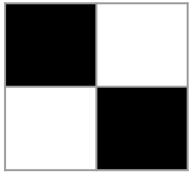
Gradient Descent

Boston University CS 506 - Lance Galletti

Logistic Regression Revisited

Given a 2 x 2 grid where each cell a_{ij} can take on one of two colors c_1 and c_2, find a function that can identify the following diagonal pattern:



■ = c_1 = -1

□ = c_2 = 1

Logistic Regression Revisited

That is, find **f** such that

$$\mathbf{f}\left(\begin{array}{|c|c|}\hline a_{00} & a_{01} \\ \hline a_{10} & a_{11} \\ \hline\end{array}\right) = \begin{cases} \text{✔ if } \begin{array}{|c|c|}\hline a_{00} & a_{01} \\ \hline a_{10} & a_{11} \\ \hline\end{array} = \begin{array}{|c|c|}\hline c_1 & c_2 \\ \hline c_2 & c_1 \\ \hline\end{array} = \begin{array}{|c|c|}\hline \blacksquare & \square \\ \hline \square & \blacksquare \\ \hline\end{array} \\ \text{✘ otherwise} \end{cases}$$

We can define: ✔ = 1 and ✘ = 0

Logistic Regression Revisited

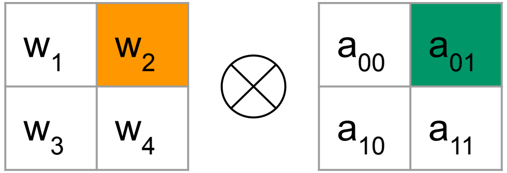
We can assign weights to each cell

w_1	w_2
w_3	w_4

Logistic Regression Revisited

$$\begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} \otimes \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} = w_1 a_{00}$$

Logistic Regression Revisited



$$\begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} \otimes \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} = w_1 a_{00} + w_2 a_{01}$$

Logistic Regression Revisited

$$\begin{array}{|c|c|} \hline w_1 & w_2 \\ \hline w_3 & w_4 \\ \hline \end{array} \otimes \begin{array}{|c|c|} \hline a_{00} & a_{01} \\ \hline a_{10} & a_{11} \\ \hline \end{array} = w_1a_{00} + w_2a_{01} + w_3a_{10}$$

Logistic Regression Revisited

$$\begin{array}{|c|c|} \hline w_1 & w_2 \\ \hline w_3 & w_4 \\ \hline \end{array} \otimes \begin{array}{|c|c|} \hline a_{00} & a_{01} \\ \hline a_{10} & a_{11} \\ \hline \end{array} = w_1a_{00} + w_2a_{01} + w_3a_{10} + w_4a_{11}$$

Logistic Regression Revisited

We can assign weights to each cell

w_1	w_2
w_3	w_4

such that:

$$\underbrace{w_1a_{00} + w_2a_{01} + w_3a_{10} + w_4a_{11}}_{\begin{array}{|c|c|}\hline w_1 & w_2 \\ \hline w_3 & w_4 \\ \hline\end{array} \otimes \begin{array}{|c|c|}\hline a_{00} & a_{01} \\ \hline a_{10} & a_{11} \\ \hline\end{array}} = b \quad \text{if diagonal pattern found}$$

Logistic Regression Revisited

For example:

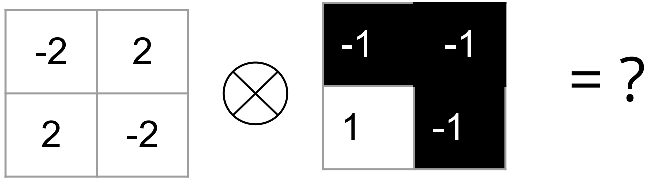
$$\begin{array}{|c|c|}\hline w_1 & w_2 \\ \hline w_3 & w_4 \\ \hline\end{array} = \begin{array}{|c|c|}\hline -2 & 2 \\ \hline 2 & -2 \\ \hline\end{array}$$

