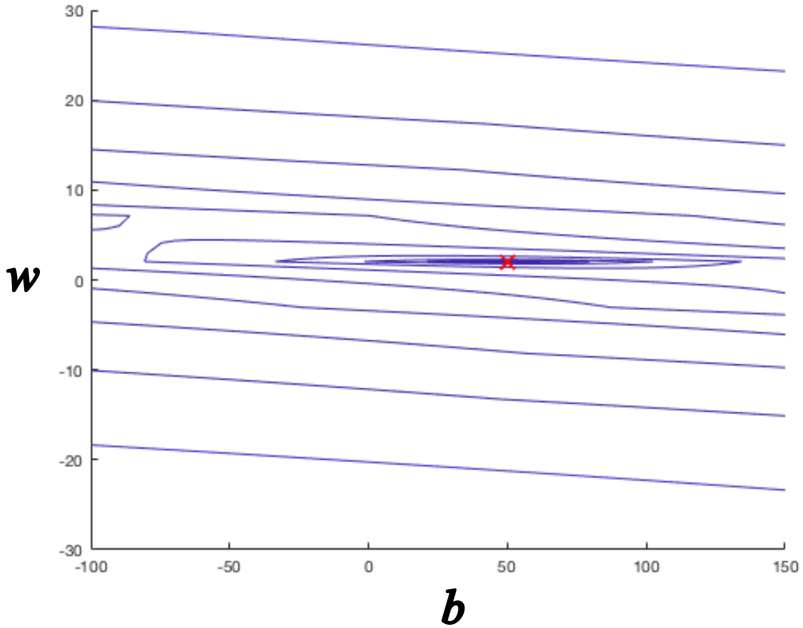




$$w = 10, b = -30$$

Visualization

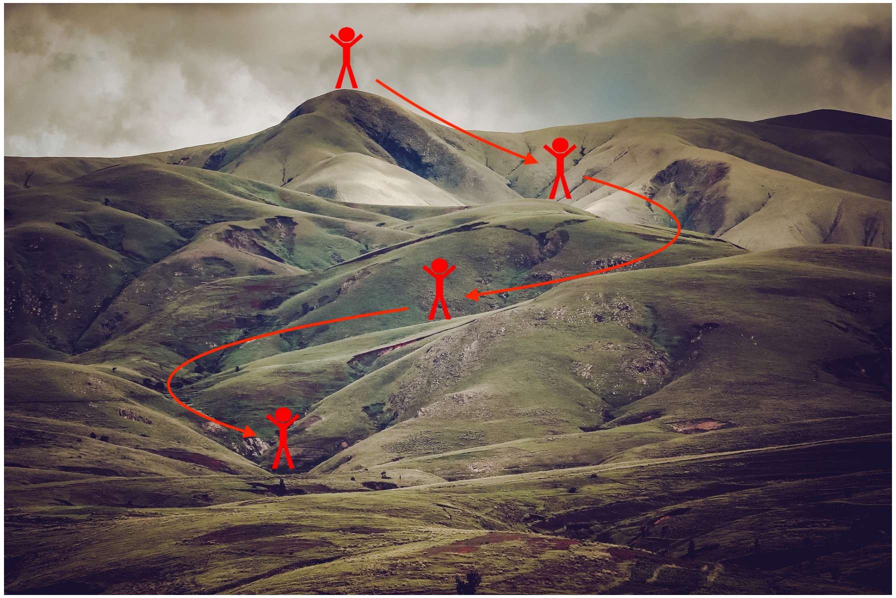




$$w = 2, b = 50$$

Optimization

- First-order methods
  - Gradient descent
- Second-order methods
  - Newton's method
  - Conjugate gradients
  - BFGS



