

Computer Vision: Logistic regression

Siheng Chen 陈思衡

Logistic regression

Supervised learning: Classification

Application: Image classification

Method: Logistic regression

Optimization

Image classification

ImageNet LSVRC-2010 contest:

- Dataset: 1.2 million labeled images, 1000 classes
- Task: Given a new image, label it with the correct class
- Multiclass classification problem

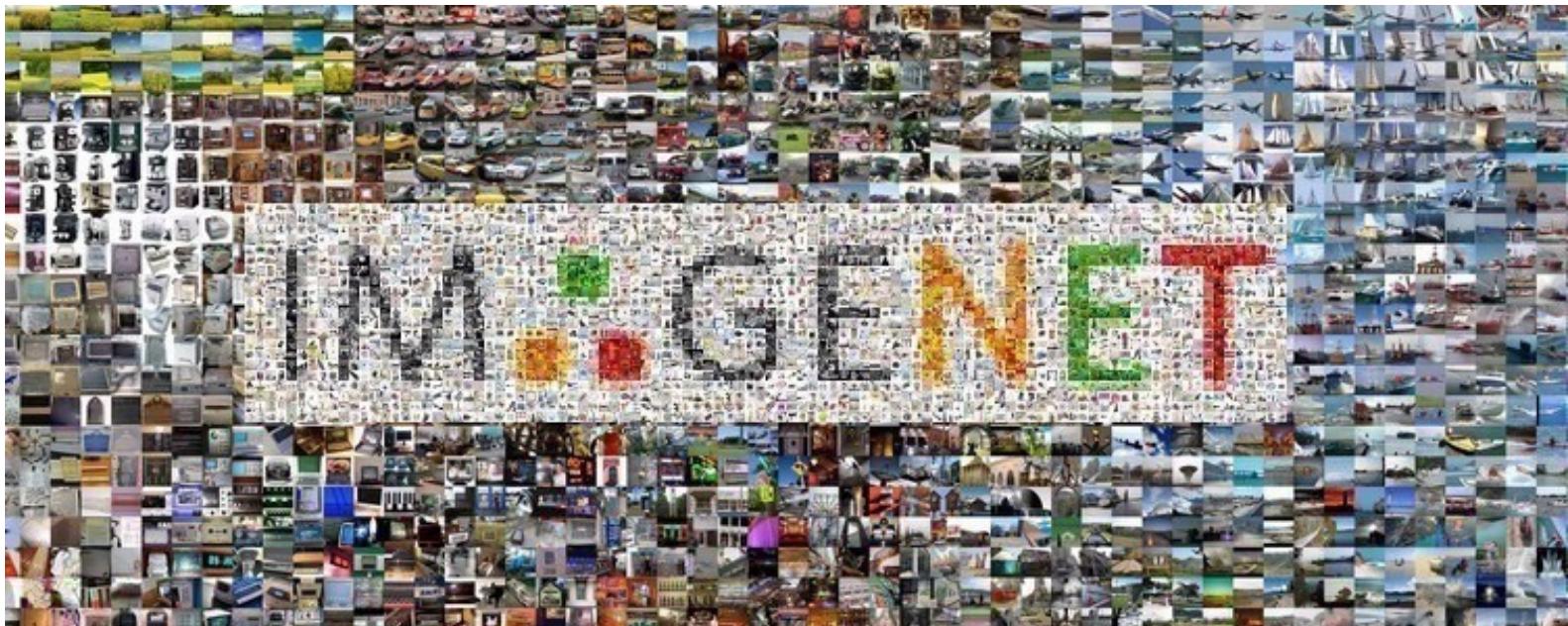


Image classification

IMAGENET

14,197,122 images, 21841 synsets indexed

SEARCH

Home | Explore | Download

Not logged in. [Login](#) | [Signup](#)

Bird

Warm-blooded egg-laying vertebrates characterized by feathers and forelimbs modified as wings

2126 pictures | 92.85% Popularity Percentile | Wordnet IDs

Treemap Visualization | Images of the Synset | Downloads

- marine animal, marine creature, sea animal, sea creature (1)
- scavenger (1)
- biped (0)
- predator, predatory animal (1)
- larva (49)
- acrodont (0)
- feeder (0)
- stunt (0)
- chordate (3087)
 - tunicate, urochordate, urochord (6)
 - cephalochordate (1)
 - vertebrate, craniate (3077)
 - mammal, mammalian (1169)
 - bird (871)
 - dickeybird, dickey-bird, dickybird, dicky-bird (0)
 - cock (1)
 - hen (0)
 - nester (0)
 - night bird (1)
 - bird of passage (0)
 - protoavis (0)
 - archaeopteryx, archeopteryx, Archaeopteryx lithographica, Sinornis (0)
 - Ibero-mesornis (0)
 - archaeornis (0)
 - ratite, ratite bird, flightless bird (10)
 - carinate, carinate bird, flying bird (0)
 - passerine, passeriform bird (279)
 - nonpasserine bird (0)
 - bird of prey, raptor, raptorial bird (80)
 - gallinaceous bird, gallinacean (114)

4

<http://www.cs.cmu.edu/~mgormley/courses/10601/slides/lecture9-logreg.pdf>

Image classification

IMAGENET 14,197,122 images, 21841 synsets indexed SEARCH

Home About Explore Download

Not logged in. [Login](#) | [Signup](#)

German iris, Iris kochii

Iris of northern Italy having deep blue-purple flowers; similar to but smaller than Iris germanica

469 pictures 49.6% Popularity Percentile Wordnet IDs

Treemap Visualization Images of the Synset Downloads

halophyte (0)
succulent (39)
cultivar (0)
cultivated plant (0)
weed (54)
evergreen, evergreen plant (0)
deciduous plant (0)
vine (272)
creeper (0)
woody plant, ligneous plant (1868)
geophyte (0)
desert plant, xerophyte, xerophytic plant, xerophile, xerophilic
mesophyte, mesophytic plant (0)
aquatic plant, water plant, hydrophyte, hydrophytic plant (11)
tuberous plant (0)
bulbous plant (179)
 ↳ iridaceous plant (27)
 ↳ iris, flag, fleur-de-lis, sword lily (19)
 ↳ bearded iris (4)
 ↳ Florentine iris, orris, Iris germanica florentina, Iris
 ↳ German iris, Iris germanica (0)
 ↳ German iris, Iris kochii (0)
 ↳ Dalmatian iris, Iris pallida (0)
 ↳ beardless iris (4)
 ↳ bulbous iris (0)
 ↳ dwarf iris, Iris cristata (0)
 ↳ stinking iris, gladdon, gladdon iris, stinking gladwyn,
 ↳ Persian iris, Iris persica (0)
 ↳ yellow iris, yellow flag, yellow water flag, Iris pseudacorus
 ↳ dwarf iris, vernal iris, Iris verna (0)
 ↳ blue flag, Iris versicolor (0)