What value b do we get when applied to the diagonal pattern?

$$\begin{array}{|c|c|}\hline -2 & 2 \\ \hline 2 & -2 \\ \hline\end{array} \otimes \begin{array}{|c|c|}\hline -1 & 1 \\ \hline 1 & -1 \\ \hline\end{array} = ?$$

Logistic Regression Revisited

Any other pattern will have a value lower:



$$\begin{bmatrix}-2 & 2\\ 2 & -2\end{bmatrix} \otimes \begin{bmatrix}-1 & -1\\ 1 & -1\end{bmatrix} = ?$$

Logistic Regression Revisited

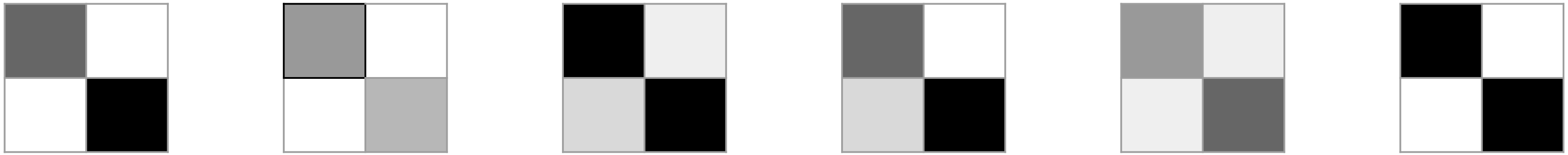
Equivalently we can decide to move the value b to the left of the equation in order for the weighted sum to reveal a diagonal pattern at 0:

$$w_1 a_{00} + w_2 a_{01} + w_3 a_{10} + w_4 a_{11} + b = 0 \quad \text{if diagonal pattern found}$$

Logistic Regression Revisited

Suppose we relax our definition of diagonal by having a continuum of colors $[c_1, c_2]$. This means there will be a continuum of values for our weighted sum to take when a diagonal pattern is found:

$$w_1 a_{00} + w_2 a_{01} + w_3 a_{10} + w_4 a_{11} + b > 0 \quad \text{if diagonal}$$



Logistic Regression Revisited

We could then find a function **σ** to apply to the result of this sum in order to get probabilities of being diagonal:

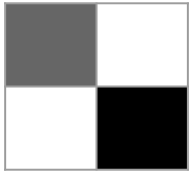
$\boldsymbol{\sigma}(w_1a_{00} + w_2a_{01} + w_3a_{10} + w_4a_{11} + b) > ½$ if $w_1a_{00} + w_2a_{01} + w_3a_{10} + w_4a_{11} + b > 0$

Logistic Regression Revisited

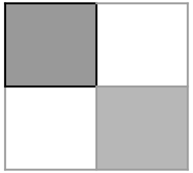
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

When **σ** is the logit^{-1} (also called sigmoid) function, this is Logistic Regression.

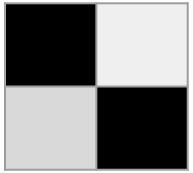
So for each cell we're looking to learn a weight w_i that makes **σ** larger for diagonal patterns. The bias term b lets us account for systemic dimming or brightening of cells (i.e. when the data is not normalized).