Minimizing the loss is like finding the lowest point in a hilly landscape

Gradient descent

- Given a function $f(x)$, our objective is

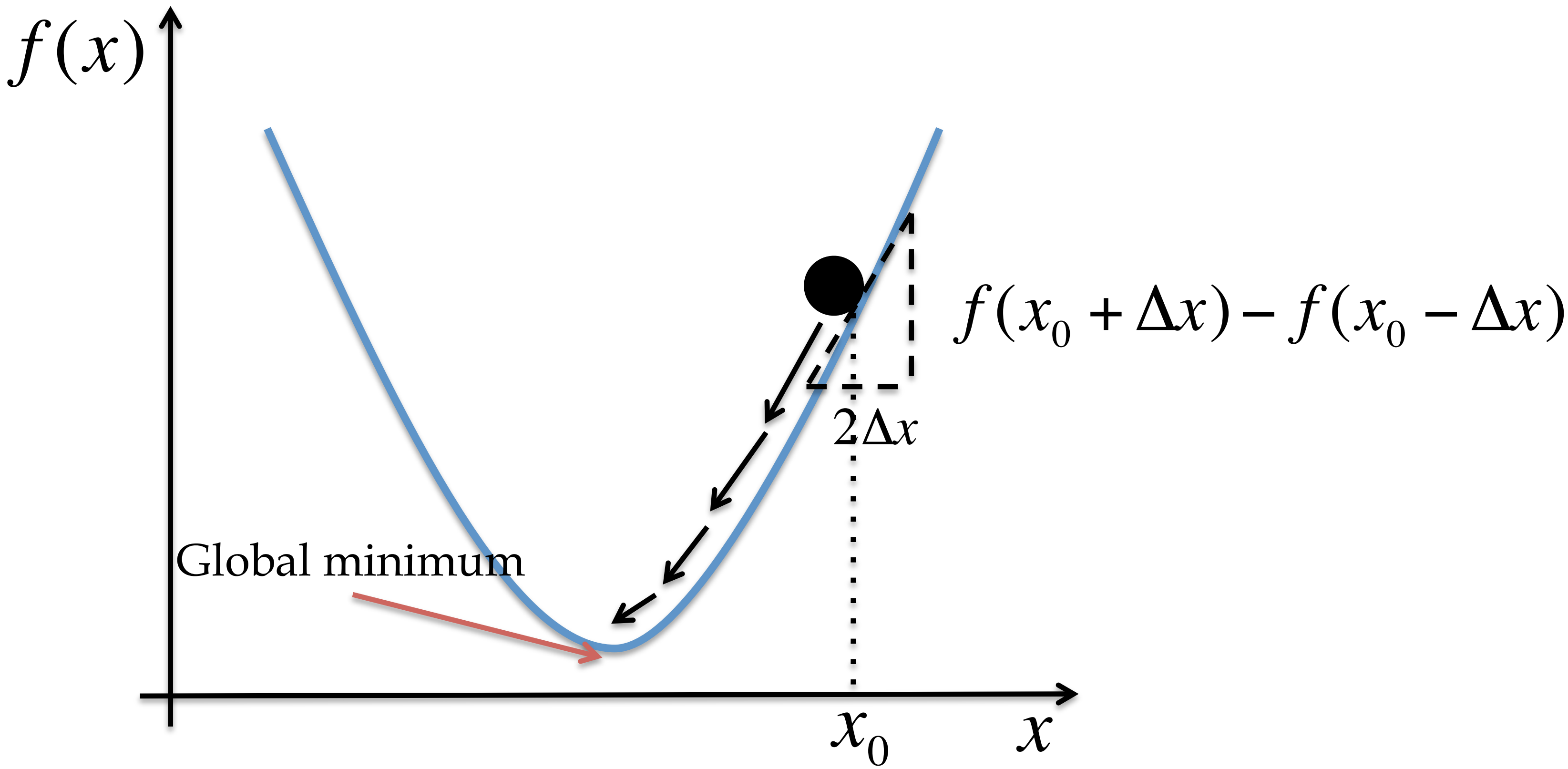
$$\min_x f(x)$$

- Repeat until converge:

$$x := x - \alpha \frac{\partial}{\partial x} f$$

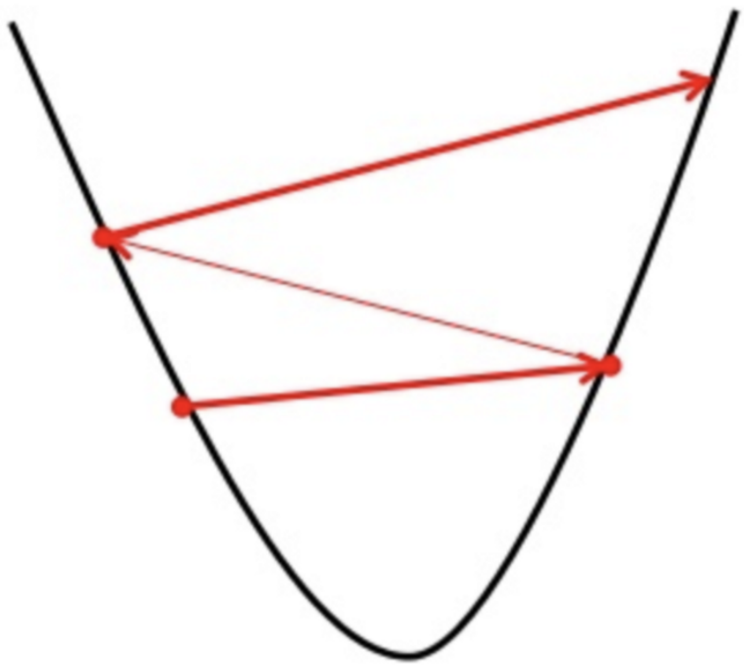
Learning rate

Gradient descent (cont.)

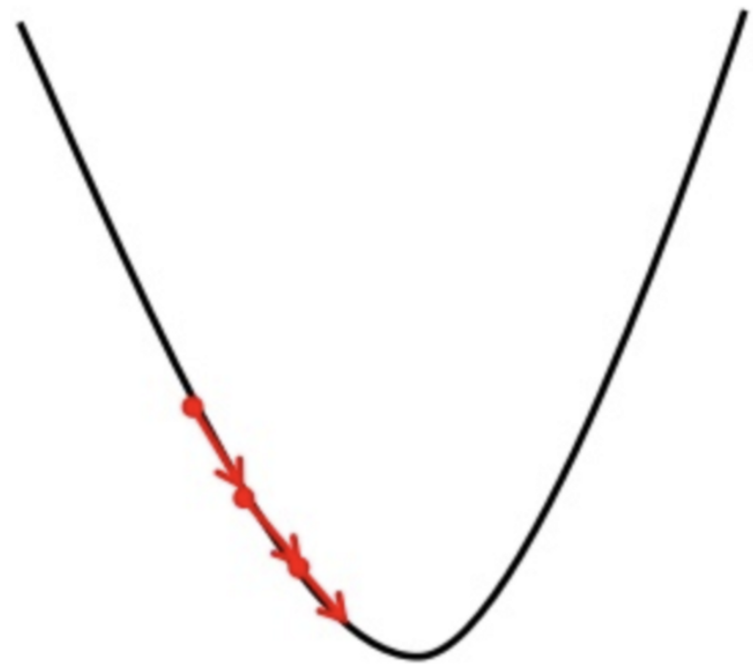


$$\frac{\partial}{\partial x} f = \lim_{\Delta x \to 0} \frac{f(x_0+\Delta x)-f(x_0-\Delta x)}{2\Delta x}$$