<http://www.cs.cmu.edu/~mgormley/courses/10601/slides/lecture9-logreg.pdf>

Image classification

CNN for Image Classification

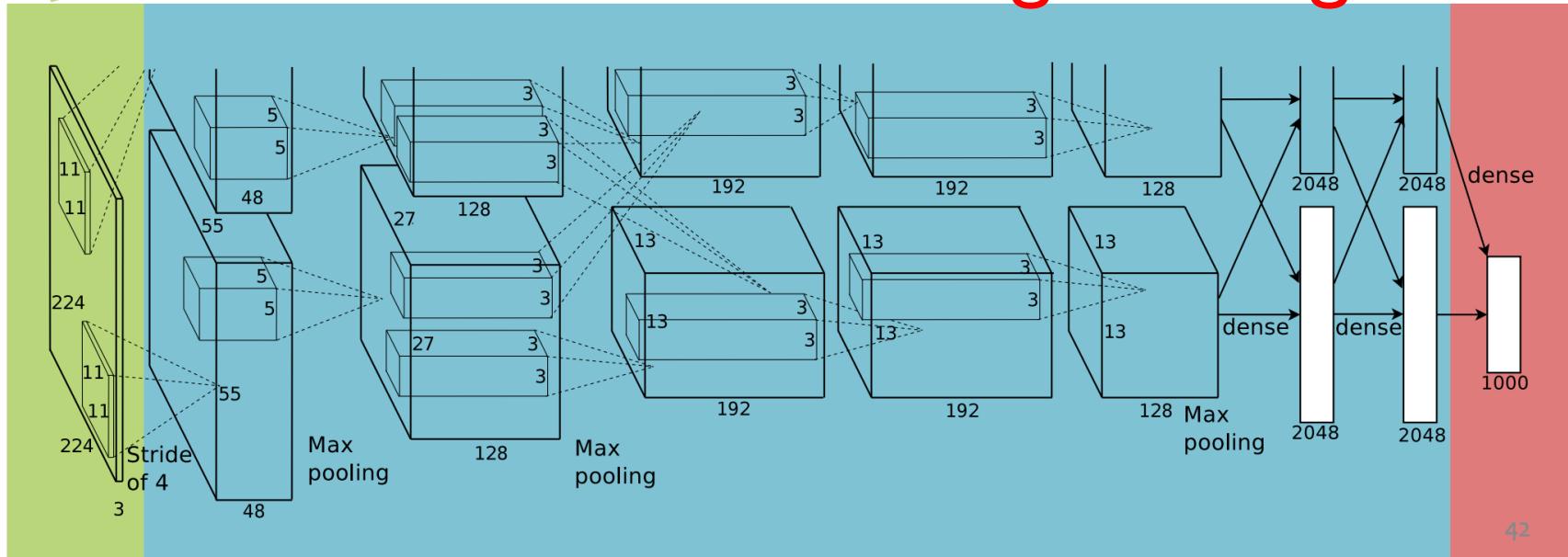
(Krizhevsky, Sutskever & Hinton, 2011)
17.5% error on ImageNet LSVRC-2010 contest

Input
image
(pixels)

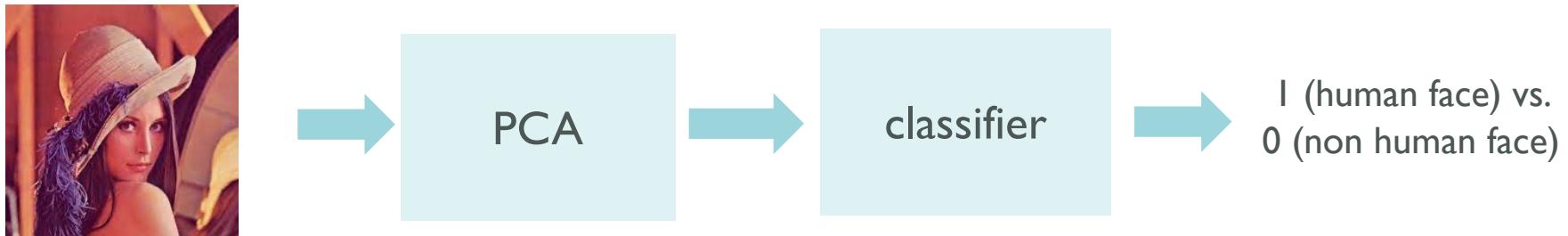
- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way
softmax

Logistic Regression!



Binary classification



$$\mathbf{I} \in \mathbb{R}^{H \times W}$$

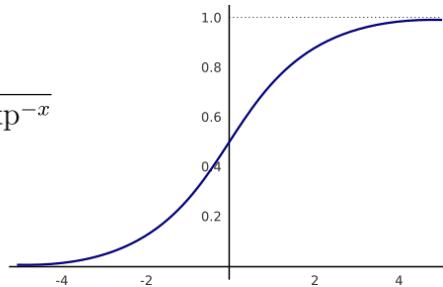
$$\mathbf{x} \in \mathbb{R}^d$$

$$y \in \{0, 1\}$$

classifier: $\hat{y} = P(y = 1 | \mathbf{x})$

logistic regression: $\hat{y} = \sigma(\mathbf{w}^T \mathbf{x} + b)$

$$\sigma(x) = \frac{1}{1 + \exp^{-x}}$$



trainable parameters

Logistic regression

m training examples: $\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\}$

model: $\hat{y} = \sigma(\mathbf{w}^T \mathbf{x} + b)$

loss function (cross entropy): $L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$

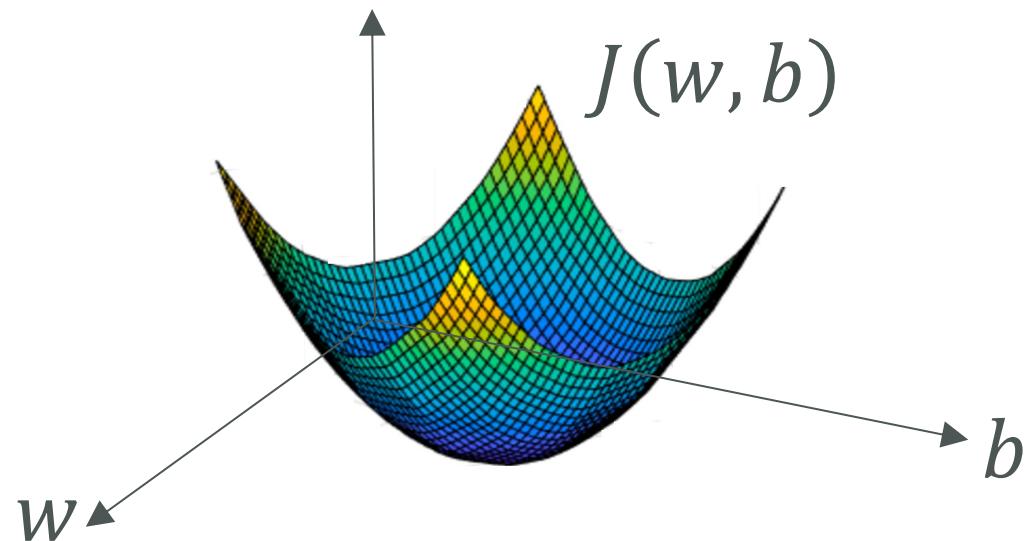
cost function: $J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)})$