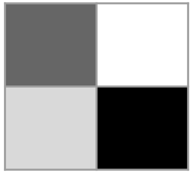
.92



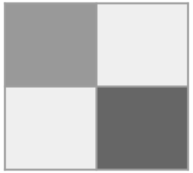
.81



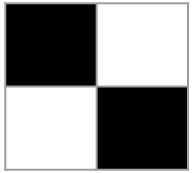
.95



.73

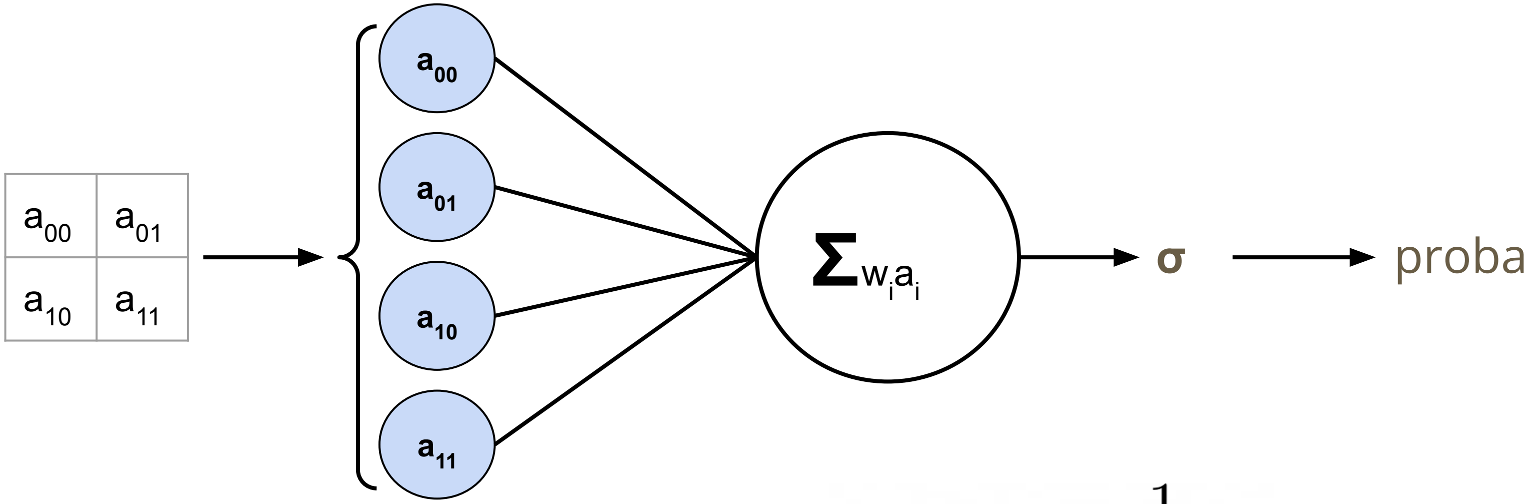


.68



.99

Logistic Regression Revisited



a_{00}	a_{01}
a_{10}	a_{11}

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

Logistic Regression Revisited

Recall that logistic regression is looking for weights and a bias that maximizes the probability of having seen the data we saw:

$$\max \prod_{i=1}^{n} P(y_i|x_i) = \prod_{i} (logit^{-1}(w^T x_i + b))^{y_i} (1 - logit^{-1}(w^T x_i + b))^{1-y_i}$$

$$= \min -\frac{1}{n} \sum_{i=1}^{n} [yi \log(\sigma(-w^T x_i + b)) + (1 - y_i) \log(1 - \sigma(-w^T x_i + b))]$$

$$= \min \mathrm{Cost}(w, b)$$

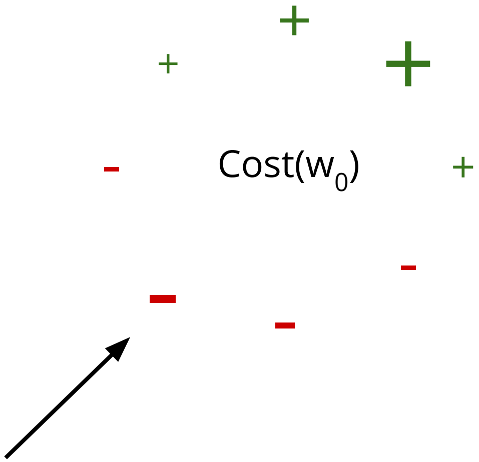
Gradient Descent (intuition)

There is no closed form solution to finding the extrema of this cost function. We can however use an iterative process by which we increment w and b gradually toward some minimum (most likely local).

