STOR566: Introduction to Deep Learning

Lecture 3: Linear regression and classification

Yao Li
UNC Chapel Hill

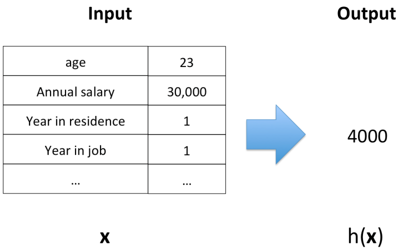
Aug 27, 2024

Materials are from *Learning from data (Caltech)* and *Deep Learning (UCLA)*

Linear Regression

Regression

- Regression: predicting a real number



Input

age	23
Annual salary	30,000
Year in residence	1
Year in job	1
...	...

Output

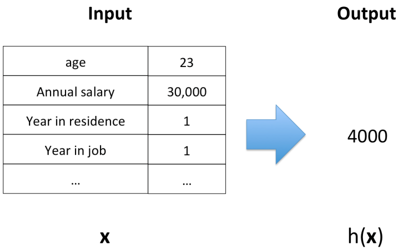
4000

$\mathbf{x}$

$h(\mathbf{x})$

Regression

- Regression: predicting a real number



Input

Output

age	23
Annual salary	30,000
Year in residence	1
Year in job	1
...	...

4000

$\mathbf{x}$

h($\mathbf{x}$)

$$\text{Linear Regression: } h(\boldsymbol{x}) = \sum_{i=0}^{d} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$$

Problem definition

- Training data:

$$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_N, y_N)$$

$\boldsymbol{x}_n \in \mathbb{R}^d$: feature vector for a sample

$y_n \in \mathbb{R}$: observed output (real number)

Problem definition

- Training data:

$$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_N, y_N)$$

  $\boldsymbol{x}_n \in \mathbb{R}^d$: feature vector for a sample

  $y_n \in \mathbb{R}$: observed output (real number)

- Linear regression: find a function $h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$ to approximate y

Problem definition

- Training data:

$$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_N, y_N)$$

  $\boldsymbol{x}_n \in \mathbb{R}^d$: feature vector for a sample

  $y_n \in \mathbb{R}$: observed output (real number)

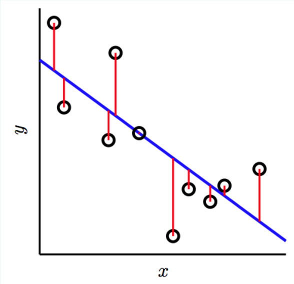
- Linear regression: find a function $h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$ to approximate y
- Measure the error by $(h(\boldsymbol{x}) - y)^2$ (square error)

$$\text{Training error}: L_{\text{train}}(h) = \frac{1}{N} \sum_{n=1}^{N} (h(\boldsymbol{x}_n) - y_n)^2$$

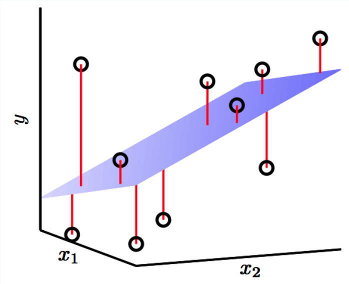
- Possible issues in the pipeline.

Illustration

$\mathbf{x} = (x) \in \mathbb{R}$



$\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$



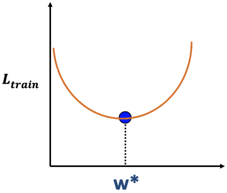
Linear regression: find linear function with small residual

Minimize L_{train}

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}) = \|X\boldsymbol{w} - \mathbf{y}\|^2$$

- $X \in \mathbb{R}^{N\times d}$, $y \in \mathbb{R}^N$
- The objective function is continuous, differentiable, convex
- The optimal $\boldsymbol{w}^*$ will satisfy:

$$\nabla f(\boldsymbol{w}^*) = \begin{bmatrix} \frac{\partial f}{\partial w_0}(\boldsymbol{w}^*) \\ \vdots \\ \frac{\partial f}{\partial w_d}(\boldsymbol{w}^*) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$



Minimizing f

$$f(\boldsymbol{w}) = \|X\boldsymbol{w} - \mathbf{y}\|^2 = \boldsymbol{w}^T X^T X \boldsymbol{w} - 2\boldsymbol{w}^T X^T \mathbf{y} + \mathbf{y}^T \mathbf{y}$$

$$\nabla f(\boldsymbol{w}) = ?$$

Minimizing f

$$\nabla f(\boldsymbol{w}^*) = 0 \Rightarrow \underbrace{X^T X \boldsymbol{w}^* = X^T \mathbf{y}}$$

