Perceptron (recap), 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

Perceptron learning

Special case: binary classification

- Instance (feature vectors): $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$
- Label: $y \in \{-1, +1\}$
- Model/Hypotheses:
  $H = \{h | h : \mathbb{X} \rightarrow \mathbb{Y}, h(\boldsymbol{x}) = sign(\sum_{d=1}^{D} w_d x_d + b)\}$.
- Learning goal: $y = h(\boldsymbol{x})$

Perceptron learning

Special case: binary classification

- Instance (feature vectors): $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$
- Label: $y \in \{-1, +1\}$
- Model/Hypotheses:
  $H = \{h | h : \mathbb{X} \to \mathbb{Y}, h(\boldsymbol{x}) = sign(\sum_{d=1}^{D} w_d x_d + b)\}$.
- Learning goal: $y = h(\boldsymbol{x})$
  - Learn $w_1, \ldots, w_D, b$.
  - Parameters: $w_1, \ldots, w_D, b$.
  - $\boldsymbol{w}$: weights, b: bias

Perceptron predict

- Input: $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$, $\boldsymbol{w} \in \mathbb{R}^{\mathsf{D}}$, $b \in \mathbb{R}$.

$$a = \sum_{d=1}^{D} w_d x_d + b = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b$$
$$\hat{y} = sign(a)$$

- Output: $\hat{y}$.
- $\sum_{d=1}^{D} w_d x_d + b$: hyperplane in D dimensions with parameters $(\boldsymbol{w}, b)$.
- $\boldsymbol{w}$: weights, b: bias
- a: activation

Perceptron learning

Iterative solving one case at a time

- REPEAT
- Pick a data point $\boldsymbol{x}_n$
- Compute $a = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n$ using the *current* $\boldsymbol{w}$
- If $ay_n > 0$, do nothing. Else,

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y_n\boldsymbol{x}_n$$

- UNTIL converged.

Properties of perceptron learning

- This is an online algorithm – looks at one instance at a time.
- Convergence
  - If training data is not linearly separable, the algorithm does not converge.
  - If the training data is linearly separable, the algorithm stops in a finite number of steps (converges).
- How long to convergence ?
  - Depends on the difficulty of the problem.
- Use $MaxIter$ hyperparameter to determine when to stop.

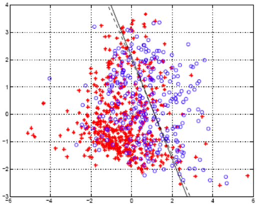
Perceptron

- Extensions
  - Voting
  - Averaging
- Limitations
  - Linear separability
- Interpreting the importance of features
  - The values of weight w_d tells us the importance of feature x_d.

Outline

Review

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



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

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

and $\sigma(\cdot)$ stands for the *sigmoid* function

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

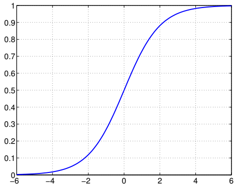
Why the sigmoid function?

What does it look like?

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

where

$$a = b + \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}$$



Properties

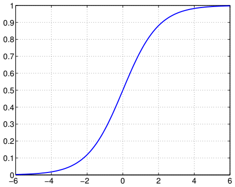
Why the sigmoid function?

What does it look like?

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

where

$$a = b + \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}$$



Properties

- Bounded between 0 and 1 $\leftarrow$ thus, interpretable as probability
- Monotonically increasing thus, usable to derive classification rules
  - $\sigma(a) > 0.5$, positive (classify as '1')
  - $\sigma(a) < 0.5$, negative (classify as '0')
  - $\sigma(a) = 0.5$, undecidable
- Nice computational properties As we will see soon

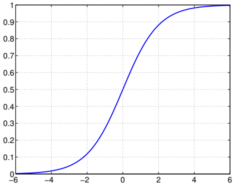
Why the sigmoid function?

What does it look like?

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

where

$$a = b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$



Properties

- Bounded between 0 and 1 $\leftarrow$ thus, interpretable as probability
- Monotonically increasing thus, usable to derive classification rules
  - $\sigma(a) > 0.5$, positive (classify as '1')
  - $\sigma(a) < 0.5$, negative (classify as '0')
  - $\sigma(a) = 0.5$, undecidable
- Nice computational properties As we will see soon

Linear or nonlinear classifier?

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

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

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

Likelihood Function

Let $X_1, \dots, X_N$ 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 $L(\theta)$,

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

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

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 mathemetical expressions (converts exponents to products and products to sums)
- Using logs helps with numerical stabilitity

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

Likelihood function for logistic regression

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

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

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

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

Log Likelihood

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

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

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

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?

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?

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 $-f$.

