Logistic Regression

Sriram Sankararaman

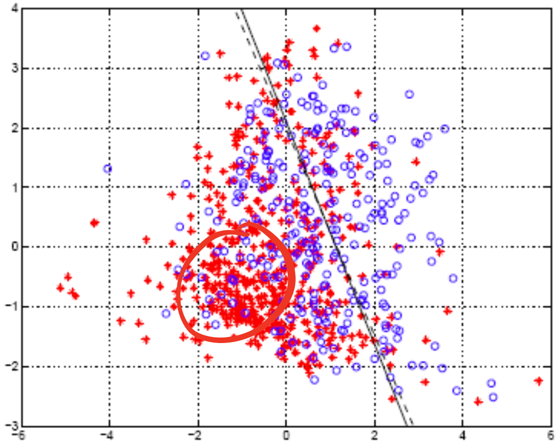
The instructor gratefully acknowledges Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Outline

1. Logistic regression
   - Prediction in logistic regression
   - Training/Learning a logistic regression model
   - Maximum likelihood estimation
2. Optimization
3. Stochastic gradient descent

Review

- Instead of predicting the class, predict the probability of instance being in a class
- Perceptron does not produce probability estimates



Prediction in logistic regression

$x \in \mathbb{R}^D$ $\quad w \in \mathbb{R}^D$

$\in \mathbb{R}$

$$P(y=1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b) = \frac{1}{1+\exp^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)}}$$

$0 \leq \quad \leq 1$

Compute $\sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)$. If this is greater than 0.5, predict 1 else 0.

Decision boundary: Linear or nonlinear?

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

$a = 0$

$\sigma(w^T x + b)$?

$> 0.5 \Rightarrow 1$

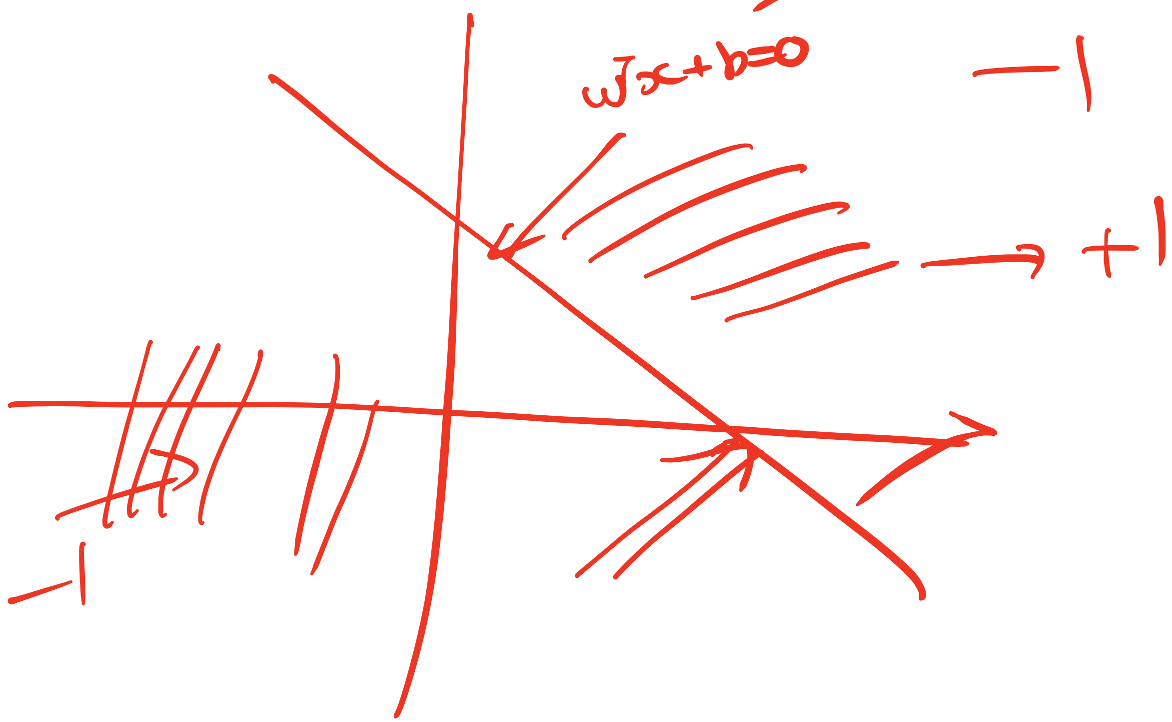
$= 0.5$

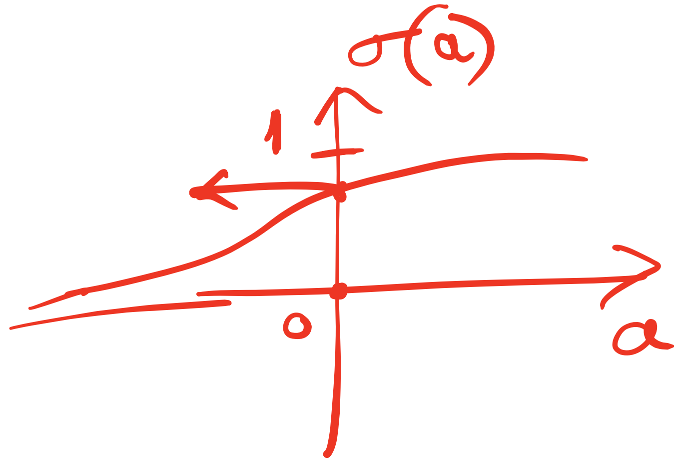
$< 0.5 \Rightarrow 0$

Perceptron

$\text{sign}(w^T x + b) = +1$

-1





$\sigma(w^T x + b) = 0.5$

$w^T x + b = 0$

Decision boundary: Linear or nonlinear?

$\sigma(a)$ **is nonlinear**, however, the decision boundary is determined by

$$\sigma(a) = 0.5 \Rightarrow a(\boldsymbol{x}) = 0 \Rightarrow a(\boldsymbol{x}) = b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} = 0$$

which is a *linear* function in $\boldsymbol{x}$

As in the case of perceptron, b the bias or offset or intercept term.

$\boldsymbol{w}$ the weights .

Logistic regression

Setup for binary classification

- Input: $\boldsymbol{x} \in \mathbb{R}^D$
- Output: $y \in \{0, 1\}$
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, N\}$
- Hypotheses/Model:

$$h_{\boldsymbol{w},b}(x) = p(y = 1|\boldsymbol{x}; b, \boldsymbol{w}) = \sigma(a(\boldsymbol{x}))$$

where

Activation fn.

$$a(\boldsymbol{x}) = b + \sum_d w_d x_d = b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$

- Given training data N samples/instances: $\mathcal{D}^{\mathrm{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{N}}, y_{\mathsf{N}})\}$, train/learn/induce $h_{\boldsymbol{w},b}$. Find values for $(\boldsymbol{w}, b)$.

Example: bag of words

$P(\text{aDaaa} \mid \text{Bag } 1) \gg P(\text{aDaaa} \mid \text{Bag } 2)$

Which bag of words is more likely to generate : aDaaa ?

1 2 ?

Example: bag of words

Which bag of words is more likely to generate : aDaaa ?

$0.7 \times 0.1 \times 0.7 \times 0.7 \times 0.7$
$= 2.401 \times 10^{-2}$

$\gg$

$0.2 \times 0.1 \times 0.2 \times 0.2 \times 0.2$
$= 1.6 \times 10^{-4}$

Example: drawing color cards from an envelope

- An envelope with two colors of cards: yellow and purple.
- Assume if you draw a card at random, probability of drawing a yellow card is θ and probability of drawing a purple card is $1 - \theta$.
- Sample with replacement n times.
- k times we get yellow, $n - k$ times we get purple.
- The joint probability (likelihood) : $\theta^k(1-\theta)^{n-k}$.
- What is the value of θ that maximizes the joint probability?