Goal: find a sequence of w_i’s (and b’s) that converge toward a minimum.

Gradient Descent (intuition)

Consider a random weight w_0. What happens to Cost(w_0) as you nudge w_0 slightly?



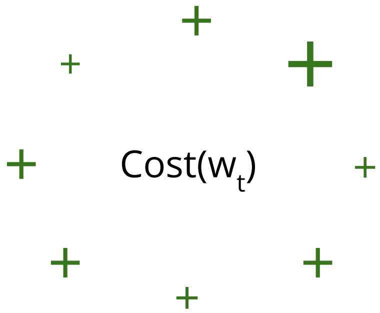
Clearly this is the best nudge to give w_0 to improve our Cost

Gradient Descent (intuition)

As such we can define the following sequence:

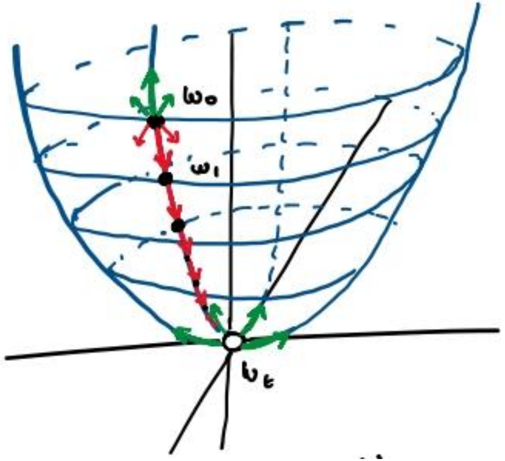
w_1 = best nudge to w_0
w_2 = best nudge to w_1
…

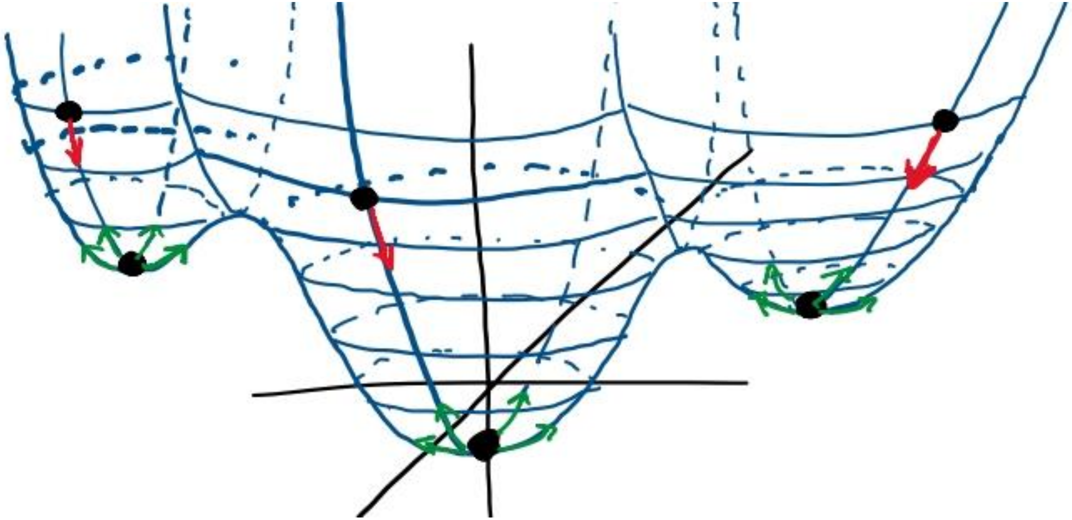
Until we reach w_t that looks like this:



At this point we can stop updating w. Why?