Learning rate

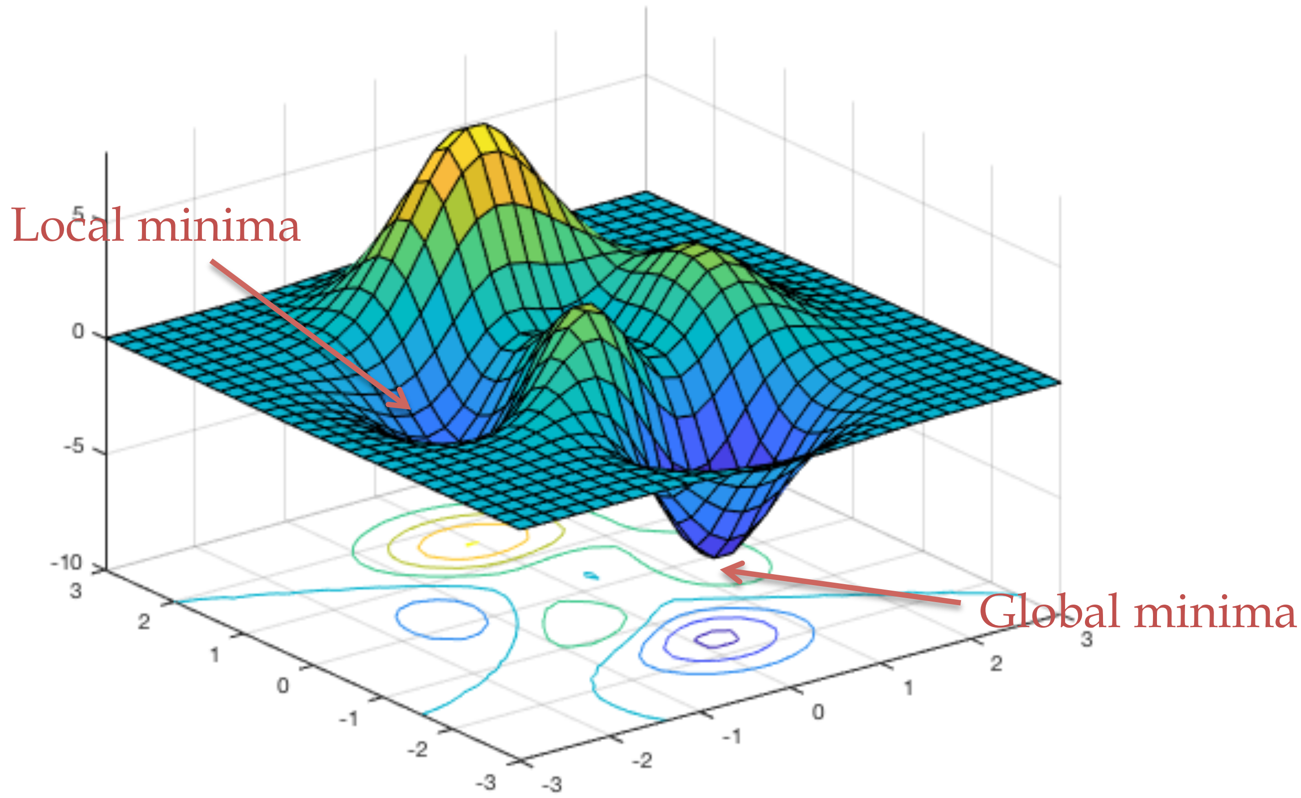


Too large



Too small

Local minima



Non-convex loss function

Gradient descent - concern

- If learning rate is too small, gradient descent can be slow, more iterations are needed
- If learning rate is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.
- May converge to a local minimum if the loss function is non-convex
- When we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.
- "Batch": all the training examples are used in each step of gradient descent

Gradient descent for linear regression (one variable)

- Repeat until converge:

$$w := w - \alpha \frac{\partial}{\partial w} L(w,b)$$

$$b := b - \alpha \frac{\partial}{\partial b} L(w,b)$$

Question: which implementation is correct?

- Repeat until converge:

$$temp0 = w - \alpha \frac{\partial}{\partial w} L(w,b)$$

$$temp1 = b - \alpha \frac{\partial}{\partial b} L(w,b)$$

$$w = temp0$$

$$b = temp1$$

- Repeat until converge:

$$temp0 = w - \alpha \frac{\partial}{\partial w} L(w,b)$$

$$w = temp0$$

$$temp1 = b - \alpha \frac{\partial}{\partial b} L(w,b)$$

$$b = temp1$$

Question: which implementation is correct?

- Repeat until converge:

$$temp0 = w - \alpha \frac{\partial}{\partial w} L(w,b)$$

$$temp1 = b - \alpha \frac{\partial}{\partial b} L(w,b)$$

$$w = temp0$$

$$b = temp1$$

✔

- Repeat until converge:

$$temp0 = w - \alpha \frac{\partial}{\partial w} L(w,b)$$

$$w = temp0$$

$$temp1 = b - \alpha \frac{\partial}{\partial b} L(w,b)$$

$$b = temp1$$

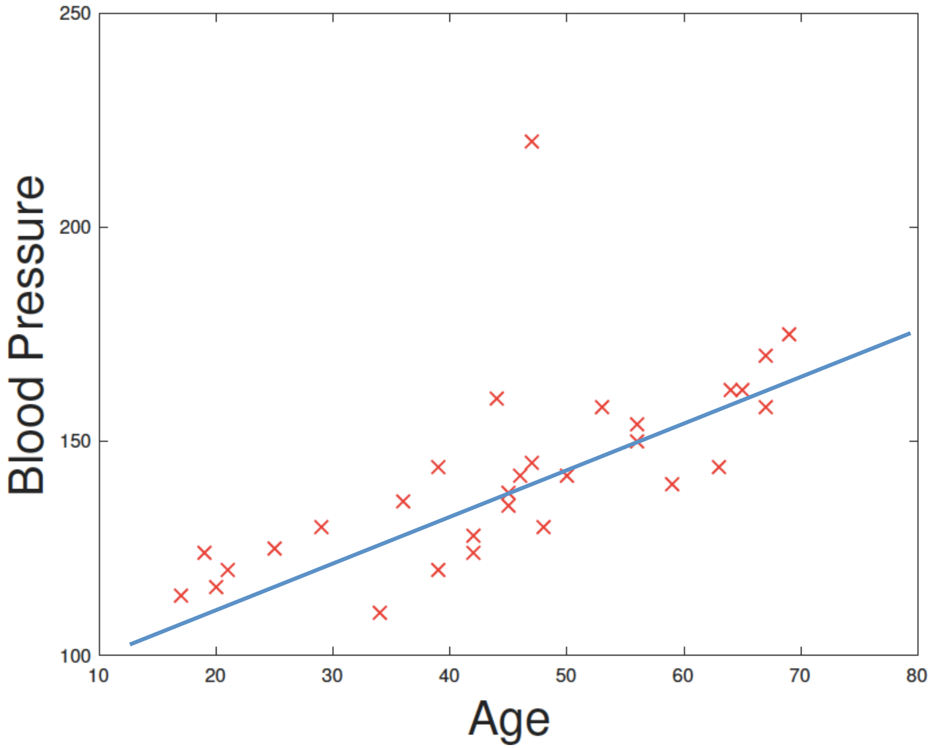
Gradient descent for linear regression (one variable)

- Repeat until converge:

$$w := w - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$b := b - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_{w,b}(x^{(i)}) - y^{(i)})$$

Linear regression: optimization



Linear regression with multiple variables

Linear regression with multiple variables

Area of site (1000 square feet)	Size of living place (1000 square feet)	Number of rooms	Ages in years	Selling price
3.472	0.998	7	42	25.9
3.531	1.500	7	62	29.5
2.275	1.175	6	40	27.9
4.050	1.232	6	54	25.9
…	…	…	…	…

- n: number of features
- $x^{(i)}$: input features of i-th training example
- $x_j^{(i)}$: values of feature j in i-th training example

Q: what is $x^{(3)}$ and $x_2^{(4)}$ in the table?