$$= -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

Logistic regression

want to find w, b that minimize $J(w, b)$

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

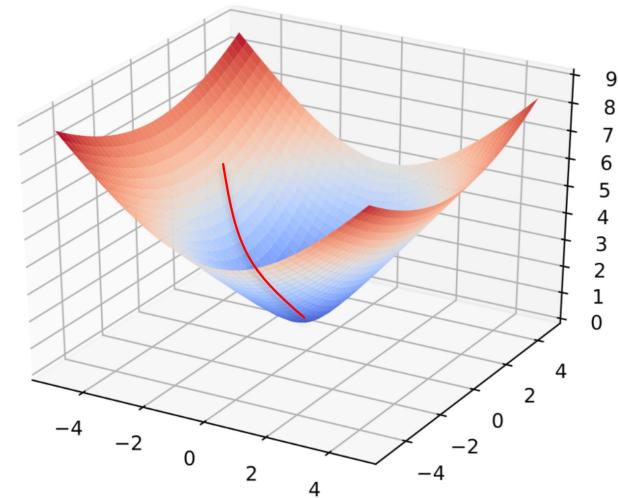


Logistic regression

gradient descent

Algorithm 1 Gradient Descent

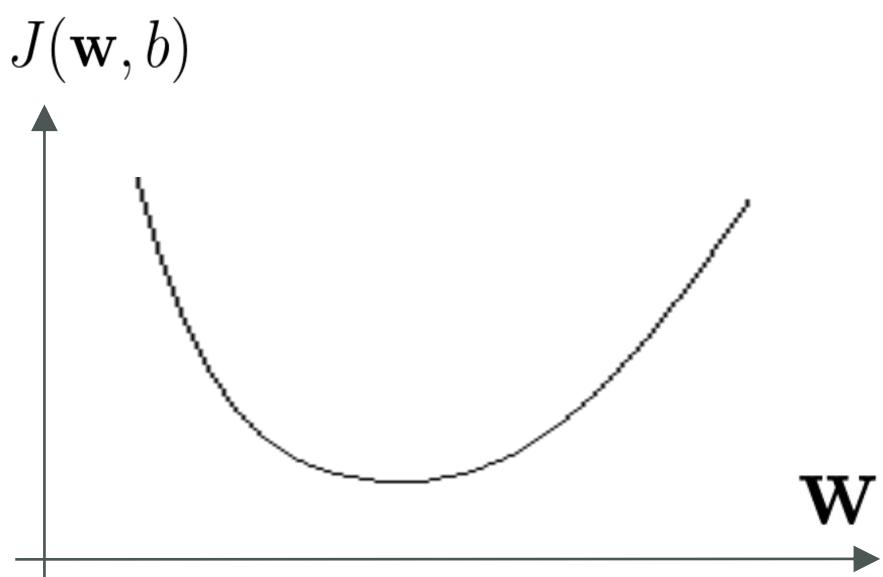
```
1: procedure GD( $\mathcal{D}$ ,  $\theta^{(0)}$ )
2:    $\theta \leftarrow \theta^{(0)}$ 
3:   while not converged do
4:      $\theta \leftarrow \theta - \gamma \nabla_{\theta} J(\theta)$ 
5:   return  $\theta$ 
```



Logistic regression

want to find w, b that minimize $J(w, b)$

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$



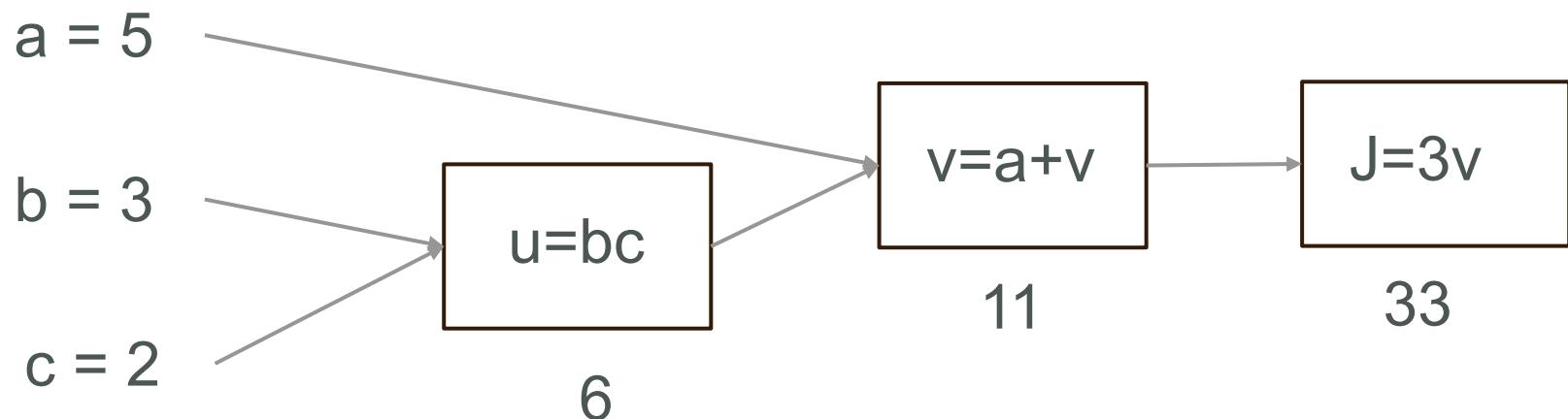
$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial J}{\partial \mathbf{w}}$$

$$b \leftarrow b - \alpha \frac{\partial J}{\partial b}$$

Logistic regression

Computational graph

$$J(a, b, c) = 3(a + bc) = 3 * (5 + 3 * 2) = 33$$



Logistic regression

Computing derivatives

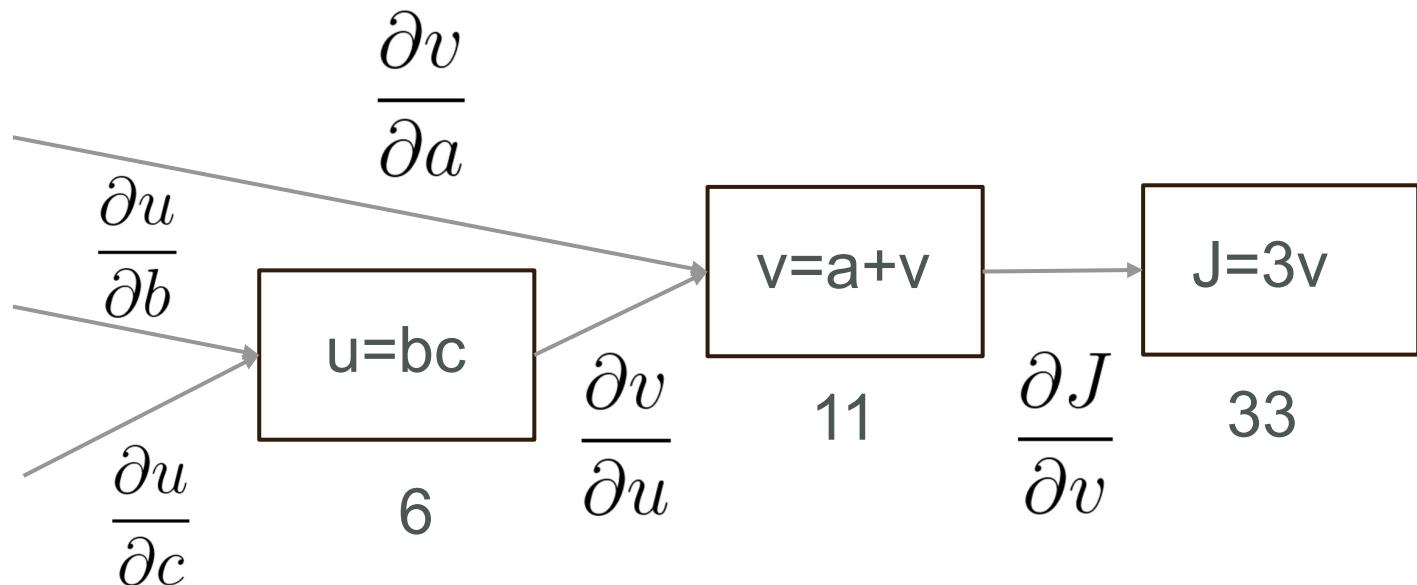
$$J(a, b, c) = 3(a + bc) = 3 * (5 + 3 * 2) = 33$$