P(k yellow & n-k purple)

$= P$(yellow in first draw) $\times P$(yellow in 2nd draw) $\times P$(yellow kth draw) $\times P$(purple in k+1) .. P(purple in nth draw)

$$= \underbrace{\theta\,\theta\ldots\theta}_{k}\ \underbrace{(1-\theta)\ldots(1-\theta)}_{(n-k)}$$

Example: drawing color cards from an envelope

Solve $argmax_\theta \theta^k (1-\theta)^{n-k}$

Equivalently, we can solve:

$$argmax_\theta \log(\theta^k (1-\theta)^{n-k}) = \qquad \log(\theta^k) + \log((1-\theta)^{n-k})$$

$$= argmax_\theta k \log\theta + (n-k)\log(1-\theta)$$

$$\frac{d}{d\theta} = 0$$

Example: drawing color cards from an envelope

Solve $argmax_\theta \theta^k (1-\theta)^{n-k}$

Equivalently, we can solve:

$$
\begin{aligned}
& argmax_\theta \log(\theta^k (1-\theta)^{n-k}) \\
& = argmax_\theta k \log\theta + (n-k)\log(1-\theta)
\end{aligned}
$$

At the optimum: $\frac{(k\log\theta + (n-k)\log(1-\theta))}{d\theta} = 0$

Example: drawing color cards from an envelope

Maximum likelihood estimate (MLE): $\hat{\theta} = \frac{k}{n}$

These are easy examples. We don't always have a closed-form solution for the MLE typically !

Likelihood Function

$p(x|\theta)$ $p(x;\theta)$

Let $X_1, \ldots, X_N$ be IID (independent and identically distributed) random variables with PDF $p(x|\theta)$ (also written as $p(x;\theta)$). The *likelihood function* is defined by $L(\theta)$,

$$L(\theta) = p(X_1, \ldots, X_N; \theta). \qquad = \prod_{i=1}^{N} p(X_i; \theta).$$

Notes The likelihood function is just the joint density of the data, except that we treat it as a function of the parameter θ.

Maximum Likelihood Estimator

Definition: The maximum likelihood estimator (MLE) $\hat{\theta}$, is the value of θ that maximizes $L(\theta)$.

The log-likelihood function is defined by $l(\theta) = \log L(\theta)$. Its maximum occurs at the same place as that of the likelihood function.

Maximum Likelihood Estimator

$L(\theta) = \prod_{i=1}^{n} P(x_i;\theta)$

$\ell(\theta) = \log(L(\theta)) = \log\left(\prod_{i=1}^{n} P(x_i;\theta)\right)$

$= \sum_{i=1}^{n} \log P(x_i;\theta)$

Definition: The maximum likelihood estimator (MLE) $\hat{\theta}$, is the value of θ that maximizes $L(\theta)$.

The log-likelihood function is defined by $l(\theta) = \log L(\theta)$. Its maximum occurs at the same place as that of the likelihood function.

- Using logs simplifies mathematical expressions (converts exponents to products and products to sums)
- Using logs helps with numerical stability

The same is true of the likelihood function times any constant. Thus we shall often drop constants in the likelihood function.

$c\,L(\theta)$ $\quad$ $L(\theta)$

Likelihood function for logistic regression

$P(Y=1 \mid x; b, w) = \sigma(w^T x + b)$

$P(Y=0 \mid x; b, w) + P(Y=1 \mid x; b, w) = 1 \Rightarrow P(Y=0 \mid x; b, w) = 1 - \sigma(w^T x + b)$

Probability of a single training sample $(\boldsymbol{x}_n, y_n)$

$$p(y_n|\boldsymbol{x}_n; b, \boldsymbol{w}) = \begin{cases} h_{\boldsymbol{w},b}(\boldsymbol{x}_n) = \sigma(b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) & \text{if} \quad y_n = 1 \\ = 1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n) = 1 - \sigma(b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) & \text{otherwise} \end{cases}$$

Likelihood function for logistic regression

Probability of a single training sample $(\boldsymbol{x}_n, y_n)$

$$p(y_n|\boldsymbol{x}_n; b, \boldsymbol{w}) = \begin{cases} h_{\boldsymbol{w},b}(\boldsymbol{x}_n) = \sigma(b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) & \text{if} \quad y_n = 1 \\ = 1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n) = 1 - \sigma(b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) & \text{otherwise} \end{cases}$$

Compact expression, exploring that y_n **is either 1 or 0**

$$p(y_n|\boldsymbol{x}_n; b; \boldsymbol{w}) = h_{\boldsymbol{w},b}(\boldsymbol{x}_n)^{y_n}[1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]^{1-y_n}$$

$y_n = 1$

$$(h_{w,b}(x_n))^1 (1 - h_{w,b}(x_n))^0$$

$$= h_{w,b}(x_n) \cdot 1$$

Log Likelihood

$$\log(L(w,b)) = \log\left(\prod_{i=1}^{n} P(y_n \mid x_n; w, b)\right)$$

Log-likelihood of the whole training data $\mathcal{D}$

$$l(\boldsymbol{w}, b) = \sum_n \{y_n \log h_{\boldsymbol{w},\boldsymbol{b}}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]\}$$

$$l(w,b) = \sum_{n=1}^{N} \log P(y_n \mid x_n; w, b)$$

$$= \sum_{n=1}^{N} \log\left(h_{w,b}(x_n)^{y_n} (1 - h_{w,b}(x_n))^{(1-y_n)} \right)$$

$$= \sum_{n=1}^{N} \log\left(h_{w,b}(x_n)^{y_n}\right) + \log\left((1 - h_{w,b}(x_n))^{(1-y_n)}\right)$$

$$\log(a^b) = b \log(a)$$

Log Likelihood

Log-likelihood of the whole training data $\mathcal{D}$

$$l(\boldsymbol{w}, b) = \sum_n \{y_n \log h_{\boldsymbol{w},\boldsymbol{b}}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]\}$$

It is convenient to work with its negation termed negative log likelihood

$$J(b, \boldsymbol{w}) = -\sum_n \{y_n \log h_{\boldsymbol{w},b}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]\}$$

We can ignore the distinction between bias and weights

(b, w)

This is for convenience

- Append 1 to $\boldsymbol{x}$
$$\boldsymbol{x} \leftarrow [1 \quad x_1 \quad x_2 \quad \cdots \quad x_D]$$
- Append b to $\boldsymbol{w}$
$$\boldsymbol{\theta} \leftarrow [b \quad w_1 \quad w_2 \quad \cdots \quad w_D]$$
- $$J(\boldsymbol{\theta}) = -\sum_n \{y_n \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]\}$$
- Same trick as in the case of perceptrons
  - we are rewriting a hyperplane in D dimensions as one in $D+1$ dimensions that passes through the origin.

How to find the optimal parameters for logistic regression?

$$\hat{\theta} = \arg\max_{\theta} \ell(\theta)$$

MLE

$$= \arg\min_{\theta} -\ell(\theta)$$

We will minimize the negative log likelihood

$$J(\boldsymbol{\theta}) = -\sum_n \{y_n \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]\}$$

How to find the optimal parameters for logistic regression?