Linear regression with multiple variables

Area of site (1000 square feet)	Size of living place (1000 square feet)	Number of rooms	Ages in years	Selling price
3.472	0.998	7	42	25.9
3.531	1.500	7	62	29.5
2.275	1.175	6	40	27.9
4.050	1.232	6	54	25.9
…	…	…	…	…

- n: number of features
- $x^{(i)}$: input features of i-th training example
- $x_j^{(i)}$: values of feature j in i-th training example

Q: what is $x^{(3)}$ and $x_2^{(4)}$ in the table?

Hypothesis

- Hypothesis for multiple features:

$$h_{w,b}(x) = w^T x + b$$

where

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \qquad w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} \quad \xrightarrow{\text{Simplify}} \quad h(x) = wx + b$$

Gradient descent for linear regression (multiple variables)

- Repeat until converge:

$$w_j := w_j - \alpha \frac{1}{m} \sum_{i=0}^{m} (h(x^{(i)}) - y^{(i)}) x_j^{(i)} \quad j = 1, \ldots, n$$

$$b := b - \alpha \frac{1}{m} \sum_{i=0}^{m} (h(x^{(i)}) - y^{(i)})$$

Linear regression with multiple variables

- Hypothesis: $h(x) = wx + b$
- Parameters: w, b
- Loss function:

$$L(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(h(x^{(i)}) - y^{(i)}\right)^2$$

- Gradient descent:
  - Repeat until converge:

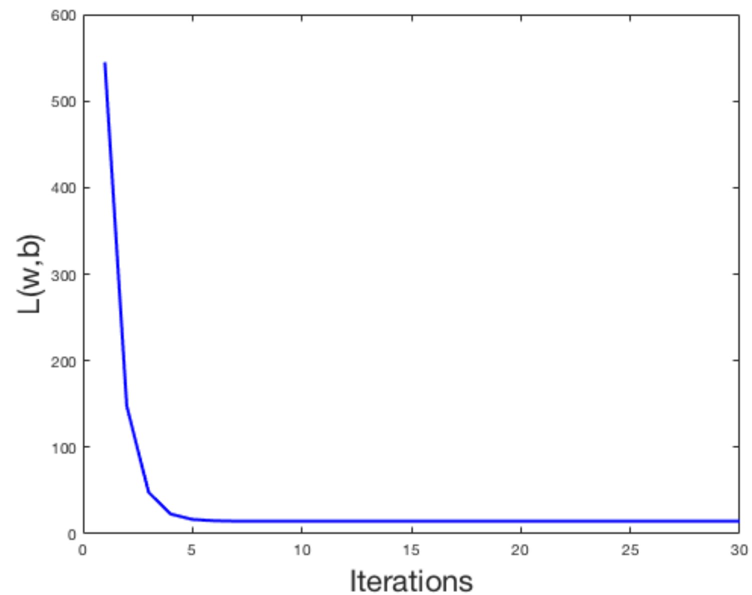
$$w_j := w_j - \alpha\frac{1}{m}\sum_{i=0}^{m}(h(x^{(i)}) - y^{(i)})x_j^{(i)} \quad j = 1,\ldots,n \qquad b := b - \alpha\frac{1}{m}\sum_{i=0}^{m}(h(x^{(i)}) - y^{(i)})$$

Suggestions on gradient descent

- How to make sure gradient descent is working correctly?
- How to choose learning rate?

Make sure GD is working correctly

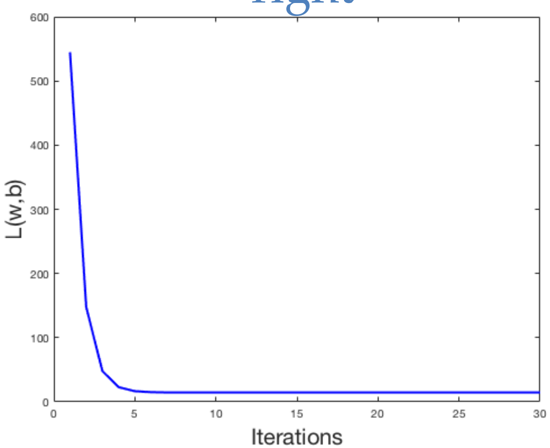
- Ideal loss output: decrease sharply, then slightly decrease
- Declare convergence if loss decrease between two iterations is less than a threshold



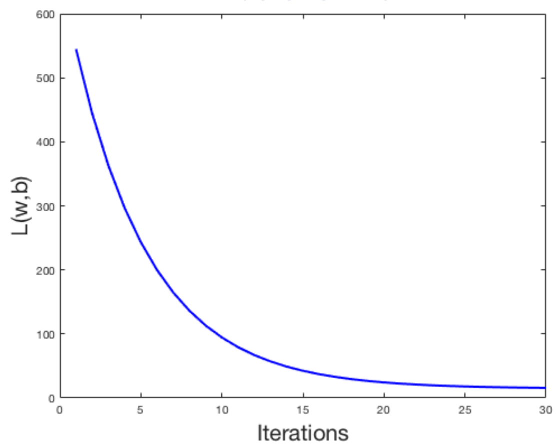
Choosing learning rate

- Learning rate is a hyper-parameter:
  - Too small, more iteration; too large, may not converge
  - To choose learning (grid search), try …, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, …