$$\frac{\partial J}{\partial a} \quad \text{How would } a \text{'s change effect } J?$$

$$\frac{\partial J}{\partial a} = \frac{\partial J}{\partial v} \frac{\partial v}{\partial a} \quad a = 5$$

$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial v} \frac{\partial v}{\partial u} \frac{\partial u}{\partial b} \quad b = 3$$

$$\frac{\partial J}{\partial c} = \frac{\partial J}{\partial v} \frac{\partial v}{\partial u} \frac{\partial u}{\partial c} \quad c = 2$$



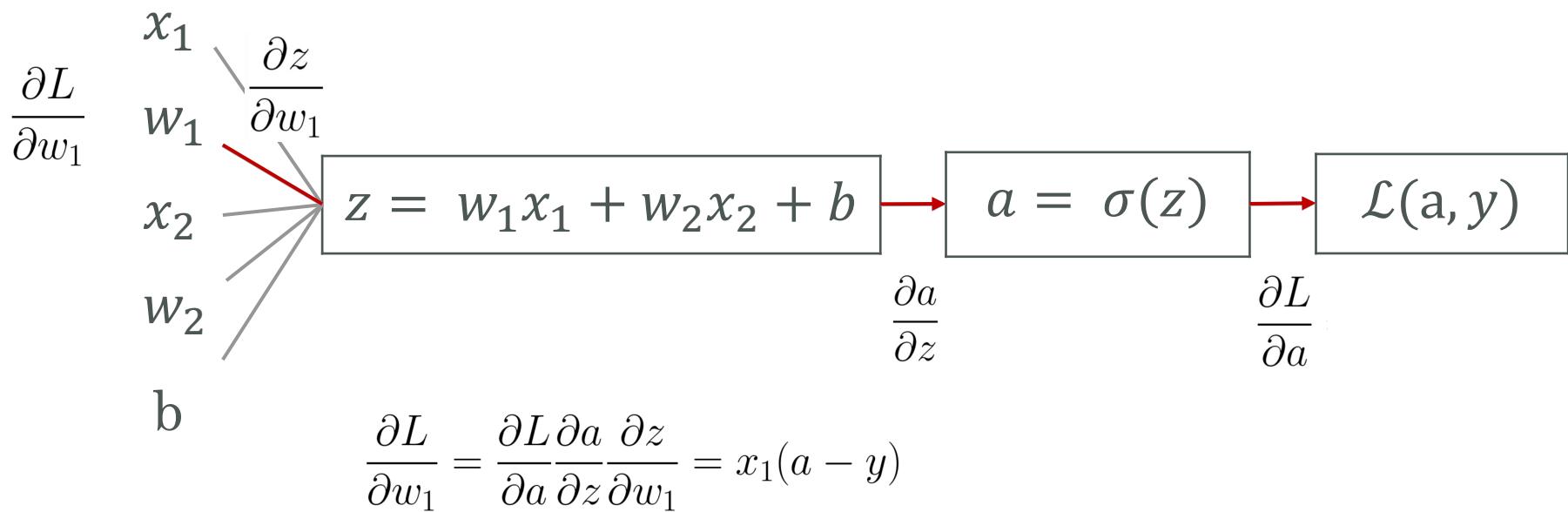
Logistic regression

A single sample

$$z = w^T x + b$$

$$\hat{y} = a = \sigma(z)$$

$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$



Logistic regression

m training samples

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^m L(a^{(i)}, y^{(i)}) \quad a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(\mathbf{w}^T \mathbf{x}^{(i)} + b)$$

Partial derivative computation

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{w}} &= \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial \mathbf{w}} L(a^{(i)}, y^{(i)}) \\ &= \frac{1}{m} \sum_{i=1}^m (a^{(i)} - y^{(i)}) \mathbf{x}^{(i)}\end{aligned}$$

$$\begin{aligned}\frac{\partial J}{\partial b} &= \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial b} L(a^{(i)}, y^{(i)}) \\ &= \frac{1}{m} \sum_{i=1}^m (a^{(i)} - y^{(i)})\end{aligned}$$

update

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial J}{\partial \mathbf{w}}$$

$$b \leftarrow b - \alpha \frac{\partial J}{\partial b}$$

Logistic regression

Vectorization

$$z^{(1)} = w^T x^{(1)} + b \quad z^{(2)} = w^T x^{(2)} + b \quad z^{(3)} = w^T x^{(3)} + b$$

$$a^{(1)} = \sigma(z^{(1)}) \quad a^{(2)} = \sigma(z^{(2)}) \quad a^{(3)} = \sigma(z^{(3)})$$

$$\mathbf{z} = \begin{bmatrix} z^{(1)} \\ z^{(2)} \\ \vdots \\ z^{(m)} \end{bmatrix} = \begin{bmatrix} \mathbf{w}^T \mathbf{x}^{(1)} + b \\ \mathbf{w}^T \mathbf{x}^{(2)} + b \\ \vdots \\ \mathbf{w}^T \mathbf{x}^{(m)} + b \end{bmatrix}$$

data matrix: $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}] \in \mathbb{R}^{d \times m}$

`z = np.dot(X.T, w) + b`

`a = 1 / (1 + np.exp(-x))`

Logistic regression

Partial derivative computation

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{w}} &= \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial \mathbf{w}} L(a^{(i)}, y^{(i)}) \\ &= \frac{1}{m} \sum_{i=1}^m (a^{(i)} - y^{(i)}) \mathbf{x}^{(i)}\end{aligned}$$

$$\begin{aligned}\frac{\partial J}{\partial b} &= \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial b} L(a^{(i)}, y^{(i)}) \\ &= \frac{1}{m} \sum_{i=1}^m (a^{(i)} - y^{(i)})\end{aligned}$$



vectorization

$$\frac{1}{m} \sum_{i=1}^m (a^{(i)} - y^{(i)}) \mathbf{x}^{(i)} = \frac{1}{m} [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}] \begin{bmatrix} a^{(1)} - y^{(1)} \\ a^{(2)} - y^{(2)} \\ \vdots \\ a^{(m)} - y^{(m)} \end{bmatrix}$$