Gradient Descent (intuition)





Gradient Descent (intuition)

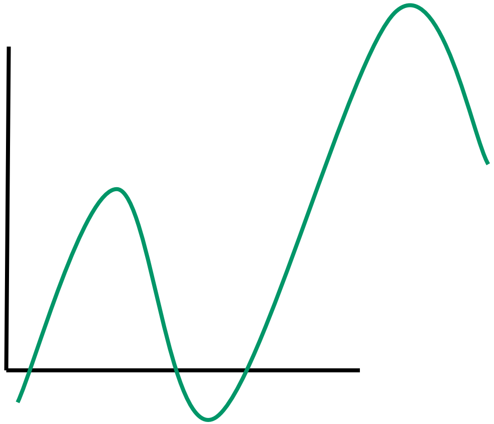
How can we know how much to nudge and in what direction?

Gradients

Intuitively the best nudge should be in the direction of the largest rate of change (steepness) of the function.

Rate of change -> think derivatives

$$\nabla f(x) = f'(x)$$

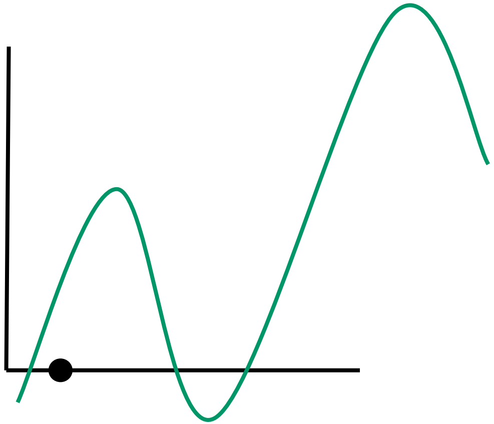


Gradients

Intuitively the best nudge should be in the direction of the largest rate of change (steepness) of the function.

Rate of change -> think derivatives

$$\nabla f(x) = f'(x)$$

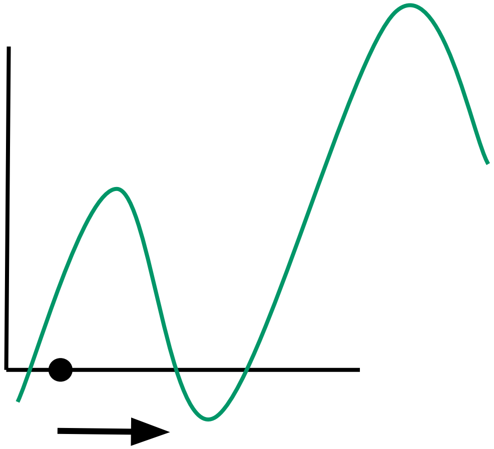


Gradients

Intuitively the best nudge should be in the direction of the largest rate of change (steepness) of the function.

Rate of change -> think derivatives

$$\nabla f(x) = f'(x)$$

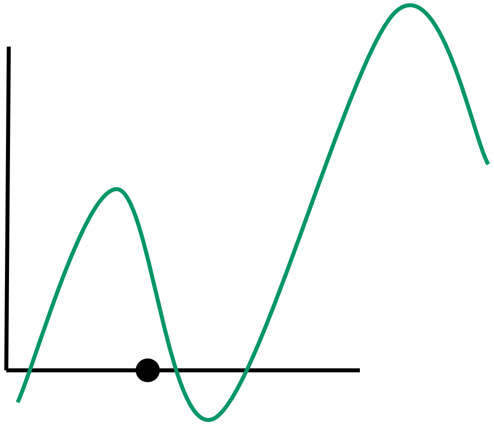


Gradients

Intuitively the best nudge should be in the direction of the largest rate of change (steepness) of the function.