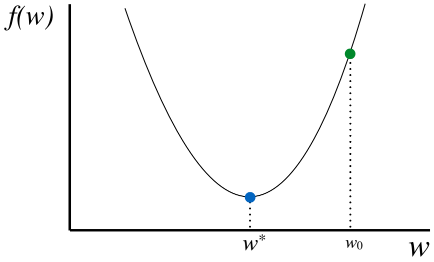
  So we only need to consider minimization problems.

One way to minimize a function f

Gradient descent

Gradient Descent

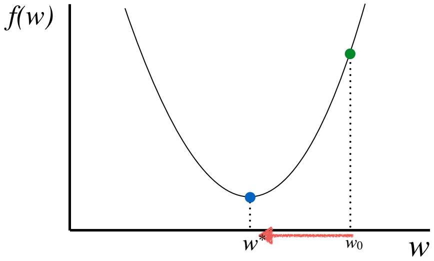
Start at a random point



Gradient Descent

Start at a random point

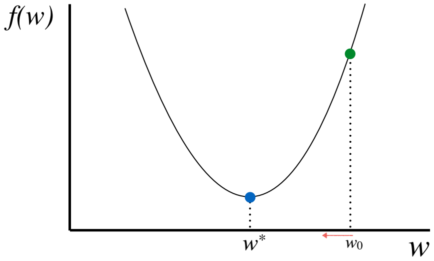
Determine a descent direction



Gradient Descent

Start at a random point

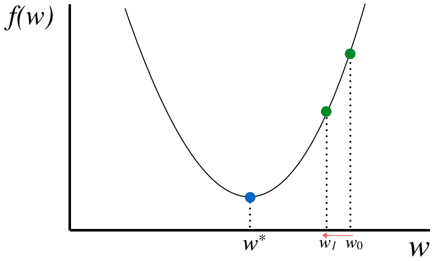
- Determine a descent direction
- Choose a step size