We will minimize the negative log likelihood

$$J(\boldsymbol{\theta}) = -\sum_n \{y_n \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + (1 - y_n)\log[1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]\}$$

How do we find its minimum?

$$\nabla J(\theta) = 0$$

$$\theta = \underbrace{(\theta_0\ \theta_1 \ldots \theta_D)}_{(D+1)}$$

$$\nabla J(\theta) = \begin{pmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \vdots \\ \frac{\partial J}{\partial \theta_D} \end{pmatrix} \Longrightarrow$$

Outline

1. Logistic regression
2. Optimization
3. Stochastic gradient descent

Maximum Likelihood

$x_1, \dots, x_n$ $\hat{\theta}$

$x_i \sim P(x; \theta)$

$P(X_i = \text{Yellow}) = \cancel{0.2}$

$P(X_i = \text{Purple}) = \cancel{0.8}$

θ_{true}

$P(X_1, \dots, X_n \mid \theta = 0.1)$

$P(X_1 \dots X_n \mid \theta = 0.8)$

$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, L(\theta)$, $L = P(X_1 \dots X_n \mid \theta)$

Optimization

Given a function $f(x)$, find its minimum (or maximum).

- f is called the objective function.

Optimization

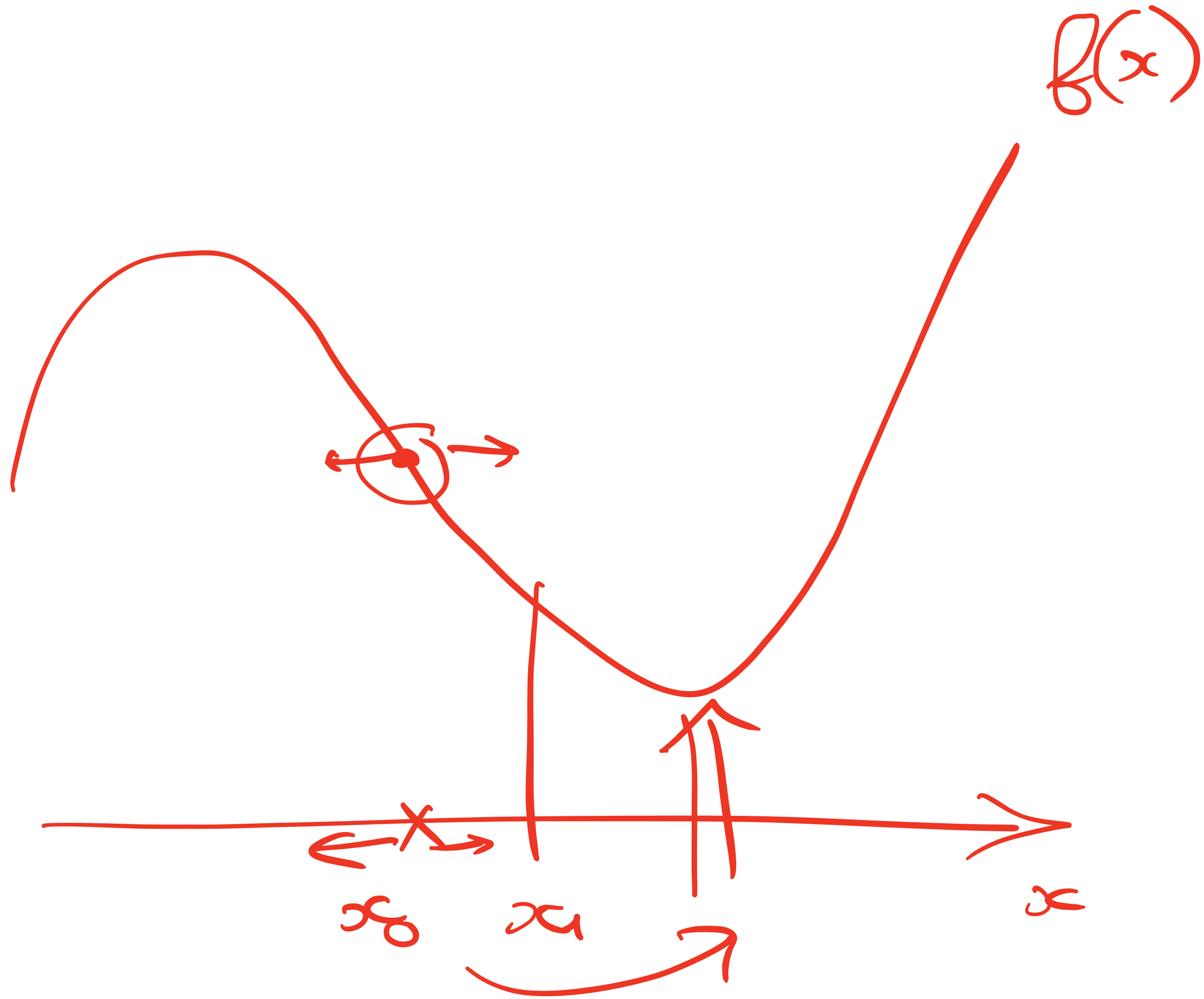
Given a function $f(x)$, find its minimum (or maximum).

- f is called the objective function.
- Maximizing f is equivalent to minimizing $-f$.

  So we only need to consider minimization problems.