Rate of change -> think derivatives

$$\nabla f(x) = f'(x)$$

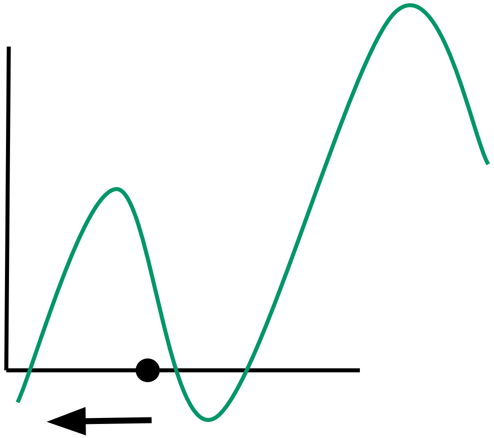


Gradients

Intuitively the best nudge should be in the direction of the largest rate of change (steepness) of the function.

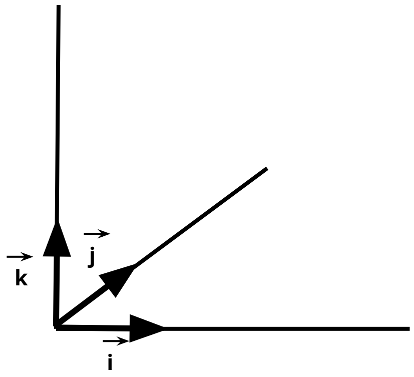
Rate of change -> think derivatives

$$\nabla f(x) = f'(x)$$



Gradients

Intuitively, the rate of change of a multi-dimensional function should be a combination of the rate change in each dimension. For a 3-dimensional function, the rate of change would be:



$$\nabla f(x,y,z) = \frac{\partial f}{\partial x}\vec{i} + \frac{\partial f}{\partial y}\vec{j} + \frac{\partial f}{\partial z}\vec{k}$$

Gradients

Example:

$$f(x) = 3x^2-2y$$

Without even computing derivatives we can see that changes in x create more positive change in f than changes in y.

$$\nabla f = 6xi - 2j$$

This is the gradient of f and can be evaluated at any point (x, y) in the space.

Gradients

$$f(x) = 3x^2\text{-}2y \quad , \quad \nabla f = 6xi - 2j$$

Evaluating ∇f at p=(0, 0):

$$\nabla f_p = 6\cdot0\cdot i - 2j = -2j$$

What happens to f as we move 1 unit away from p in the direction of the gradient?

$$p_{new} = 1\cdot \nabla f_p + p = (0, -2)$$

$$f(p_{new}) = 3\cdot0^2 - 2\cdot (-2) = 4 > f(p) = 0$$

Gradients

$$f(x) = 3x^2\text{-}2y \quad , \quad \nabla f = 6xi - 2j$$

What happens to f if we move 1 unit away from p in a random direction (not following ∇f)? Say (1,0) = 1i + 0j:

$$p_{new} = 1\cdot (1,0) + p = (1, 0)$$

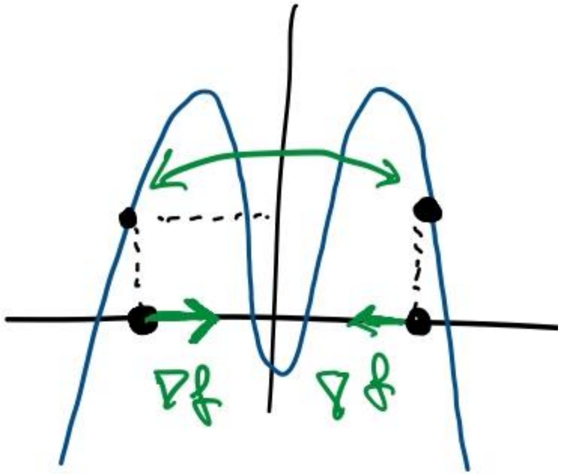
$$f(p_{new}) = 3 < 4$$

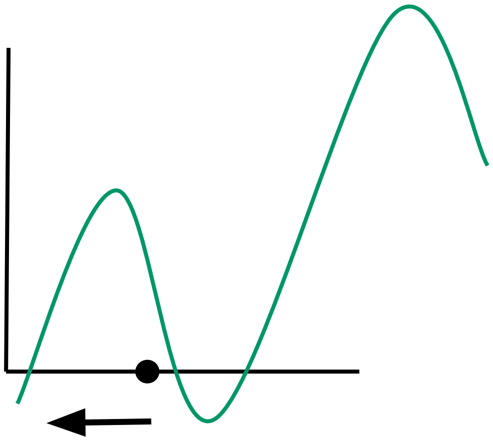
Moving p along the gradient will result in the fastest increase in f from p.