Logistic regression

update

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial J}{\partial \mathbf{w}}$$

$$b \leftarrow b - \alpha \frac{\partial J}{\partial b}$$



vectorization

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{m} \mathbf{X}(\mathbf{a} - \mathbf{y})$$

$$b \leftarrow b - \frac{\alpha}{m} \mathbf{1}_m^T (\mathbf{a} - \mathbf{y})$$

Logistic regression

data matrix: $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}] \in \mathbb{R}^{d \times m}$

label matrix: $\mathbf{Y} = [y^{(1)}, y^{(2)}, \dots, y^{(m)}] \in \mathbb{R}^{1 \times m}$

model: $\hat{y} = \sigma(\mathbf{w}^T \mathbf{x} + b)$

loss function (cross entropy): $L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$

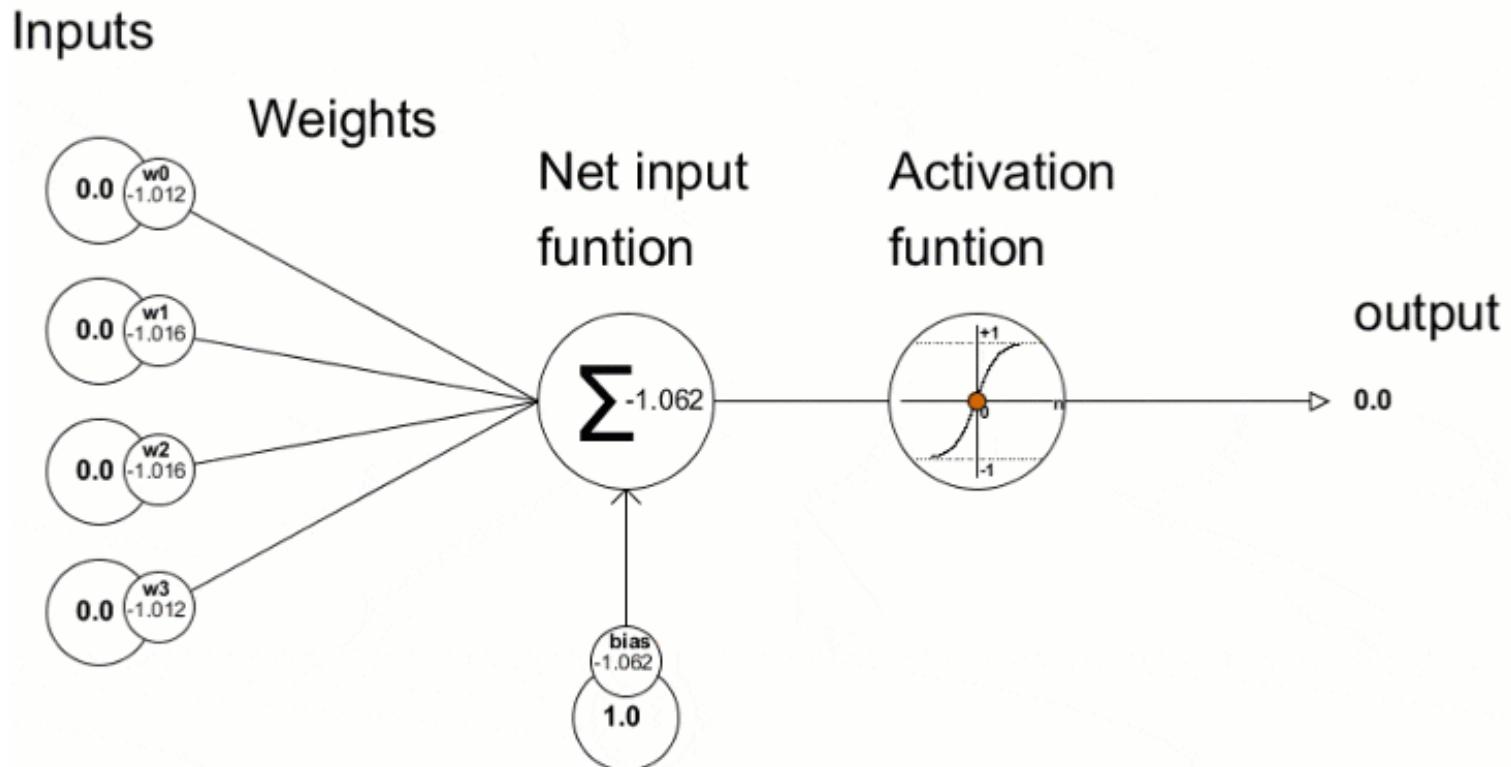
minimize cost function: $J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)})$

iterative solver: $\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{m} \mathbf{X}(\mathbf{a} - \mathbf{y})$

$b \leftarrow b - \frac{\alpha}{m} \mathbf{1}_m^T (\mathbf{a} - \mathbf{y})$

Logistic regression

Training



Logistic regression

data matrix: $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}] \in \mathbb{R}^{d \times m}$

label matrix: $\mathbf{Y} = [y^{(1)}, y^{(2)}, \dots, y^{(m)}] \in \mathbb{R}^{1 \times m}$

model: $\hat{y} = \sigma(\mathbf{w}^T \mathbf{x} + b)$

$$\frac{e^{z_k}}{\sum_j e^{z_j}}$$

loss function (cross entropy): $L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$

$$-\sum_{c=1}^C y_c \log \hat{y}_c$$

minimize cost function: $J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)})$

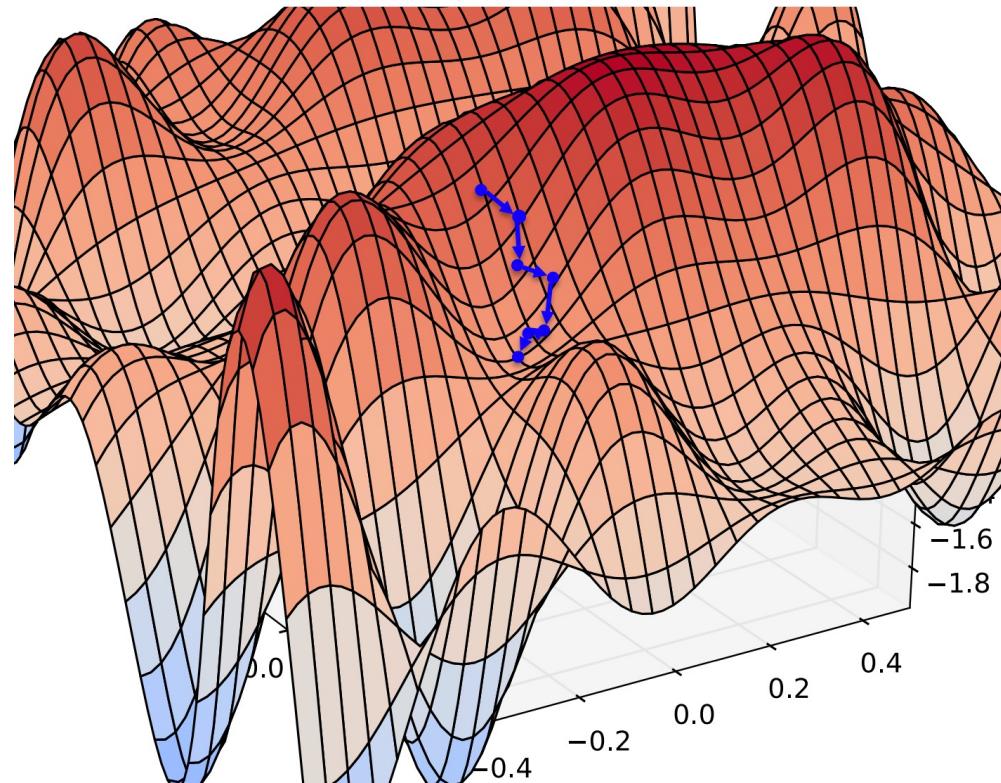
iterative solver: $\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{m} \mathbf{X}(\mathbf{a} - \mathbf{y})$