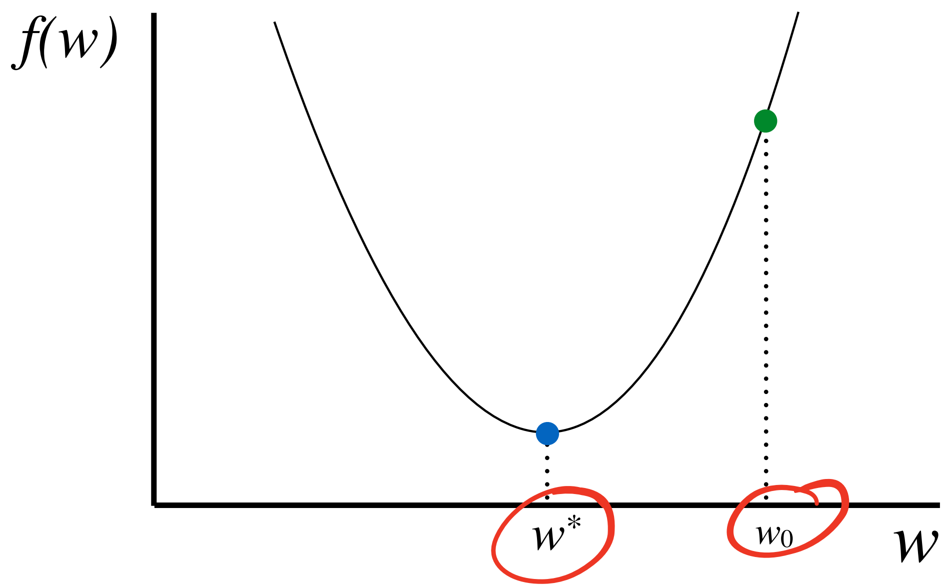
One way to minimize a function f

Gradient descent



Gradient Descent

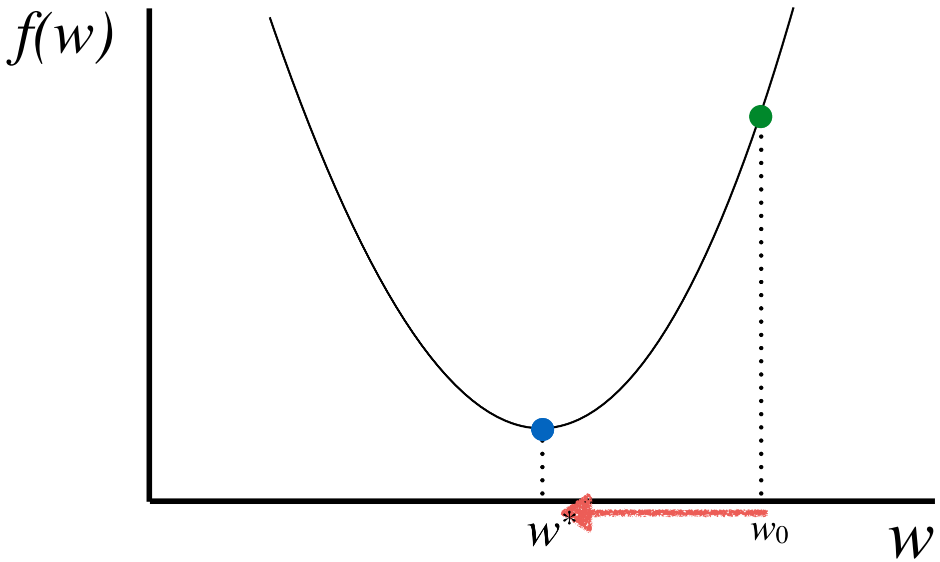
Start at a random point



Gradient Descent

Start at a random point

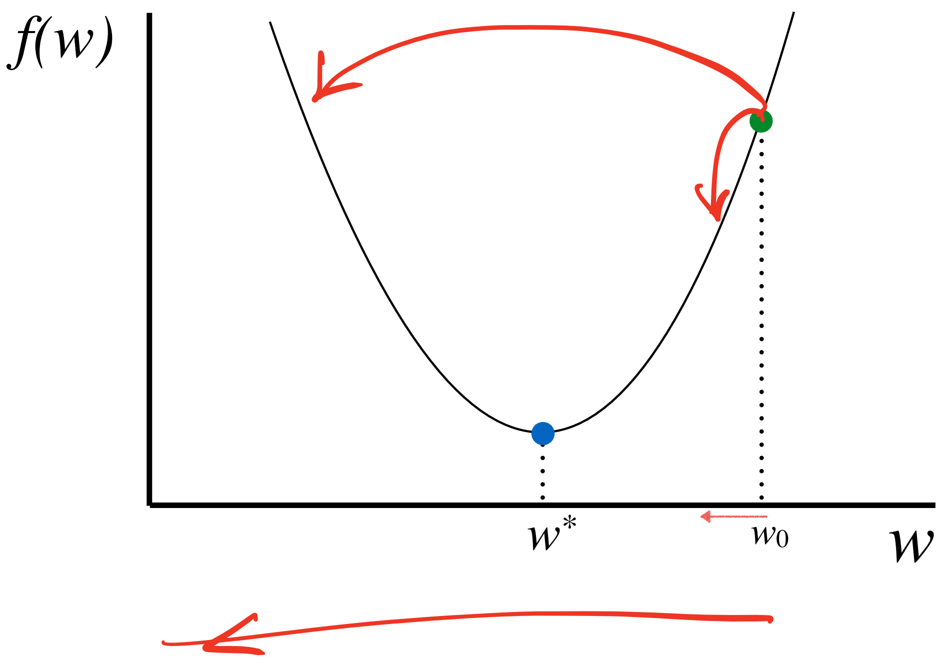
Determine a descent direction



Gradient Descent

Start at a random point

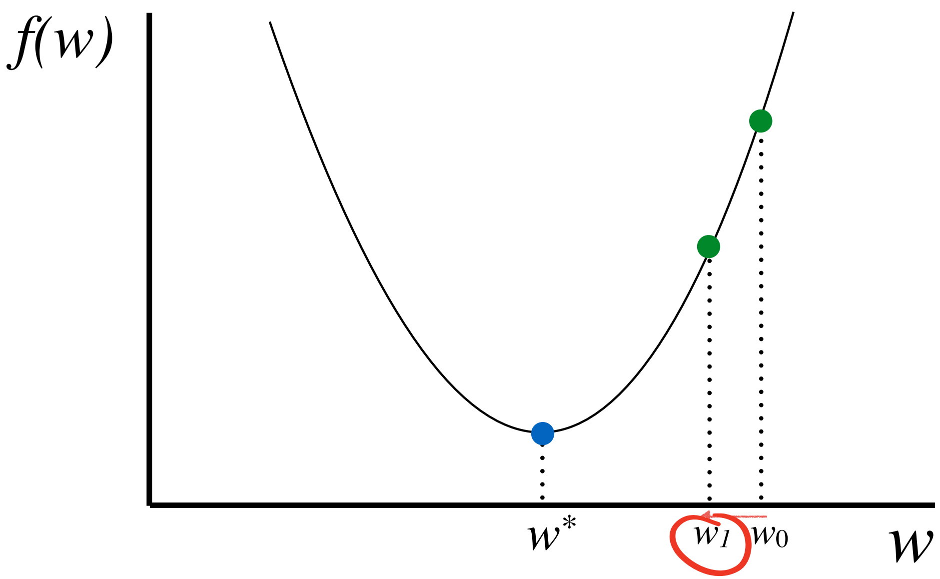
Determine a descent direction
Choose a step size



Gradient Descent

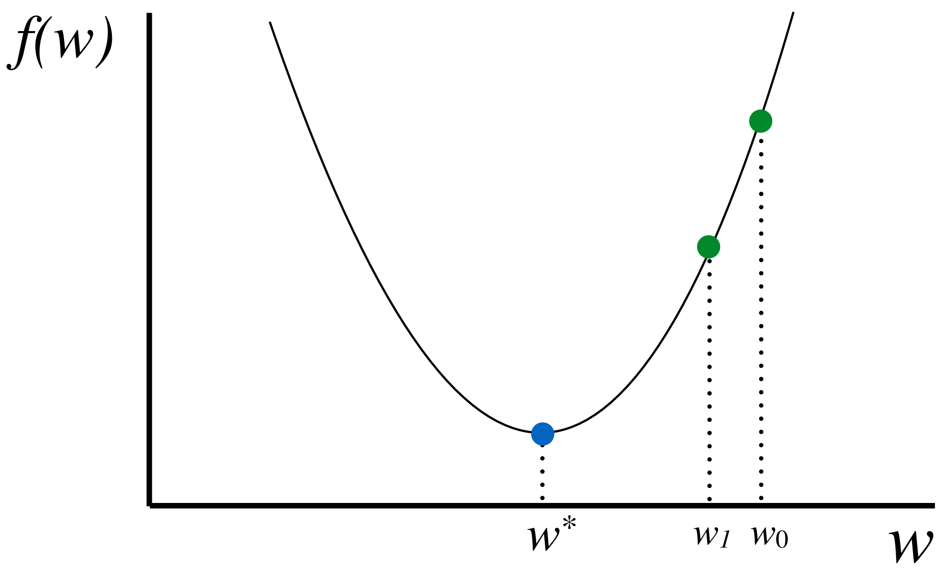
Start at a random point

Determine a descent direction
Choose a step size
Update



Gradient Descent

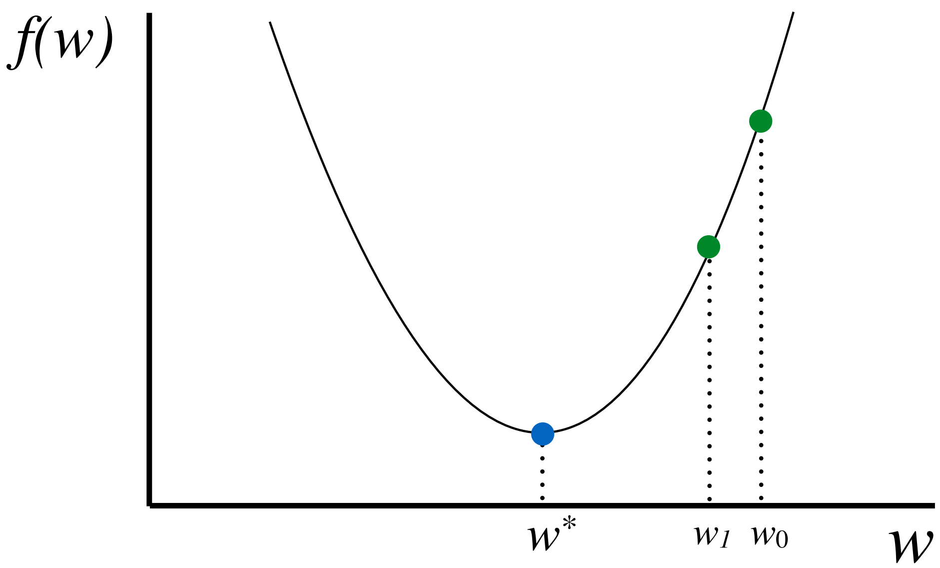
Start at a random point
Repeat
    Determine a descent direction
    Choose a step size
    Update
Until stopping criterion is satisfied