Gradients

However, the gradient expresses the **instantaneous** rate of change. At p, ∇f_p is the steepest but the highest value of f will depend on how many units we step in that direction. If we step too many units away, the instantaneous change in f is no longer representative of what values f will take.

Example:





Gradient Descent

Given a “smooth” function f for which there exists no closed form solution for finding its **maximum**, we can find a local maximum through the following steps:

1. Define a step size $\boldsymbol{\alpha}$ (tuning parameter)
2. Initialize p to be random
3. $p_{new} = \boldsymbol{\alpha} \nabla f_p + p$
4. p □ p_{new}
5. Repeat 3 & 4 until $p \sim p_{new}$

To find a local **minimum**, just use $-\nabla f_p$

Gradient Descent

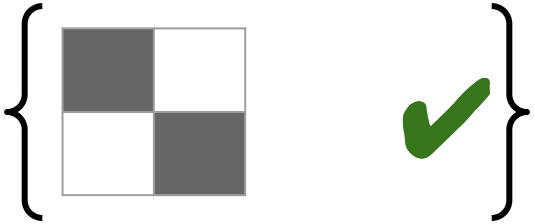
Notes about α:

- If α is too large, GD may overshoot the maximum, take a long time to or never be able to converge
- If α is too small, GD may take too long to converge

Gradient Descent

Let’s apply this to our diagonal problem to find the weights and bias for logistic regression.

Assume we have the following dataset:



$[0\ 1\ 1\ 0]^T$

Gradient Descent

Recall:

$$\mathrm{Cost}(w, b)$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[yi\log(\sigma(-w^T x_i + b)) + (1 - y_i)\log(1 - \sigma(-w^T x_i + b))\right]$$

Gradient Descent

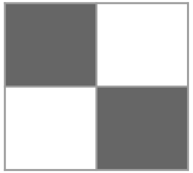
We need to compute ∇Cost(w, b):

$$\nabla \mathrm{Cost}(w,b) = \left[\frac{\partial}{\partial w}\mathrm{Cost}, \frac{\partial}{\partial b}\mathrm{Cost}\right]$$

$$\frac{\partial}{\partial w}\mathrm{Cost} = \frac{1}{n}\sum_{i=1}^{n} x_i(y_i - \sigma(-w^T x_i + b))$$

$$\frac{\partial}{\partial b}\mathrm{Cost} = \frac{1}{n}\sum_{i=1}^{n} \sigma(-w^T x_i + b) - y_i$$

Gradient Descent



$= [0\ 1\ 1\ 0]^T$

1. Start with random w and b:

$w = [0\ 0\ 0\ 0]^T$, $b = 0$

Note: $\boldsymbol{\sigma}(0) = 0.5$

Gradient Descent

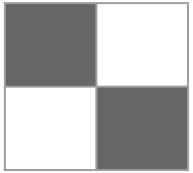
$$-\frac{1}{n}\sum_{i=1}^{n}\left[yi\log(\sigma(-w^{T}x_i+b))+(1-y_i)\log(1-\sigma(-w^{T}x_i+b))\right]$$

2. Compute the Cost(w, b)

Cost([0 0 0 0]^{T} , 0) = -1 log(**σ**(0)) = -log(0.5)

Gradient Descent

$$\frac{\partial}{\partial w}\text{Cost} = \frac{1}{n}\sum_{i=1}^{n} x_i(y_i - \sigma(-w^T x_i + b))$$