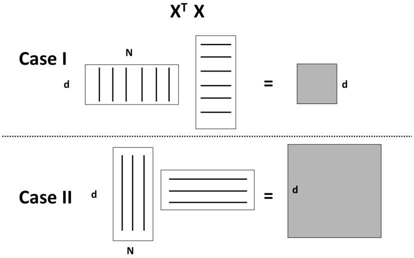
Minimizing f

$$\nabla f(\boldsymbol{w}^*) = 0 \Rightarrow \underbrace{X^T X \boldsymbol{w}^* = X^T \mathbf{y}}$$

$$\Rightarrow \boldsymbol{w}^* = (X^T X)^{-1} X^T \mathbf{y} \quad ??$$

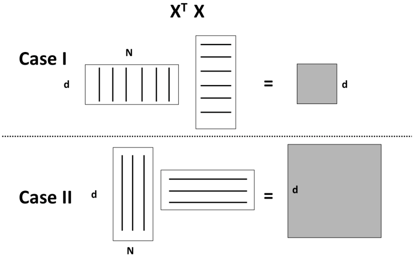
More on Linear Regression Solutions

- Case I: $X^T X$ is invertible $\Rightarrow$ Unique solution
  - Often when $N > d$
  - $\boldsymbol{w}^* = (X^T X)^{-1} X^T \mathbf{y}$
- Case II: $X^T X$ is non-invertible $\Rightarrow$ Many solutions
  - Often when $d > N$



More on Linear Regression Solutions

- Case I: X^TX is invertible ⇒ Unique solution
  - Often when $N > d$
  - $\boldsymbol{w}^* = (X^TX)^{-1}X^T\mathbf{y}$
- Case II: X^TX is non-invertible ⇒ Many solutions
  - Often when $d > N$



pseudo-inverse of X^TX

Logistic Regression

Binary Classification

- Input: training data $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ and corresponding outputs $y_1, y_2, \ldots, y_n \in \{+1, -1\}$
- Training: compute a function f such that $\text{sign}(f(\boldsymbol{x}_i)) \approx y_i$ for all i
- Prediction: given a testing sample $\tilde{\boldsymbol{x}}$, predict the output as $\text{sign}(f(\tilde{\boldsymbol{x}}))$

Logistic Regression

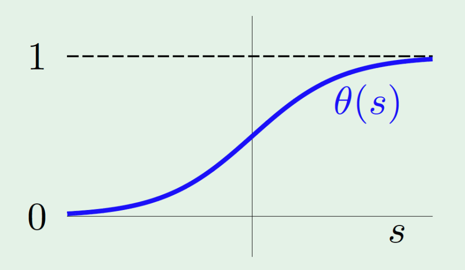
- Assume linear scoring function: $s = f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$

Logistic Regression

- Assume linear scoring function: $s = f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$
- Logistic hypothesis:

$$P(y = 1 \mid \boldsymbol{x}) = \theta(\boldsymbol{w}^T \boldsymbol{x}),$$

where $\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$



Logistic Regression

- Assume linear scoring function: $s = f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$
- Logistic hypothesis:

$$P(y = 1 \mid \boldsymbol{x}) = \theta(\boldsymbol{w}^T\boldsymbol{x}),$$

  where $\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$.
- Therefore, $P(y = 1 \mid \boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$.

Logistic Regression

- Assume linear scoring function: $s = f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
- Logistic hypothesis:

$$P(y = 1 \mid \mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x}),$$

  where $\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$.
- Therefore, $P(y = 1 \mid \mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$.
- How about $P(y = -1 \mid \mathbf{x})$?

Logistic Regression

- Assume linear scoring function: $s = f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$
- Logistic hypothesis:

$$P(y = 1 \mid \boldsymbol{x}) = \theta(\boldsymbol{w}^T\boldsymbol{x}),$$

  where $\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$.
- Therefore, $P(y = 1 \mid \boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$.
- How about $P(y = -1 \mid \boldsymbol{x})$?
  $P(y = -1 \mid \boldsymbol{x}) = 1 - \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}} = \frac{1}{1+e^{\boldsymbol{w}^T\boldsymbol{x}}} = \theta(-\boldsymbol{w}^T\boldsymbol{x})$

Logistic Regression

- Assume linear scoring function: $s = f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$
- Logistic hypothesis:

$$P(y = 1 \mid \boldsymbol{x}) = \theta(\boldsymbol{w}^T\boldsymbol{x}),$$

  where $\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$.
- Therefore, $P(y = 1 \mid \boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$.