Gradient Descent

Start at a random point
Repeat
- Determine a descent direction
- Choose a step size
- Update

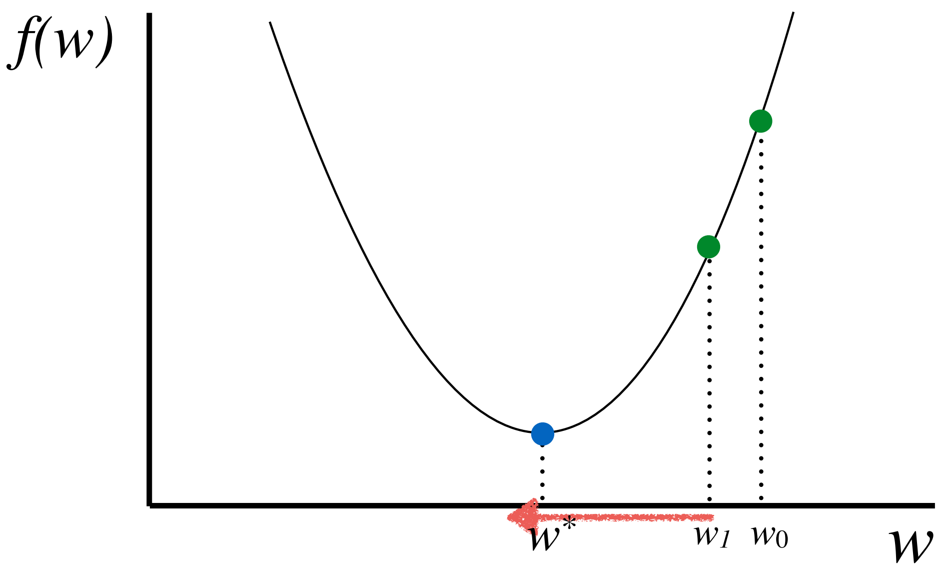
Until stopping criterion is satisfied



Gradient Descent

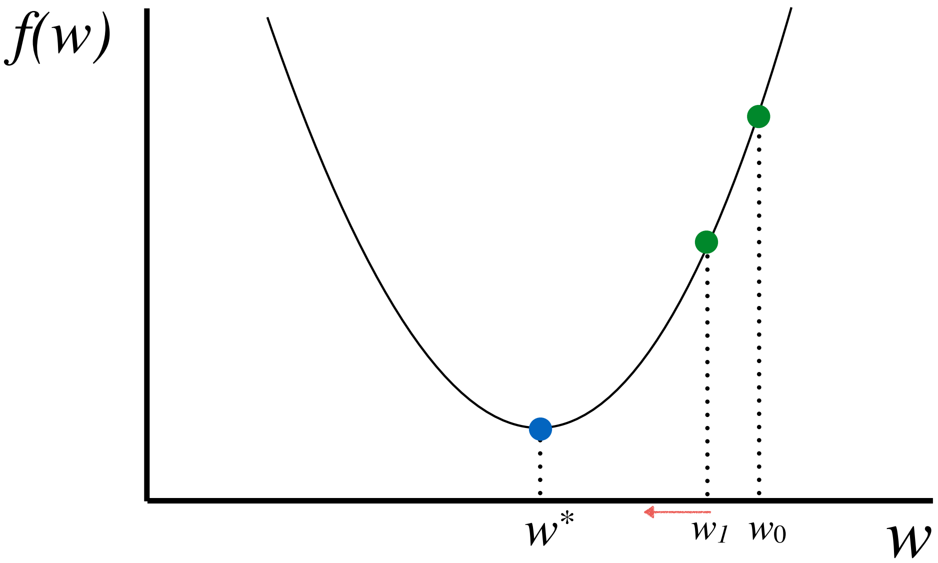
Start at a random point
Repeat
- Determine a descent direction
- Choose a step size
- Update

Until stopping criterion is satisfied



Gradient Descent

Start at a random point
Repeat
    Determine a descent direction
    Choose a step size
    Update
Until stopping criterion is satisfied