$b \leftarrow b - \frac{\alpha}{m} \mathbf{1}_m^T (\mathbf{a} - \mathbf{y})$

Binary classification  multi-class classification

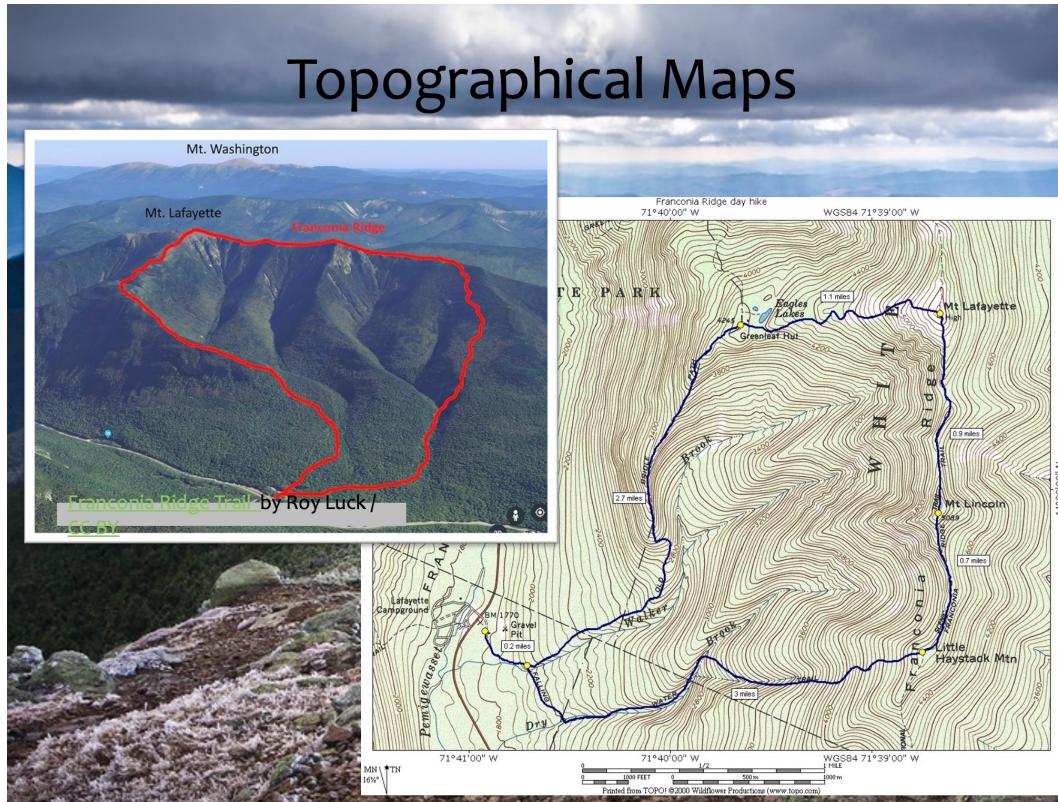
More about optimization

$$\text{minimize} \quad J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)})$$



More about optimization

$$\text{minimize} \quad J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)})$$

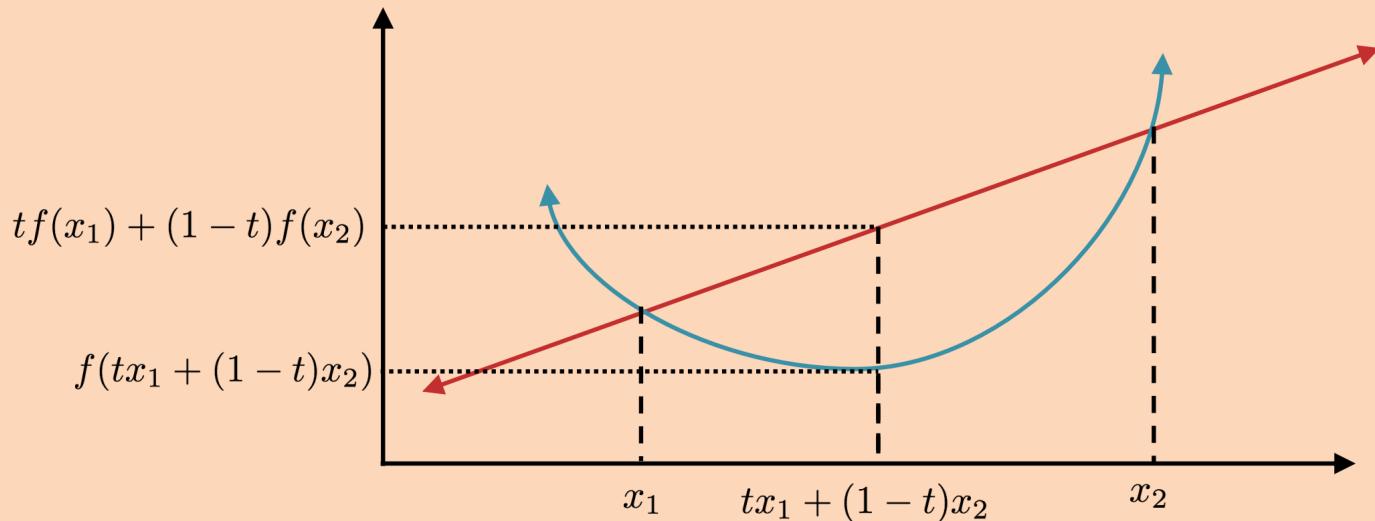


More about optimization

Function $f : \mathbb{R}^M \rightarrow \mathbb{R}$ is **convex**

if $\forall \mathbf{x}_1 \in \mathbb{R}^M, \mathbf{x}_2 \in \mathbb{R}^M, 0 \leq t \leq 1$:

$$f(t\mathbf{x}_1 + (1 - t)\mathbf{x}_2) \leq tf(\mathbf{x}_1) + (1 - t)f(\mathbf{x}_2)$$

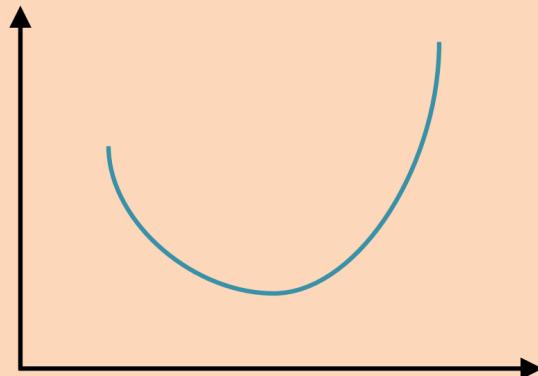


More about optimization

Suppose we have a function $f(x) : \mathcal{X} \rightarrow \mathcal{Y}$.