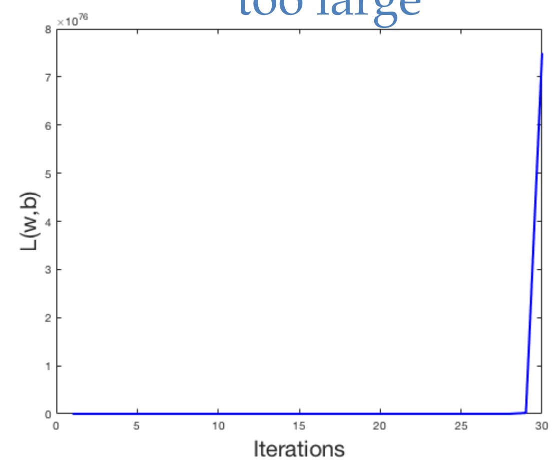
right



too small

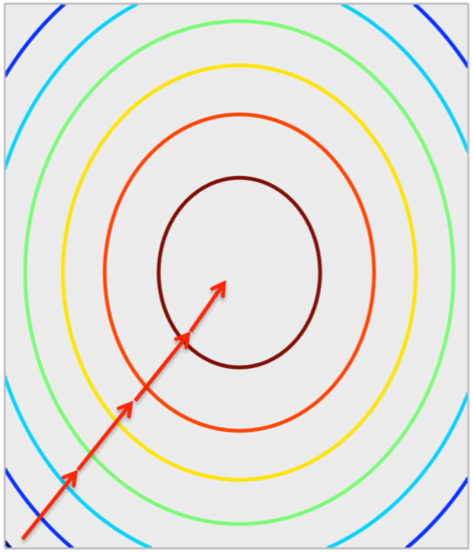


too large

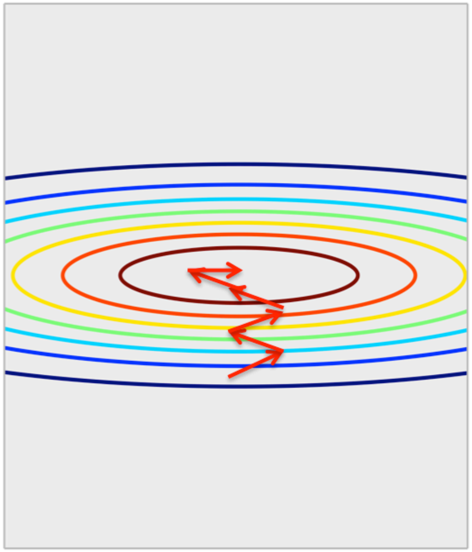


Feature normalization

- Each feature has a different scale, which may generate an oval shape
- Result: more iterations to converge
- Four forms:
  - Mean subtraction
  - Normalization
  - PCA
  - Whitening



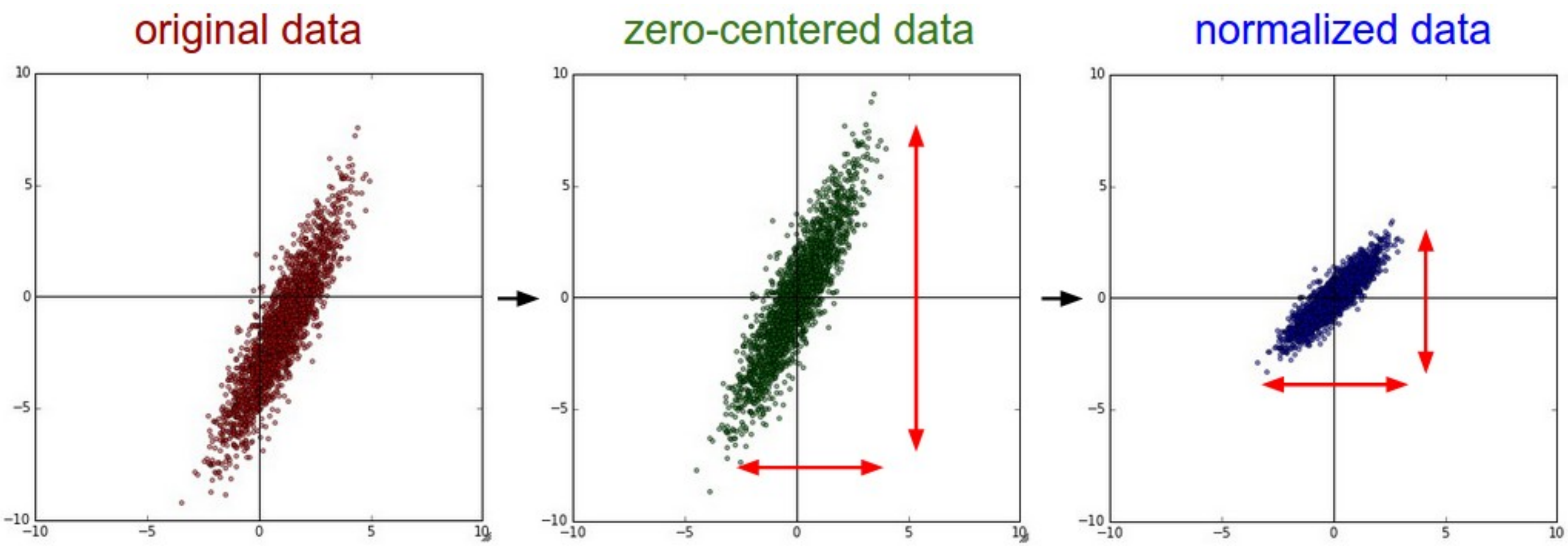
With normalization



Without normalization

Mean subtraction and normalization

- Mean subtraction:
  - most common form of preprocessing
  - subtract the mean across every individual feature in the data
- Normalization:
  - normalize the data dimensions so that they are of approximately the same scale
  - divide each dimension by its standard deviation after mean subtraction



Mean subtraction and normalization

- Mean subtraction

$$x_j^{(i)} = x_j^{(i)} - \mu_j \quad j = 1, 2, \ldots, n$$

- Normalization

$$x_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j} \quad j = 1, 2, \ldots, n$$

Note: If you apply feature normalization in your training set, you should do the same process given a new data for prediction

Normal equation for linear regression

	Area of site (1000 square feet)	Size of living place (1000 square feet)	Number of rooms	Ages in years	Selling price
1	3.472	0.998	7	42	25.9
1	3.531	1.500	7	62	29.5
1	2.275	1.175	6	40	27.9
1	4.050	1.232	6	54	25.9

$$X = \begin{bmatrix} 1 & 3.472 & 0.998 & 7 & 42 \\ 1 & 3.531 & 1.500 & 7 & 62 \\ 1 & 2.275 & 1.175 & 6 & 40 \\ 1 & 4.050 & 1.232 & 6 & 54 \end{bmatrix} \qquad y = \begin{bmatrix} 25.9 \\ 29.5 \\ 27.9 \\ 25.9 \end{bmatrix}$$

Normal equation for liner regression

- Instead of gradient descent, we can obtain the best solution using the following equations:

$$\begin{bmatrix} b \\ w \end{bmatrix} = (X^T X)^{-1} X^T y$$

- One line Matlab code!