Gradient Descent

Start at a random point

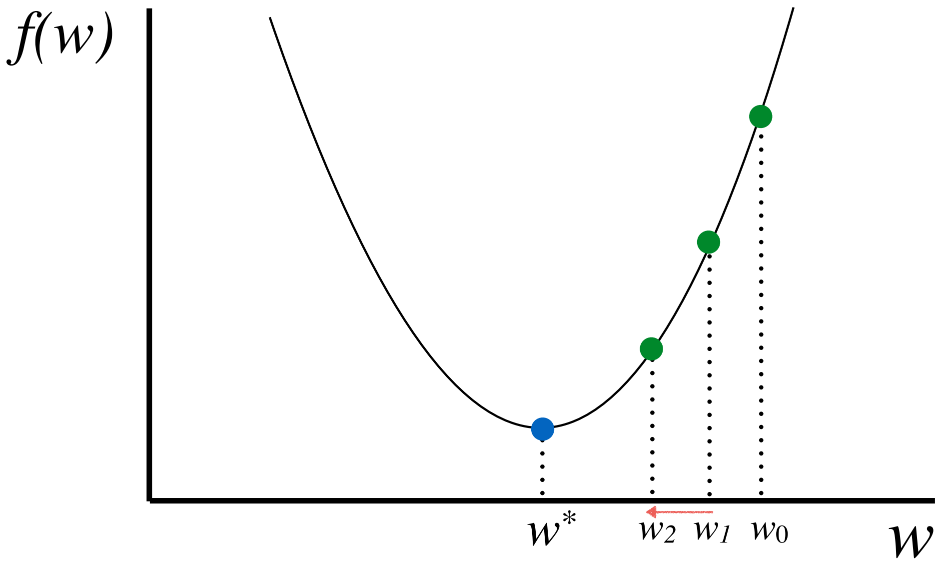
Repeat

Determine a descent direction

Choose a step size

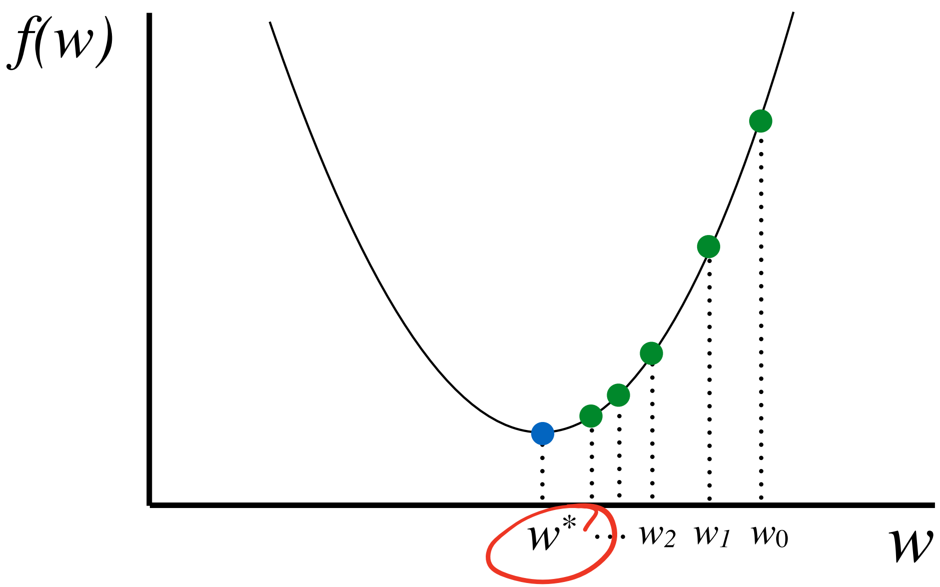
Update

Until stopping criterion is satisfied

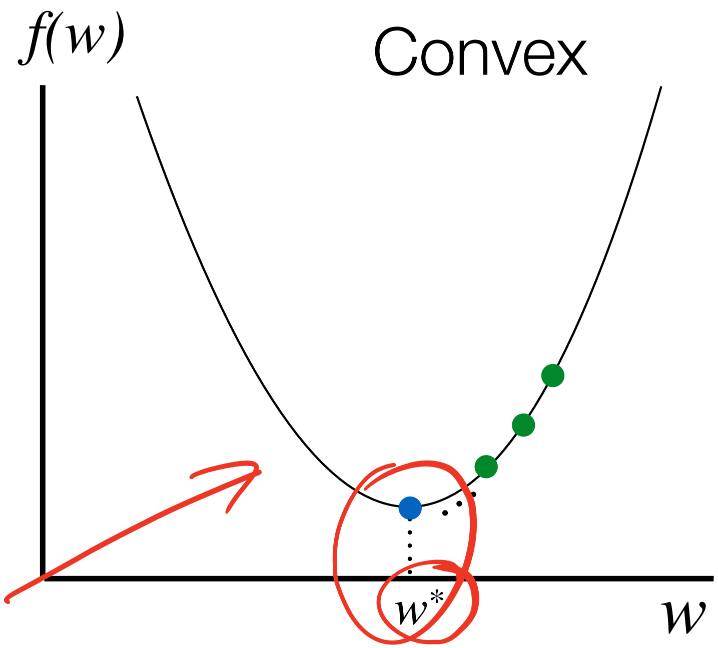


Gradient Descent

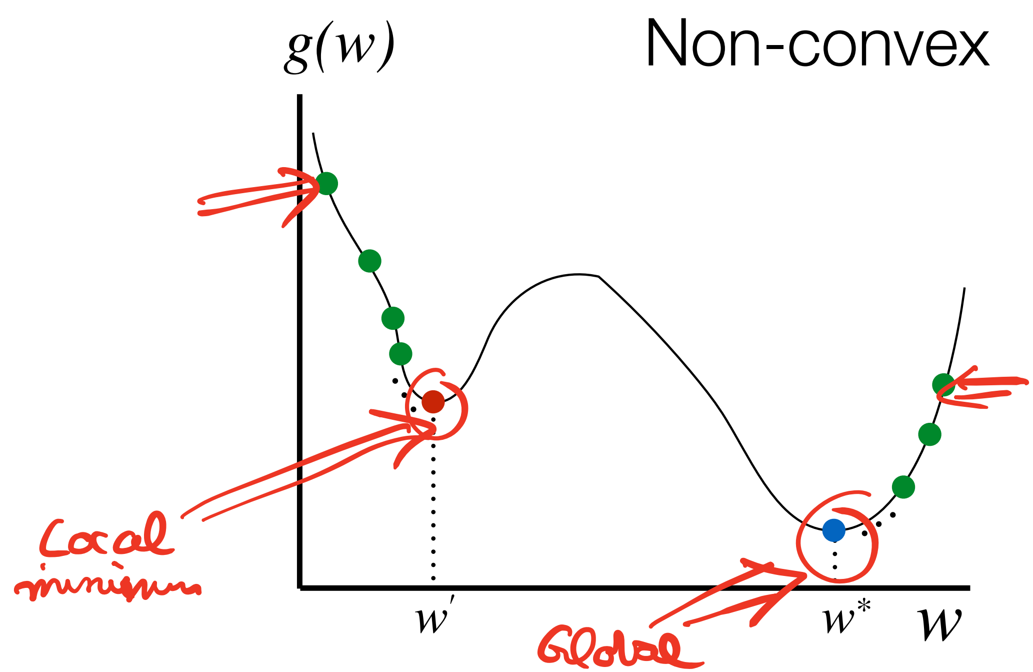
Start at a random point
Repeat
  Determine a descent direction
  Choose a step size
  Update
Until stopping criterion is satisfied



Where Will We Converge?



Any local minimum is a global minimum



Multiple local minima may exist

Least Squares, Ridge Regression and Logistic Regression are all convex!