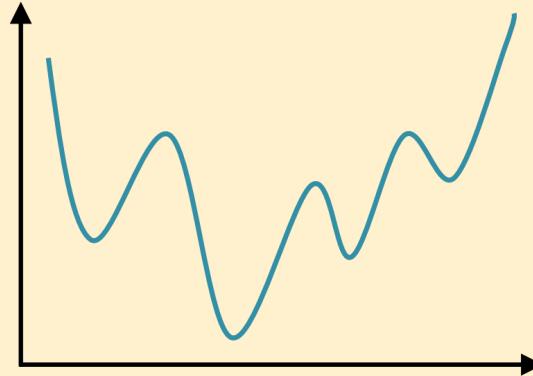
- The value x^* is a **global minimum** of f iff $f(x^*) \leq f(x), \forall x \in \mathcal{X}$.
- The value x^* is a **local minimum** of f iff $\exists \epsilon$ s.t. $f(x^*) \leq f(x), \forall x \in [x^* - \epsilon, x^* + \epsilon]$.

Convex Function



- Each **local minimum** is a **global minimum**

Nonconvex Function

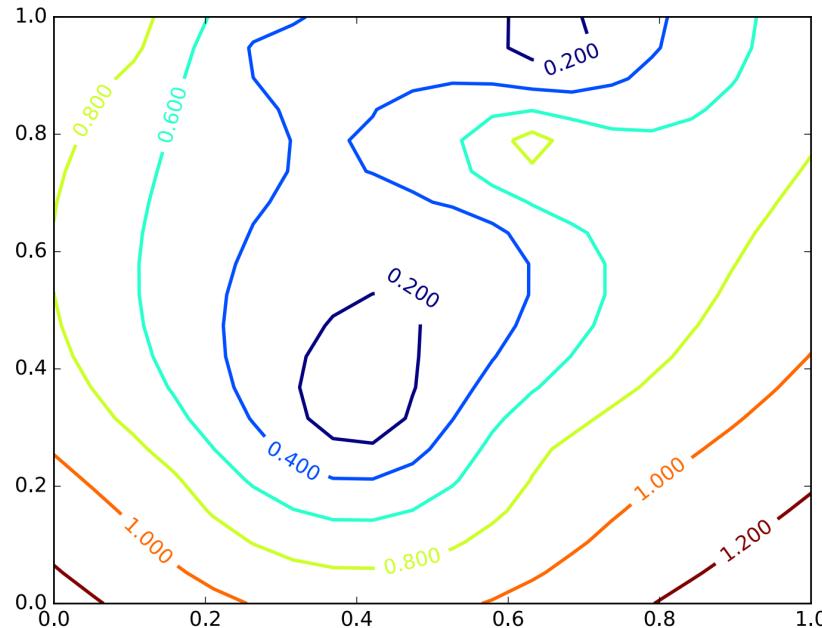


- A nonconvex function is **not convex**
- Each **local minimum** is **not necessarily a global minimum**

More about optimization

To solve an optimization problem

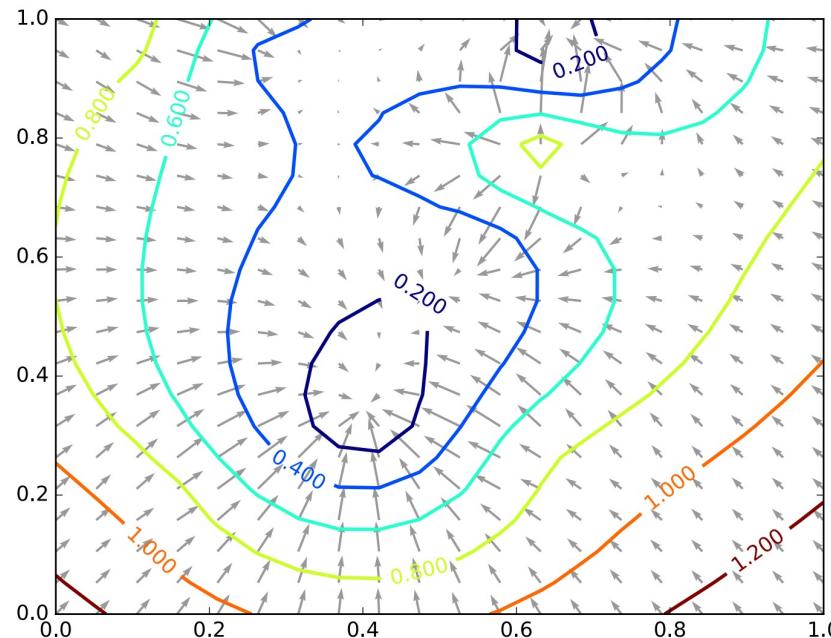
- Closed-form solution (PCA)
- Iterative solver (Logistic regression)
 - Gradient descent



More about optimization

To solve an optimization problem

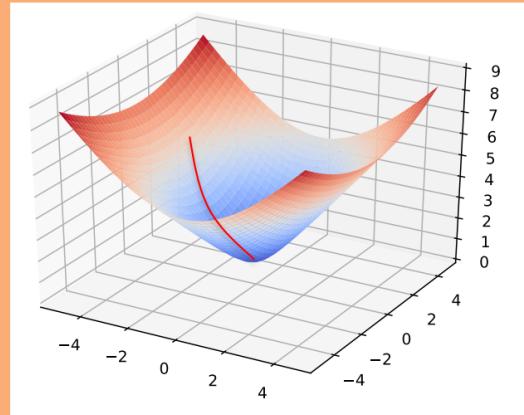
- Closed-form solution (PCA)
- Iterative solver (Logistic regression)
 - Gradient descent



More about optimization

Algorithm 1 Gradient Descent

```
1: procedure GD( $\mathcal{D}$ ,  $\theta^{(0)}$ )
2:    $\theta \leftarrow \theta^{(0)}$ 
3:   while not converged do
4:      $\theta \leftarrow \theta - \gamma \nabla_{\theta} J(\theta)$ 
5:   return  $\theta$ 
```



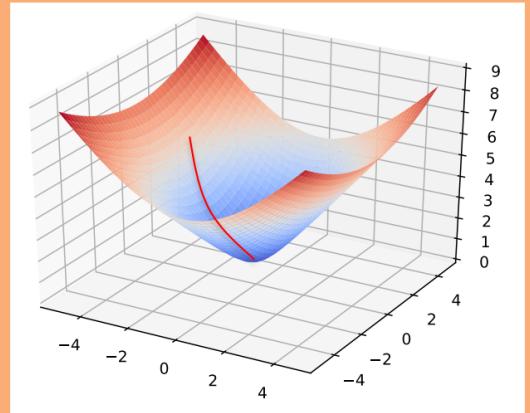
In order to apply GD to Linear Regression all we need is the **gradient** of the objective function (i.e. vector of partial derivatives).

$$\nabla_{\theta} J(\theta) = \begin{bmatrix} \frac{d}{d\theta_1} J(\theta) \\ \frac{d}{d\theta_2} J(\theta) \\ \vdots \\ \frac{d}{d\theta_M} J(\theta) \end{bmatrix}$$

More about optimization

Algorithm 1 Gradient Descent

```
1: procedure GD( $\mathcal{D}$ ,  $\theta^{(0)}$ )
2:    $\theta \leftarrow \theta^{(0)}$ 
3:   while not converged do
4:      $\theta \leftarrow \theta - \gamma \nabla_{\theta} J(\theta)$ 
5:   return  $\theta$ 
```



There are many possible ways to detect **convergence**.