- How about $P(y = -1 \mid \boldsymbol{x})$?
  $P(y = -1 \mid \boldsymbol{x}) = 1 - \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}} = \frac{1}{1+e^{\boldsymbol{w}^T\boldsymbol{x}}} = \theta(-\boldsymbol{w}^T\boldsymbol{x})$
- Therefore, $P(y \mid \boldsymbol{x}) = \theta(y\boldsymbol{w}^T\boldsymbol{x})$

Maximizing the likelihood

- Likelihood of $\mathcal{D} = (\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)$:

$$\Pi_{n=1}^{N} P(y_n \mid \boldsymbol{x}_n) = \Pi_{n=1}^{N} \theta(y_n \boldsymbol{w}^T \boldsymbol{x}_n)$$

Maximizing the likelihood

- Likelihood of $\mathcal{D} = (\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)$:

$$\Pi_{n=1}^{N} P(y_n \mid \boldsymbol{x}_n) = \Pi_{n=1}^{N} \theta(y_n \boldsymbol{w}^T \boldsymbol{x}_n)$$

- Find $\boldsymbol{w}$ to maximize the likelihood!

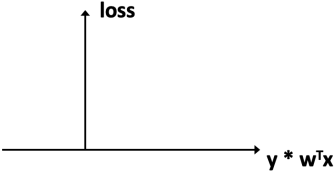
$$\begin{aligned} & \max_{\boldsymbol{w}} \Pi_{n=1}^{N} \theta(y_n \boldsymbol{w}^T \boldsymbol{x}_n) \\ \Leftrightarrow & \max_{\boldsymbol{w}} \log(\Pi_{n=1}^{N} \theta(y_n \boldsymbol{w}^T \boldsymbol{x}_n)) \\ \Leftrightarrow & \min_{\boldsymbol{w}} -\sum_{n=1}^{N} \log(\theta(y_n \boldsymbol{w}^T \boldsymbol{x}_n)) \\ \Leftrightarrow & \min_{\boldsymbol{w}} \sum_{n=1}^{N} \log(1 + e^{-y_n \boldsymbol{w}^T \boldsymbol{x}_n}) \end{aligned}$$

Empirical Risk Minimization (linear)

- Linear classification/regression:

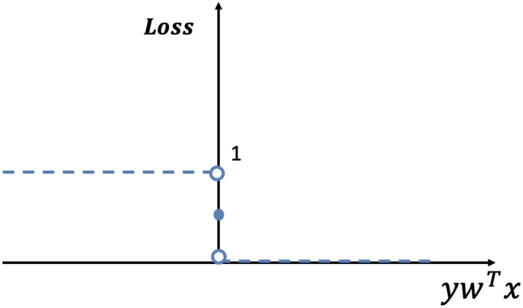
$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^{N} \text{loss}(\underbrace{\boldsymbol{w}^T \boldsymbol{x}_n}_{\hat{y}_n\text{:the predicted score}}, y_n)$$

  - Linear regression: $\text{loss}(h(\boldsymbol{x}_n), y_n) = (\boldsymbol{w}^T \boldsymbol{x}_n - y_n)^2$
  - Logistic regression: $\text{loss}(h(\boldsymbol{x}_n), y_n) = \log(1 + e^{-y_n \boldsymbol{w}^T \boldsymbol{x}_n})$
  - Hinge loss (SVM): $\text{loss}(h(\boldsymbol{x}_n), y_n) = \max(0, 1 - y_n \boldsymbol{w}^T \boldsymbol{x}_n)$