Gradient descent vs. normal equation

Gradient descent

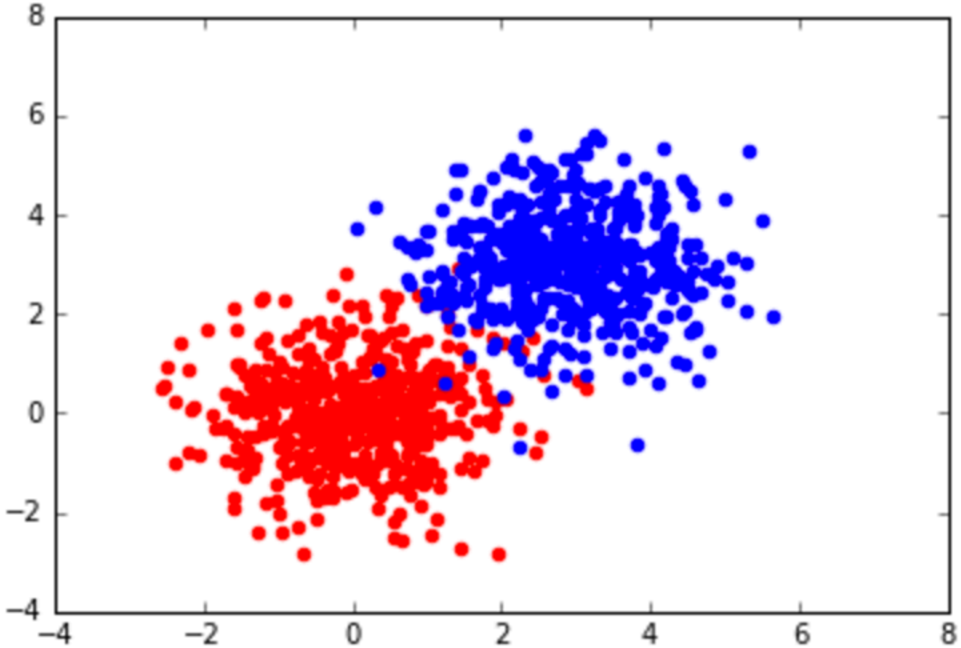
- Need to choose learning rate
- Need many iterations
- Works well even when number of features is very large

Normal equation

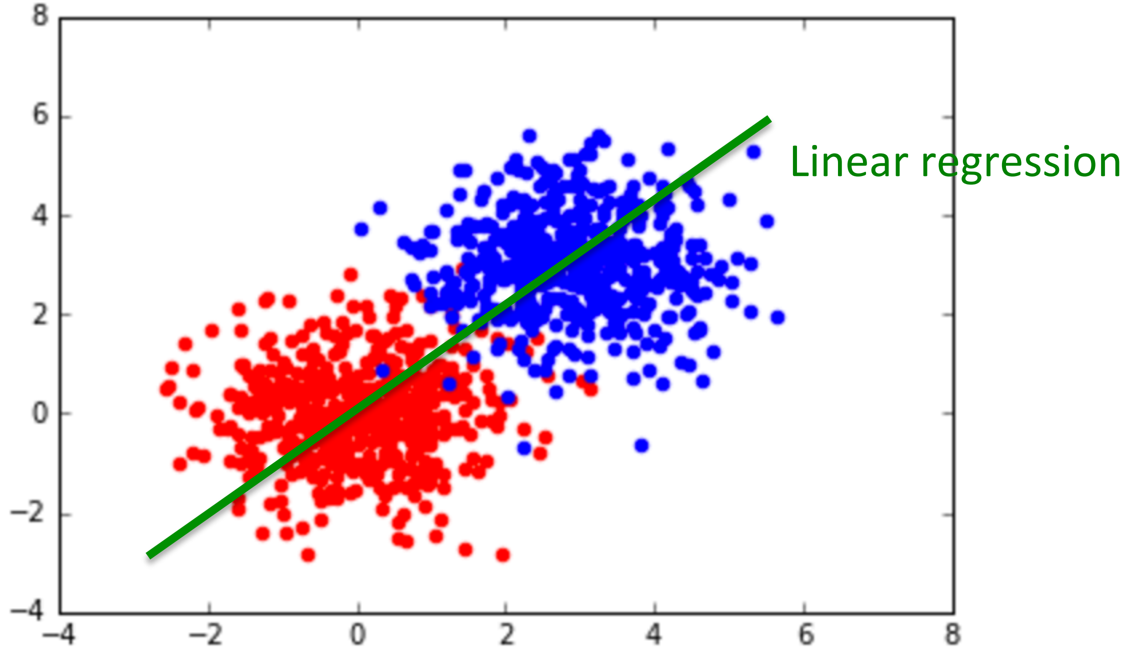
- No learning rate
- No iterations
- Need to compute $(X^T X)^{-1}$, slow when number of feature is very large, non-invertible

Classification: logistic regression

Binary classification

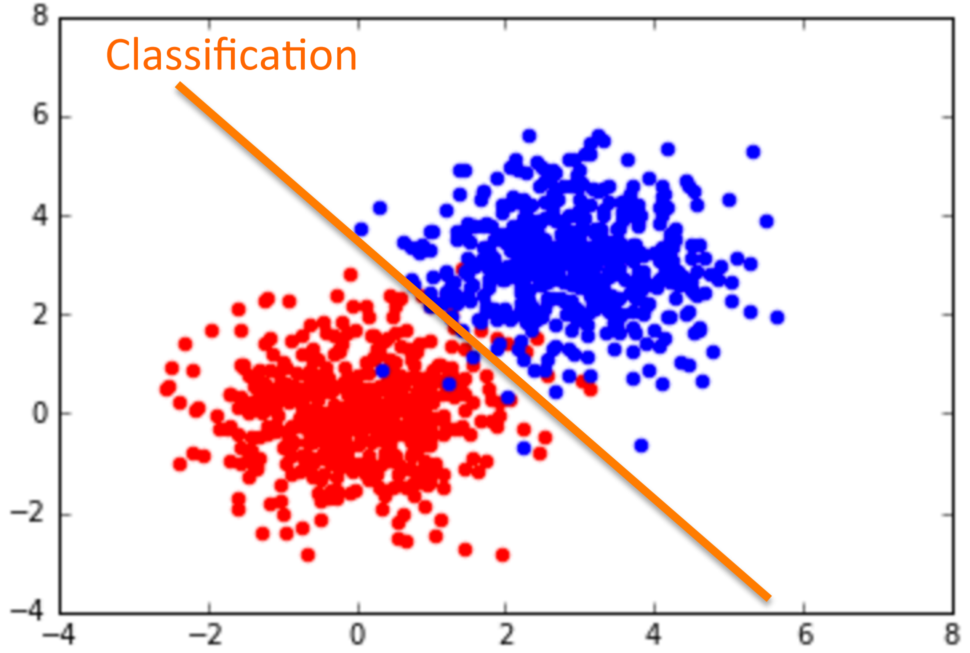


Binary classification



MSE is not suitable for binary classification!