For example, we could check whether the L2 norm of the gradient is below some small tolerance.

$$\|\nabla_{\theta} J(\theta)\|_2 \leq \epsilon$$

Alternatively we could check that the reduction in the objective function from one iteration to the next is small.

More about optimization

Linear regression

1. Assume \mathcal{D} generated as:

$$\begin{aligned}\mathbf{x}^{(i)} &\sim p^*(\cdot) \\ y^{(i)} &= h^*(\mathbf{x}^{(i)})\end{aligned}$$

2. Choose hypothesis space, \mathcal{H} :
all linear functions in M -dimensional space

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}, \boldsymbol{\theta} \in \mathbb{R}^M\}$$

3. Choose an objective function:
mean squared error (MSE)

$$\begin{aligned}J(\boldsymbol{\theta}) &= \frac{1}{N} \sum_{i=1}^N e_i^2 \\ &= \frac{1}{N} \sum_{i=1}^N \left(y^{(i)} - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \right)^2 \\ &= \frac{1}{N} \sum_{i=1}^N \left(y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)} \right)^2\end{aligned}$$

4. Solve the unconstrained optimization problem via favorite method:

- gradient descent
- closed form
- stochastic gradient descent
- ...

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} J(\boldsymbol{\theta})$$

5. Test time: given a new \mathbf{x} , make prediction \hat{y}

$$\hat{y} = h_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) = \hat{\boldsymbol{\theta}}^T \mathbf{x}$$

More about optimization

Linear regression

To solve the Ordinary Least Squares problem we compute:

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} = \frac{1}{N} \sum_{i=1}^N \frac{1}{2}(y^{(i)} - (\boldsymbol{\theta}^T \mathbf{x}^{(i)}))^2 \\ &= (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{Y})\end{aligned}$$

Closed-form solution

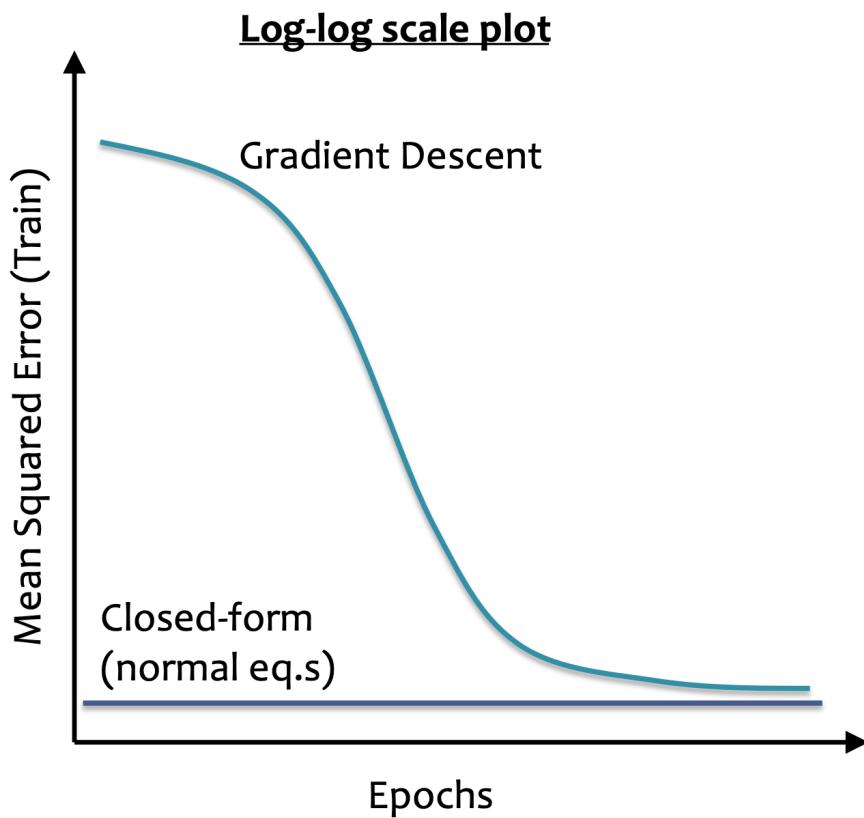
Algorithm 1 GD for Linear Regression

```
1: procedure GDLR( $\mathcal{D}, \boldsymbol{\theta}^{(0)}$ )  
2:    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$                                  $\triangleright$  Initialize parameters  
3:   while not converged do  
4:      $\mathbf{g} \leftarrow \sum_{i=1}^N (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$        $\triangleright$  Compute gradient  
5:      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \mathbf{g}$                        $\triangleright$  Update parameters  
6:   return  $\boldsymbol{\theta}$ 
```

Iterative solver by gradient descent

More about optimization

Linear regression



- Def: an **epoch** is a single pass through the training data
 1. For GD, only **one update** per epoch
 2. For SGD, **N updates** per epoch
 $N = (\# \text{ train examples})$

- SGD reduces MSE much more rapidly than GD
- For GD / SGD, training MSE is initially large due to uninformed initialization

More about optimization

Linear regression

To solve the Ordinary Least Squares problem we compute:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} = \frac{1}{N} \sum_{i=1}^N \frac{1}{2} (y^{(i)} - (\theta^T \mathbf{x}^{(i)}))^2 \\ = (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{Y})$$

Closed-form solution



High computational cost!

Computational Complexity of OLS:

$\mathbf{X}^T \mathbf{X}$	$O(M^2 N)$
$(\quad)^{-1}$	$O(M^{2.373})$
$\mathbf{X}^T \mathbf{Y}$	$O(MN)$
$(\quad)^{-1} (\quad)$	$O(M^2)$
total	$O(M^2 N + M^{2.373})$

Algorithm 1 GD for Linear Regression

```
1: procedure GDLR( $\mathcal{D}, \theta^{(0)}$ )
2:    $\theta \leftarrow \theta^{(0)}$                                  $\triangleright$  Initialize parameters
3:   while not converged do
4:      $\mathbf{g} \leftarrow \sum_{i=1}^N (\theta^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$        $\triangleright$  Compute gradient
5:      $\theta \leftarrow \theta - \gamma \mathbf{g}$                                  $\triangleright$  Update parameters
6:   return  $\theta$ 
```

Iterative solver by gradient descent



determined the number of iterations

Thank you very much!

sihengc@sjtu.edu.cn