$= [0\ 1\ 1\ 0]^T$

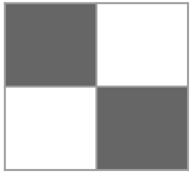
3. Compute the gradient ∇Cost at (w, b)

$$\frac{\partial}{\partial w}\text{Cost} = \frac{1}{1}\sum_{i=1}^{1}\begin{bmatrix}0\\1\\1\\0\end{bmatrix}(1 - \sigma(0)) = \begin{bmatrix}0\\1/2\\1/2\\0\end{bmatrix}$$

Recall we only have one data point

Gradient Descent

$$\frac{\partial}{\partial b}\text{Cost} = \frac{1}{n}\sum_{i=1}^{n}\sigma(-w^T x_i + b) - y_i$$



= [0 1 1 0]^T

3. Compute the gradient ∇Cost at (w, b)

$$\frac{\partial}{\partial b}\text{Cost} = \frac{1}{1}\sum_{i=1}^{1}(\sigma(0) - 1) = -\frac{1}{2}$$

Recall we only have one data point

Gradient Descent

4. Adjust w & b by taking **α** steps in the direction of -∇Cost$_{(w, b)}$

$$w_{\text{new}} = -\alpha \begin{bmatrix} 0 \\ 1/2 \\ 1/2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -\alpha/2 \\ -\alpha/2 \\ 0 \end{bmatrix}$$

$$b_{\text{new}} = \alpha \frac{1}{2} + 0 = \frac{\alpha}{2}$$

Gradient Descent

5. Compute the updated Cost

$$\mathrm{Cost}(\begin{bmatrix} 0 \\ -\alpha/2 \\ -\alpha/2 \\ 0 \end{bmatrix}, \frac{\alpha}{2}) = -\log(\sigma(\alpha + \frac{1}{2}))$$

For what values of α is the Cost reduced?

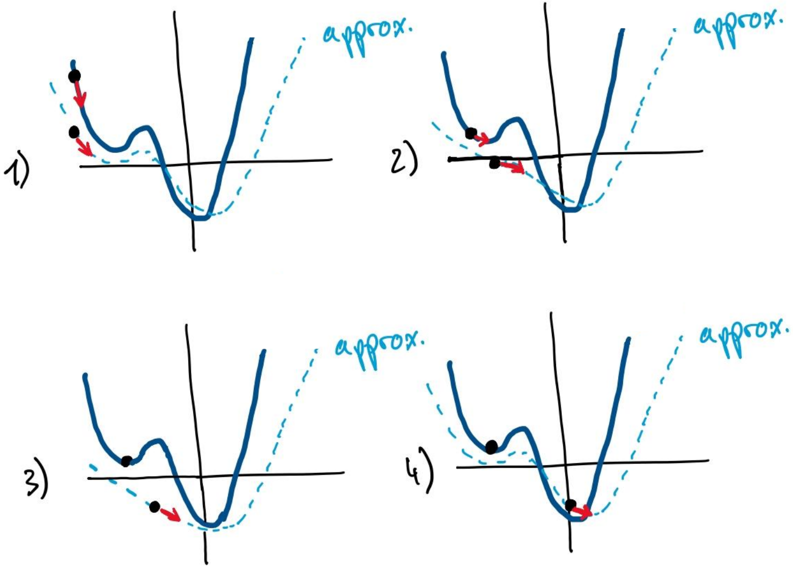
Stochastic Gradient Descent

Recall the Cost is computed for the entire dataset. This has some limitations:

1. It’s expensive to run
2. The result we get depends only on the initial starting point

Stochastic Gradient Descent

Goal: Approximate the gradient of the Cost using a sample of the data (batch)



Note

The magnitude of ∇f_p depends on p. A p gets closer to the min / max, the size of ∇f_p decreases.

This also means that points p that contain more “information” have larger gradients. So the order with which this process is exposed to examples matters.