Binary classification



Binary classification

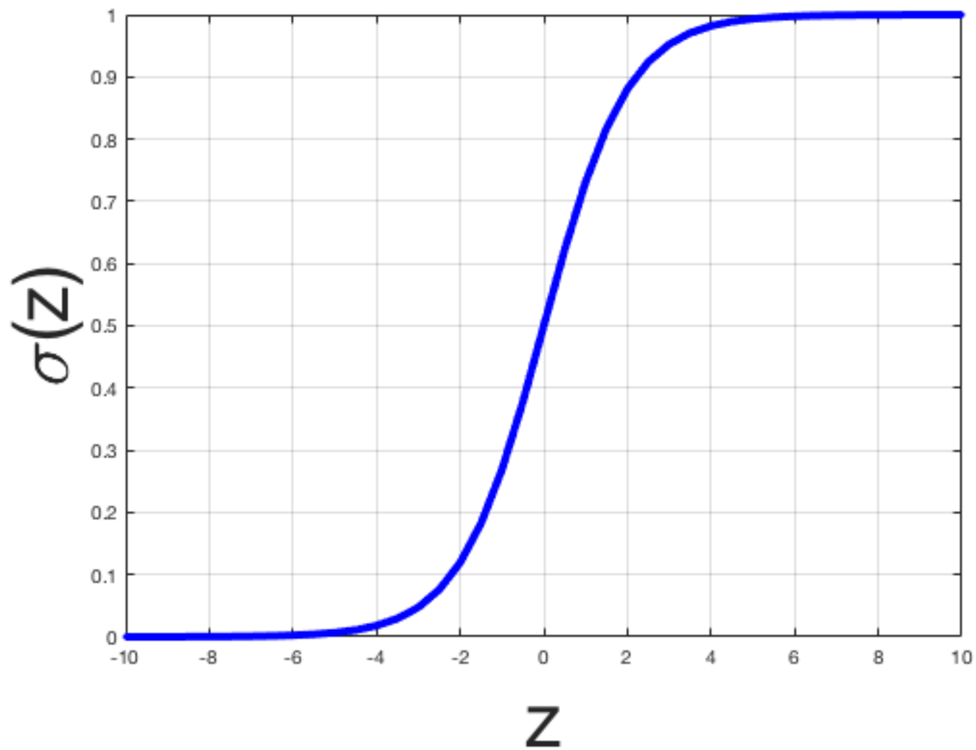
- Examples:
  - Email: spam / not spam
  - Tumor: malignant / benign
  - Image: face / not face
- $y \in \{0,1\}$, where 1 is positive class, and 0 is negative class, e.g., face (1) vs. not face (0)
- Intuitively, negative class conveys absence of something

Hypothesis

- $0 \le h(x) \le 1$
- If $h(x) \ge 0.5$, predict "y=1"
- If $h(x) < 0.5$, predict "y=0"

Logistic/sigmoid function

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Hypothesis of logistic regression

$$h(x) = \sigma(wx + b)$$

$$\Rightarrow \quad h_{w,b}(x) = \frac{1}{1 + e^{-(wx+b)}}$$

Interpretation of hypothesis output

- Estimated probability that $y=1$, given x, "parameterized" by w,b:

$$h_{w,b}(x)=P(y=1\mid x;w,b)$$

- Example: given an image for face detection, if we have $h_{w,b}(x)=0.7$, which means: 70% chance of the image is a face image
- As we only have two classes, we have:

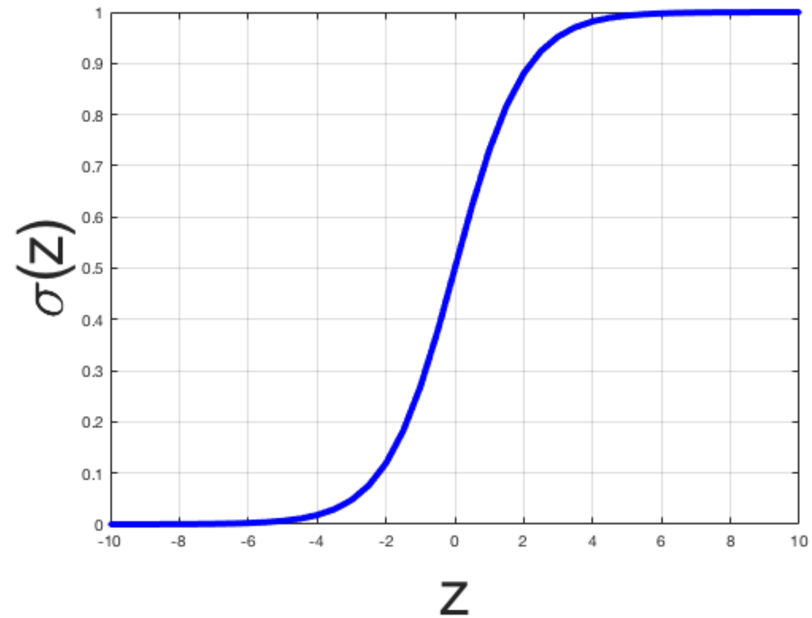
$$P(y=0\mid x;w,b)+P(y=1\mid x;w,b)=1$$

$$\Rightarrow P(y=0\mid x;w,b)=1-P(y=1\mid x;w,b)$$

Hypothesis of logistic regression (revisit)