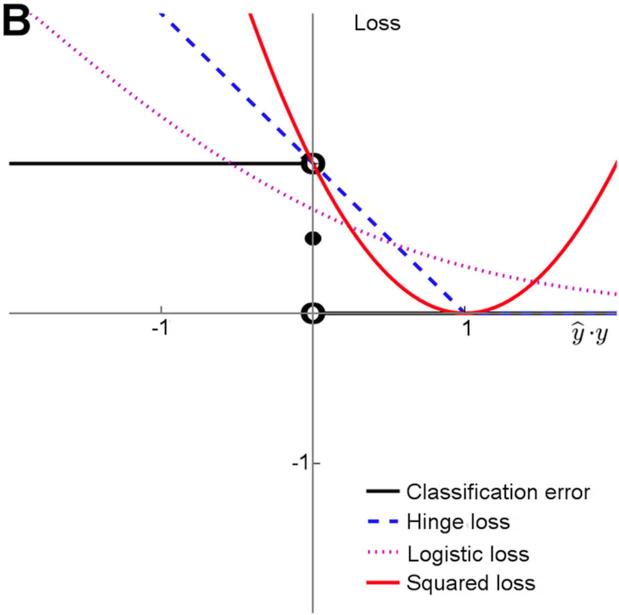


Binary Classification Loss

- Linear regression: $\text{loss}(h(\boldsymbol{x}_n), y_n) = (\boldsymbol{w}^T \boldsymbol{x}_n - y_n)^2$
- Logistic regression: $\text{loss}(h(\boldsymbol{x}_n), y_n) = \log(1 + e^{-y_n \boldsymbol{w}^T \boldsymbol{x}_n})$
- Hinge loss (SVM): $\text{loss}(h(\boldsymbol{x}_n), y_n) = \max(0, 1 - y_n \boldsymbol{w}^T \boldsymbol{x}_n)$



Binary Classification Loss



Empirical Risk Minimization (general)

- Assume $f_W(\boldsymbol{x})$ is the decision function to be learned (W is the parameters of the function)
- General empirical risk minimization:

$$\min_{W} \frac{1}{N} \sum_{n=1}^{N} \text{loss}(f_W(\boldsymbol{x}_n), y_n)$$

- Example: Neural network ($f_W(\cdot)$ is the network)

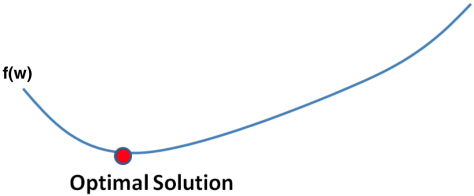
Gradient descent and SGD

Optimization

- Goal: find the minimizer of a function

$$\min_{\boldsymbol{w}} f(\boldsymbol{w})$$

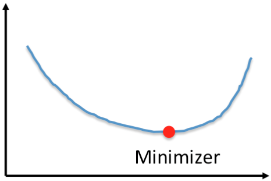
For now we assume f is twice differentiable



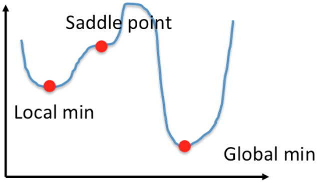
Convex vs Nonconvex

- Convex function:
  - $\nabla f(\boldsymbol{x}) = 0 \Leftrightarrow$ Global minimum
  - A function is convex if $\nabla^2 f(\boldsymbol{x})$ is positive definite
  - Example: linear regression, logistic regression, $\cdots$
- Non-convex function:
  - $\nabla f(\boldsymbol{x}) = 0 \Leftrightarrow$ Global min, local min, or saddle point
    most algorithms only converge to gradient$= 0$
  - Example: neural network, $\cdots$

Convex



Non-Convex



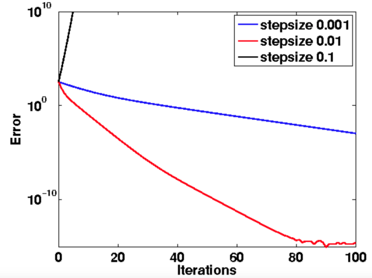
Gradient Descent

- Gradient descent: repeatedly do

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \alpha \nabla f(\boldsymbol{w}_t)$$

  $\alpha > 0$ is the step size

- Step size too large $\Rightarrow$ diverge; too small $\Rightarrow$ slow convergence



Why gradient descent?