Convex functions

$f(\lambda a + (1-\lambda) b)$ $\leq$

A function $f(x)$ is convex if

$$f(\lambda a + (1-\lambda)b) \leq \lambda f(a) + (1-\lambda) f(b)$$

for

$x_\lambda = \lambda a + (1-\lambda) b$

$$0 \leq \lambda \leq 1$$

$0 \leq \lambda \leq 1$

$\lambda = 1$

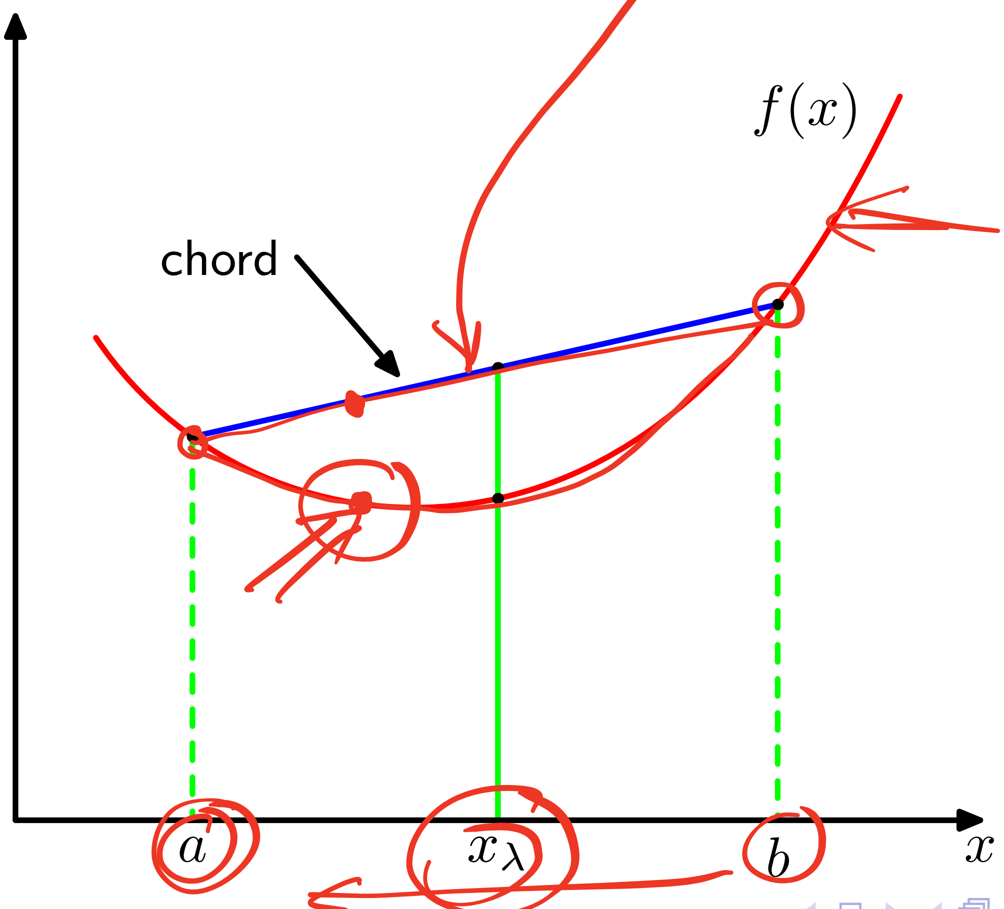
$x = a$

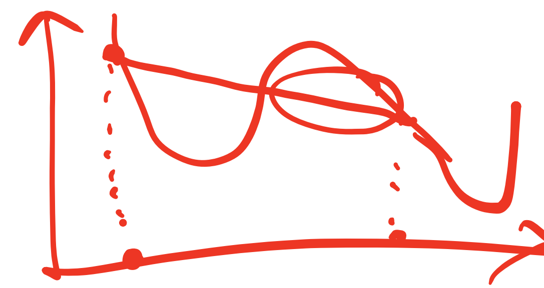
$\lambda = 0$

$x = b$

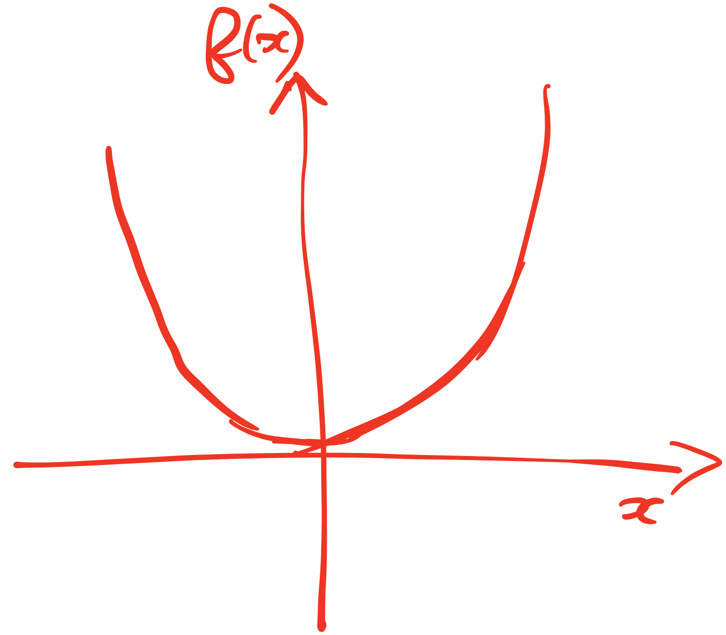
$\lambda = \frac{1}{2}$

$x = \frac{1}{2}a + \frac{1}{2}b$





How to determine convexity?



$f(x)$ is convex if

$$f''(x) \geq 0$$

Examples:

$$f(x) = x^2, f''(x) = 2 > 0$$

$f''(x) = 2$ $\qquad$ $f'(x) = 2x$

Examples

Convex functions

$$f(x) = ax + b$$

$f''(x) = 0 \geq 0$

$$f(x) = x^2$$

$$f(x) = e^x$$

$f''(x) = e^x \geq 0$

$$f(x) = \frac{1}{x}, x \geq 0$$

Examples

Nonconvex functions

$$f(x) = \cos(x)$$

$$f(x) = e^x - x^2$$

$$f(x) = \log(x)$$

Multi-variate functions

Definition

$f(\boldsymbol{x})$ is convex

$$f(\lambda \boldsymbol{a} + (1-\lambda)\boldsymbol{b}) \leq \lambda f(\boldsymbol{a}) + (1-\lambda) f(\boldsymbol{b})$$

for all $\boldsymbol{a}$, $\boldsymbol{b}$, $0 \leq \lambda \leq 1$

Multi-variate functions

$$\frac{\partial^2 f}{\partial x_i \partial x_j}$$

How to determine convexity in this case?

$f''(x)$

$= \frac{d^2 f(x)}{dx^2} \geq 0$

Matrix of second-order derivatives (Hessian)

$$\boldsymbol{H} = \begin{pmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial {x_1}^2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_D} \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_D} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_D} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_D} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_D^2} \end{pmatrix}$$

$x_1 x_3$

$D \times D$

Multi-variate functions

How to determine convexity in this case?

If the Hessian is positive semi-definite $\boldsymbol{H} \succeq 0$, then f is convex.

A matrix $\boldsymbol{H}$ is positive semi-definite if and only if

$$\boldsymbol{z}^T \boldsymbol{H} \boldsymbol{z} = \sum_{j,k} H_{j,k} z_j z_k \geq 0$$

for all $\boldsymbol{z}$.

$\underset{D \times D}{H}$ $\underset{D}{z}$

$\underset{1 \times D}{z^T} \underset{D \times D}{H} \underset{D \times 1}{z} \geq 0$

Multi-variate functions

$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

Example

$$f(\boldsymbol{x}) = x_1^2 + 2x_2^2$$

$$\boldsymbol{H} = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$$

symmetric

$$z^T H z = 2z_1^2 + 4z_2^2$$

$$\geq 0$$

Multi-variate functions

Example

$$f(\boldsymbol{x}) = x_1^2 + 2x_2^2$$

$$\boldsymbol{H} = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$$

$$\boldsymbol{z}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{z} = 2z_1^2 + 4z_2^2 \geq 0$$

Example: $\min f(\boldsymbol{\theta}) = 0.5(\theta_1^2 - \theta_2)^2 + 0.5(\theta_1 - 1)^2$

- We compute the gradients

$$\frac{\partial f}{\partial \theta_1} = 2(\theta_1^2 - \theta_2)\theta_1 + \theta_1 - 1 \quad (1)$$

$$\frac{\partial f}{\partial \theta_2} = -(\theta_1^2 - \theta_2) \quad (2)$$

- Use the following *iterative* procedure for *gradient descent*

1. Initialize $\theta_1^{(0)}$ and $\theta_2^{(0)}$, and $t = 0$
2. do

$$\theta_1^{(t+1)} \leftarrow \theta_1^{(t)} - \eta \left[2({\theta_1^{(t)}}^2 - \theta_2^{(t)})\theta_1^{(t)} + \theta_1^{(t)} - 1 \right] \quad (3)$$

$$\theta_2^{(t+1)} \leftarrow \theta_2^{(t)} - \eta \left[-({\theta_1^{(t)}}^2 - \theta_2^{(t)}) \right] \quad (4)$$

$$t \leftarrow t + 1 \quad (5)$$

3. until $f(\boldsymbol{\theta}^{(t)})$ *does not change much*

Gradient descent

General form for minimizing $f(\boldsymbol{\theta})$

$$\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^{t} - \eta \nabla f(\boldsymbol{\theta}^{t})$$

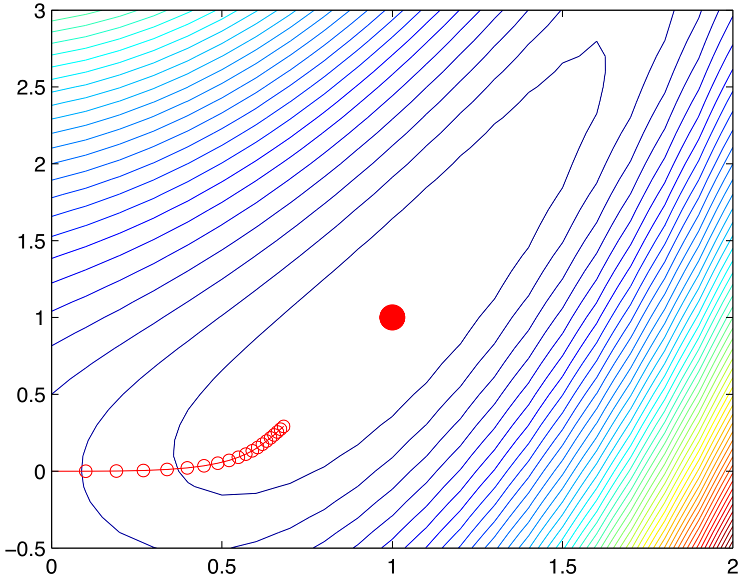
Remarks

- η is often called *step size* – literally, how far our update will go along the the direction of the negative gradient
- Note that this is for *minimizing* a function, hence the subtraction $(-\eta)$
- With a *suitable* choice of η, the iterative procedure converges to a stationary point where
$$\nabla f(\boldsymbol{\theta}) = 0$$
- A stationary point is only necessary for being the minimum.