$$h(x) = \sigma(wx + b)$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



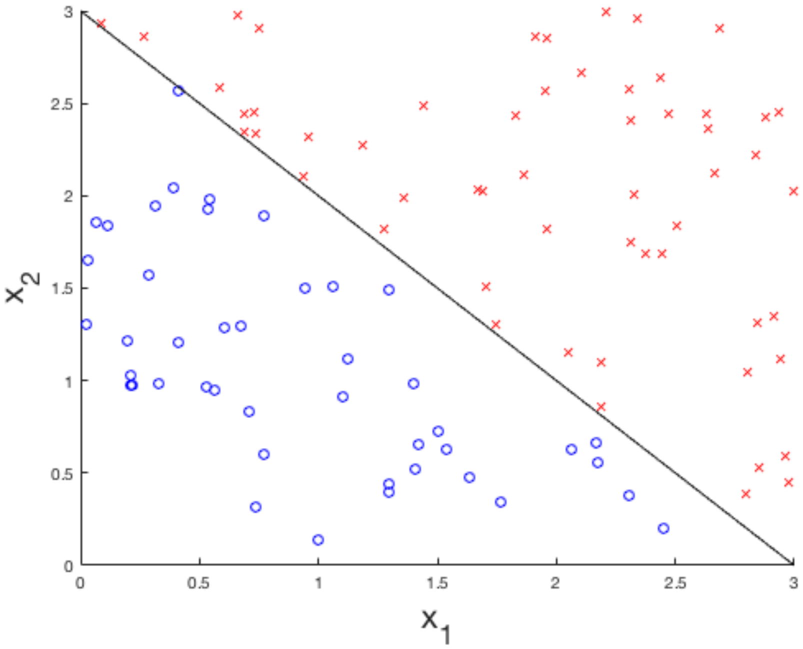
- Predict "y=1" when $h(x) \geq 0.5 \Rightarrow wx + b \geq 0$
- Predict "y=0" when $h(x) < 0.5 \Rightarrow wx + b < 0$

Decision boundary

- $h(x) = \sigma(w_1 x_1 + w_2 x_2 + b)$
- Decision boundary:

$$x_1 + x_2 - 3 = 0$$

- Note different ML model generates different decision boundary



Rewrite loss function

- A new representation of loss function:

$$L(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Loss}(h(x^{(i)}), y^{(i)})$$

- In linear regression, it is given by:

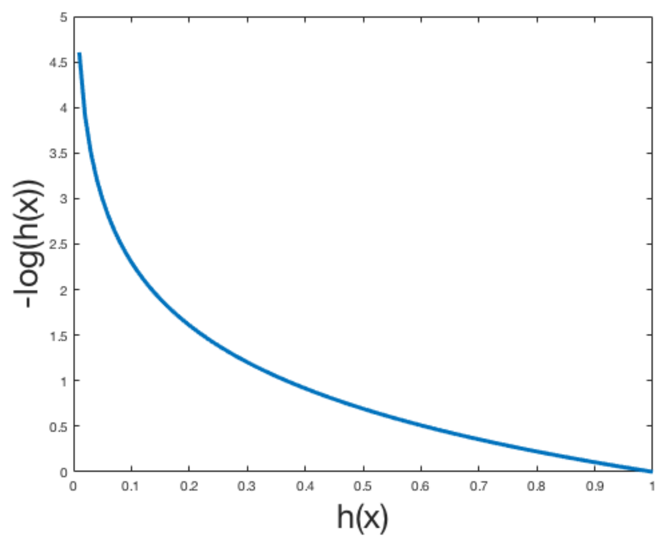
$$\mathrm{Loss}(h(x), y) = \frac{1}{2}(h(x) - y)^2$$

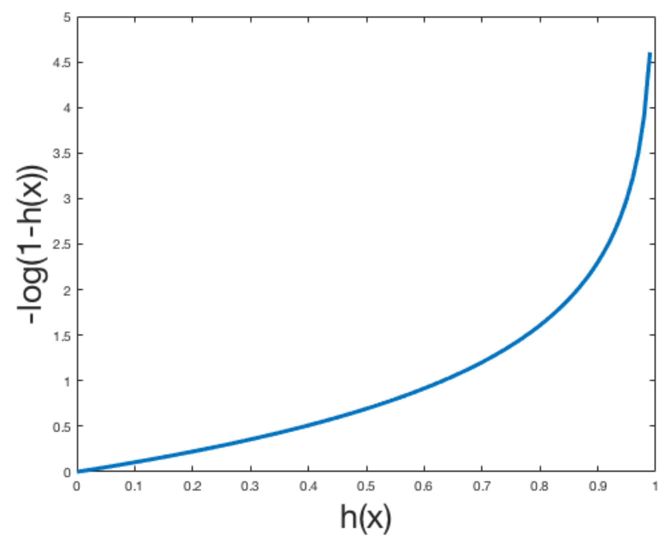
Loss function of logistic regression

- Loss function:

$$\text{Loss}(h(x),y)=\begin{cases} -\log(h(x)) \; \textit{if} \;\; y=1 \\ -\log(1-h(x)) \; \textit{if} \;\; y=0 \end{cases}$$

- Intuition: if $h(x)=0$, but $y=1$, the learning algorithm will be penalized by a vary large loss





Loss function of logistic regression

- We can compact two cases into one equation:

$$\text{Loss}(h(x), y) = -y \log(h(x)) - (1 - y)\log(1 - h(x))$$

Logistic regression (overview)

- Hypothesis: $h(x) = \dfrac{1}{1 + e^{-(wx+b)}}$
- Parameter: w, b
- Loss function:

$$L(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log(h(x^{(i)})) + (1 - y^{(i)})\log(1 - h(x^{(i)}))\right)$$

- Goal: $\min\limits_{w,b} L(w,b)$

Gradient descent for logistic regression

- Given the loss function:

$$L(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left(y\log(h(x)) + (1-y)\log(1-h(x))\right)$$

our objective is to $\min_{w,b} L(w,b)$

- Repeat until converge:

$$w_j := w_j - \alpha\frac{1}{m}\sum_{i=0}^{m}(h(x^{(i)}) - y^{(i)})x_j^{(i)} \quad j = 1,\ldots,n$$

$$b := b - \alpha\frac{1}{m}\sum_{i=0}^{m}(h(x^{(i)}) - y^{(i)})$$

Identical to linear regression except the hypothesis is different!