- At each iteration, form an approximation function of $f(\cdot)$:

$$f(\boldsymbol{w}_t + \boldsymbol{d}) \approx g(\boldsymbol{d}) := f(\boldsymbol{w}_t) + \nabla f(\boldsymbol{w}_t)^T \boldsymbol{d} + \frac{1}{2\alpha}\|\boldsymbol{d}\|^2$$

- Update solution by $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \boldsymbol{d}^*$
- $\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} g(\boldsymbol{d})$
  $\nabla g(\boldsymbol{d}^*) = 0 \Rightarrow \nabla f(\boldsymbol{w}_t) + \frac{1}{\alpha}\boldsymbol{d}^* = 0 \Rightarrow \boldsymbol{d}^* = -\alpha \nabla f(\boldsymbol{w}_t)$
- $\boldsymbol{d}^*$ will decrease $f(\cdot)$ if α (step size) is sufficiently small

Illustration of gradient descent

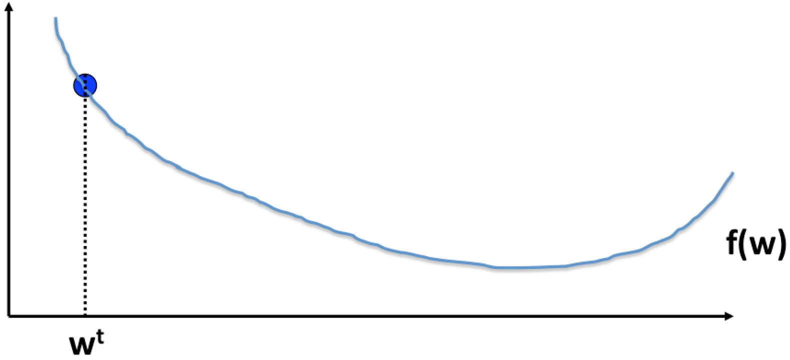
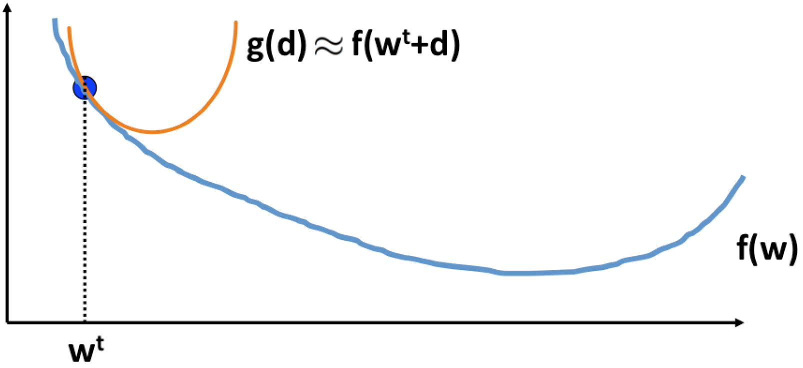


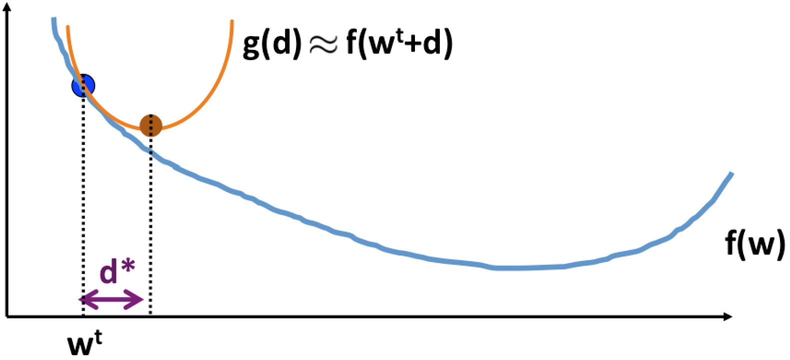
Illustration of gradient descent



Form a quadratic approximation

$$f(\boldsymbol{w}_t + \boldsymbol{d}) \approx g(\boldsymbol{d}) = f(\boldsymbol{w}_t) + \nabla f(\boldsymbol{w}_t)^T \boldsymbol{d} + \frac{1}{2\alpha}\|\boldsymbol{d}\|^2$$

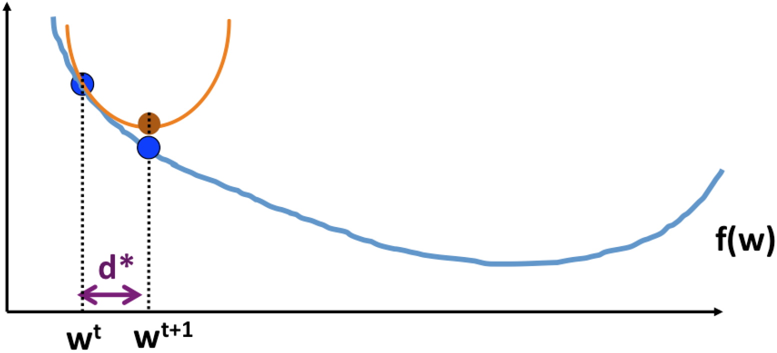
Illustration of gradient descent



Minimize $g(\boldsymbol{d})$:

$$\nabla g(\boldsymbol{d}^*) = 0 \Rightarrow \nabla f(\boldsymbol{w}_t) + \frac{1}{\alpha}\boldsymbol{d}^* = 0 \Rightarrow \boldsymbol{d}^* = -\alpha \nabla f(\boldsymbol{w}_t)$$