Gradient Descent

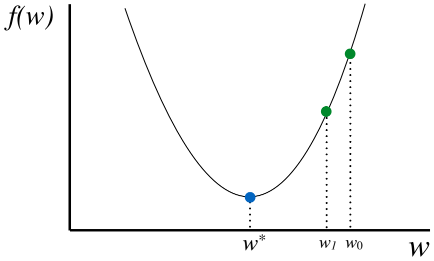
Start at a random point

- Determine a descent direction
- Choose a step size
- Update



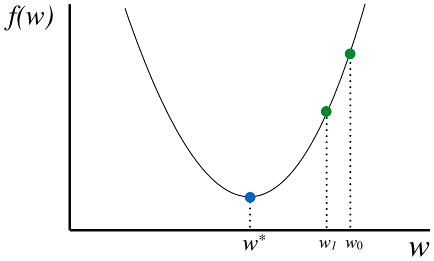
Gradient Descent

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



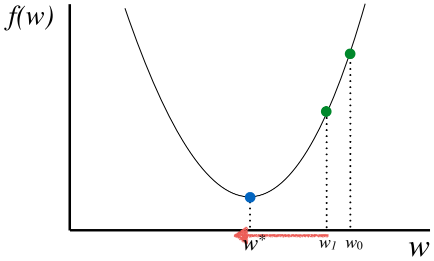
Gradient Descent

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



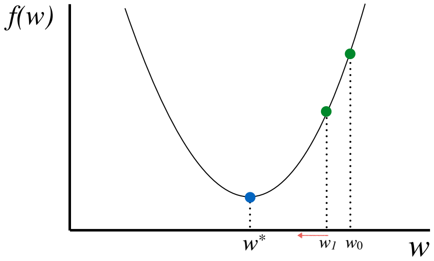
Gradient Descent

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



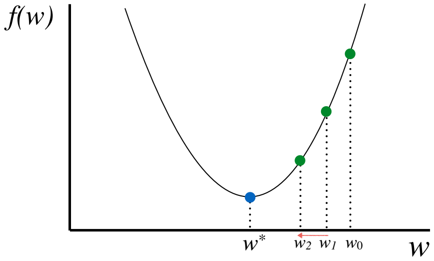
Gradient Descent

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



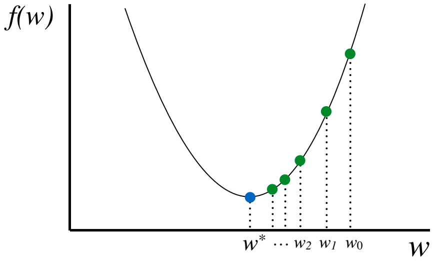
Gradient Descent

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

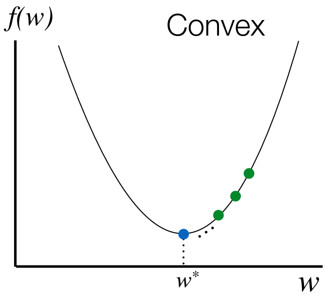


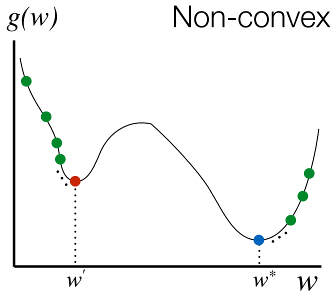
Gradient Descent

```
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!

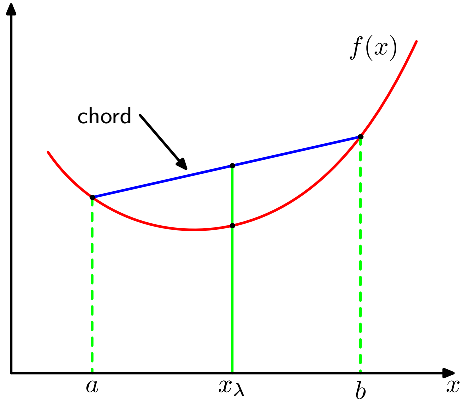
Convex functions

A function $f(x)$ is convex if

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

for

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



How to determine convexity?

$f(x)$ is convex if

$$f^{''}(x) \geq 0$$

Examples:

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

Examples

Convex functions

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

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

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

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

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

How to determine convexity in this case?

Matrix of second-order derivatives (Hessian)

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

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 $\boldsymbol{z}$.

Multi-variate functions

Example

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

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

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

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

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

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

$$\theta_1^{(t+1)} \leftarrow \theta_1^{(t)} - \eta \left[2({\theta_1^{(t)}}^2 - \theta_2^{(t)})\theta_1^{(t)} + \theta_1^{(t)} - 1 \right] \qquad (3)$$
$$\theta_2^{(t+1)} \leftarrow \theta_2^{(t)} - \eta \left[-({\theta_1^{(t)}}^2 - \theta_2^{(t)}) \right] \qquad (4)$$
$$t \leftarrow t + 1 \qquad (5)$$

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

Gradient descent

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

$$\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta} - \eta \frac{\partial f}{\partial \boldsymbol{\theta}}$$

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
$$\frac{\partial f}{\partial \boldsymbol{\theta}} = 0$$
- A stationary point is only necessary for being the minimum.

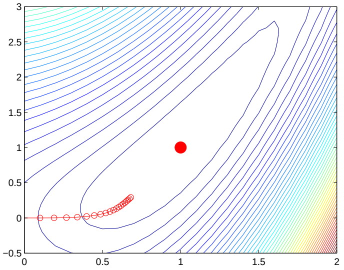
Seeing in action

Choosing the right η is important

Seeing in action

Choosing the right η is important

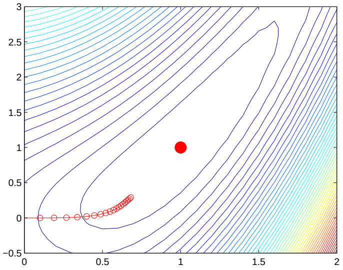
small η is too slow?



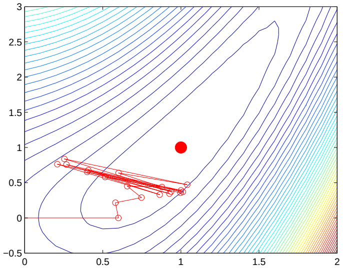
Seeing in action

Choosing the right η is important

small η is too slow?



large η is too unstable?



Gradient Descent Update for Logistic Regression

Simple fact: derivatives of $\sigma(a)$

$$\begin{aligned}
\frac{d\,\sigma(a)}{d\,a} &= \frac{d}{d\,a}\left(1+e^{-a}\right)^{-1} = \frac{-(1+e^{-a})'}{(1+e^{-a})^2} \\
&= \frac{e^{-a}}{(1+e^{-a})^2} = \frac{1}{1+e^{-a}}\frac{e^{-a}}{1+e^{-a}} \\
&= \sigma(a)[1-\sigma(a)]
\end{aligned}$$

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

$$\begin{aligned}\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= -\sum_n \left\{y_n[1-\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\boldsymbol{x}_n - (1-y_n)\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\boldsymbol{x}_n\right\} && (6)\\ &= \sum_n \left\{\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n) - y_n\right\}\boldsymbol{x}_n && (7)\\ &= \sum_n \left\{h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - y_n\right\}\boldsymbol{x}_n && (8)\end{aligned}$$

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

$$\begin{aligned}
\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= -\sum_n \left\{y_n[1 - \sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\boldsymbol{x}_n - (1 - y_n)\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)]\boldsymbol{x}_n\right\} && (6)\\
&= \sum_n \left\{\sigma(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n) - y_n\right\} \boldsymbol{x}_n && (7)\\
&= \sum_n \left\{h_{\boldsymbol{\theta}}(\boldsymbol{x}_n) - y_n\right\} \boldsymbol{x}_n && (8)
\end{aligned}$$

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*

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