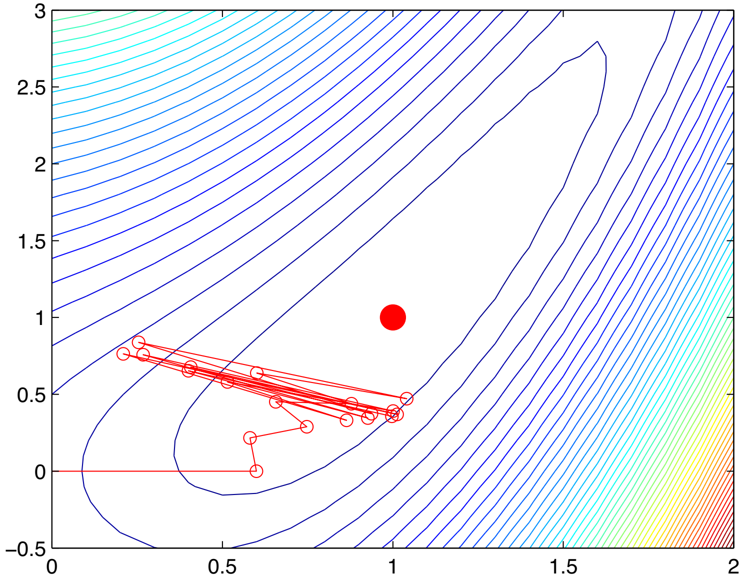
Seeing in action

Choosing the right η is important

small η is too slow?



large η is too unstable?



Gradient descent

$$\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_\mathsf{N}, y_\mathsf{N})\}$$

Algorithm 1 Gradient descent

1: $\boldsymbol{\theta} \leftarrow \mathbf{0}$.
2: **for** $epoch = 1 \ldots T$ **do**
3: $\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla J(\boldsymbol{\theta})$
4: **end for**
5: **return** $\boldsymbol{\theta}$

Gradient Descent Update for Logistic Regression

Derivatives of $\sigma(a)$

$$\frac{d\,\sigma(a)}{d\,a} = \frac{d}{d\,a}\left(1+e^{-a}\right)^{-1} = \frac{-(1+e^{-a})'}{(1+e^{-a})^2}$$

$$= \frac{e^{-a}}{(1+e^{-a})^2} = \frac{1}{1+e^{-a}}\frac{e^{-a}}{1+e^{-a}}$$

$$= \sigma(a)[1-\sigma(a)]$$

Gradients of the negative log likelihood

Negative log likelihood

$$J(\boldsymbol{\theta}) = -\sum_n \{y_n \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]\}$$

Gradients

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\sum_n \left\{y_n[1 - \sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\boldsymbol{x}_n - (1 - y_n)\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\boldsymbol{x}_n\right\} \quad (6)$$

$$= \sum_n \left\{\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n) - y_n\right\} \boldsymbol{x}_n \quad (7)$$

$$= \sum_n \left\{h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - y_n\right\} \boldsymbol{x}_n \quad (8)$$

Remark

Gradients of the negative log likelihood

Negative log likelihood

$$J(\boldsymbol{\theta}) = -\sum_n \{y_n \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + (1-y_n)\log[1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]\}$$

Gradients

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\sum_n \left\{y_n[1-\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\boldsymbol{x}_n - (1-y_n)\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\boldsymbol{x}_n\right\} \quad (6)$$

$$= \sum_n \left\{\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n) - y_n\right\}\boldsymbol{x}_n \quad (7)$$

$$= \sum_n \left\{h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - y_n\right\}\boldsymbol{x}_n \quad (8)$$

Remark

- $e_n = \{h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - y_n\}$ is called *error* for the nth training sample.

Numerical optimization

Gradient descent

- Choose a proper step size $\eta > 0$
- Iteratively update the parameters following the negative gradient to minimize the error function

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \sum_n \left\{\sigma(\boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}_n) - y_n\right\} \boldsymbol{x}_n$$

Remarks

- The step size needs to be chosen carefully to ensure convergence.
- The step size can be adaptive (i.e. varying from iteration to iteration). For example, we can use techniques such as *line search*
- There is a variant called *stochastic* gradient descent, also popularly used.

Outline

Stochastic gradient descent

$$\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_\mathsf{N}, y_\mathsf{N})\}$$

$$J(\boldsymbol{\theta}) = -\sum_n \{y_n \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + (1 - y_n) \log[1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]\}$$

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_n J_n(\boldsymbol{\theta})$$

$$\nabla J(\boldsymbol{\theta}) = \frac{1}{N} \sum_n \nabla J_n(\boldsymbol{\theta}) = \mathbb{E}_{n \sim \mathcal{D}} \nabla J_n(\boldsymbol{\theta})$$

Approximate the gradient by the gradient computed on a single example at a time.

- Repeat until convergence
  - Randomly pick one example $(\boldsymbol{x}_n, y_n)$.
  - Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla J_n(\boldsymbol{\theta})$.

Stochastic Gradient descent (SGD)

$\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_\mathsf{N}, y_\mathsf{N})\}$

Algorithm 2 Stochastic Gradient descent

$\boldsymbol{\theta} \leftarrow \mathbf{0}$.
for $epoch = 1 \ldots T$ **do**
    for $(\boldsymbol{x}, y) \in \mathcal{D}$ **do**
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla J_{(\boldsymbol{x}, y)}(\boldsymbol{\theta})$
    end for
end for
return $\boldsymbol{\theta}$

Compare to perceptron learning

$\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_\mathsf{N}, y_\mathsf{N})\}$

Algorithm 3 PerceptronTrain

```
θ ← 0
for iter = 1 ... MaxIter do
    for (x, y) ∈ D do
        a ← θᵀx
        if ay ≤ 0 then
            θ ← θ + yx
        end if
    end for
end for
return θ
```

What is the objective function?

Compare to perceptron learning

$\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_\mathsf{N}, y_\mathsf{N})\}$

Algorithm 4 PerceptronTrain

$\boldsymbol{\theta} \leftarrow \mathbf{0}$
for $iter = 1 \ldots MaxIter$ **do**
  for $(\boldsymbol{x}, y) \in \mathcal{D}$ **do**
    $a \leftarrow \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}$
    if $ay \leq 0$ **then**
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y\boldsymbol{x}$
    end if
  end for
end for
return $\boldsymbol{\theta}$

Perceptron effectively minimizes:

$$J(\boldsymbol{\theta}) = \sum_n max(0, 1 - y_n \boldsymbol{\theta}^T \boldsymbol{x}_n)$$

Summary

Setup for binary classification

- Logistic Regression models conditional distribution as: $p(y=1|\boldsymbol{x};\boldsymbol{\theta}) = \sigma[a(\boldsymbol{x})]$ where $a(\boldsymbol{x}) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}$
- Linear decision boundary: $a(\boldsymbol{x}) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x} = 0$

Minimizing the negative log-likelihood

- $J(\boldsymbol{\theta}) = -\sum_n \{y_n \log \sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n) + (1-y_n)\log[1-\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\}$
- No closed form solution; must rely on iterative solvers

Numerical optimization

- Gradient descent: simple, scalable to large-scale problems
  - move in direction opposite of gradient!
  - gradient of logistic function takes nice form