Illustration of gradient descent



Update $\boldsymbol{w}$:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{d}^* = \boldsymbol{w}_t - \alpha \nabla f(\boldsymbol{w}_t)$$

Illustration of gradient descent

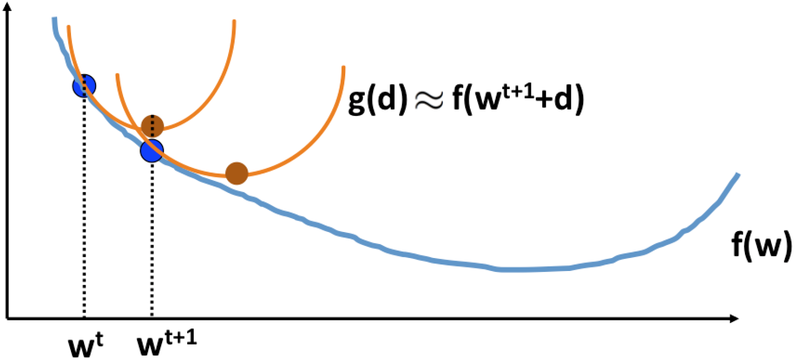
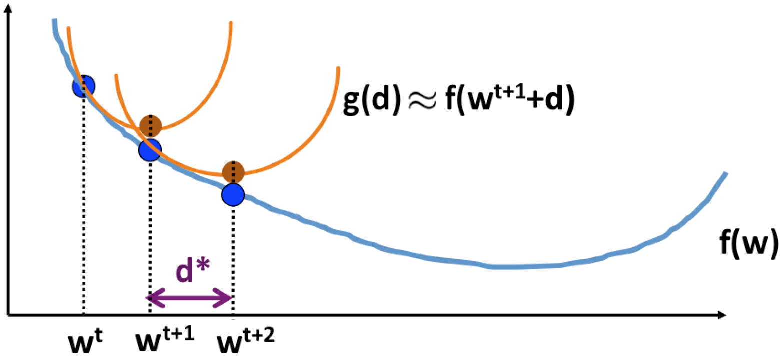


Illustration of gradient descent



Convergence

- Let L be a constant such that $\nabla^2 f(\boldsymbol{x}) \preceq LI$ for all $\boldsymbol{x}$
- **Theorem:** gradient descent converges if $\alpha < \frac{2}{L}$
- Optimal choice: $\alpha < \frac{1}{L}$
- In practice, we do not know $L \cdots$

  need to tune step size when running gradient descent

Applying to Logistic regression

gradient descent for logistic regression

- Initialize the weights $\boldsymbol{w}_0$
- For $t = 1, 2, \cdots$
  - Compute the gradient

$$\nabla f(\boldsymbol{w}) = -\frac{1}{N} \sum_{n=1}^{N} \frac{y_n \boldsymbol{x}_n}{1 + e^{y_n \boldsymbol{w}_t \boldsymbol{x}_n}}$$

  - Update the weights: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla f(\boldsymbol{w})$
- Return the final weights $\boldsymbol{w}$

Applying to Logistic regression

gradient descent for logistic regression

- Initialize the weights $\boldsymbol{w}_0$
- For $t = 1, 2, \cdots$
  - Compute the gradient

$$\nabla f(\boldsymbol{w}) = -\frac{1}{N}\sum_{n=1}^{N} \frac{y_n \boldsymbol{x}_n}{1 + e^{y_n \boldsymbol{w}_t \boldsymbol{x}_n}}$$

  - Update the weights: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla f(\boldsymbol{w})$
- Return the final weights $\boldsymbol{w}$

When to stop?

- Fixed number of iterations, or
- Stop when $\|\nabla f(\boldsymbol{w})\| < \epsilon$

Conclusions

- Linear regression:
  - Square loss ⇒ solving a linear system
  - Closed form solution
- Logistic regression:
  - A classification model based on a probability assumption
- Gradient descent: an iterative solver

Questions?