CMPS 460 – Spring 2022

MACHINE LEARNING

Tamer Elsayed

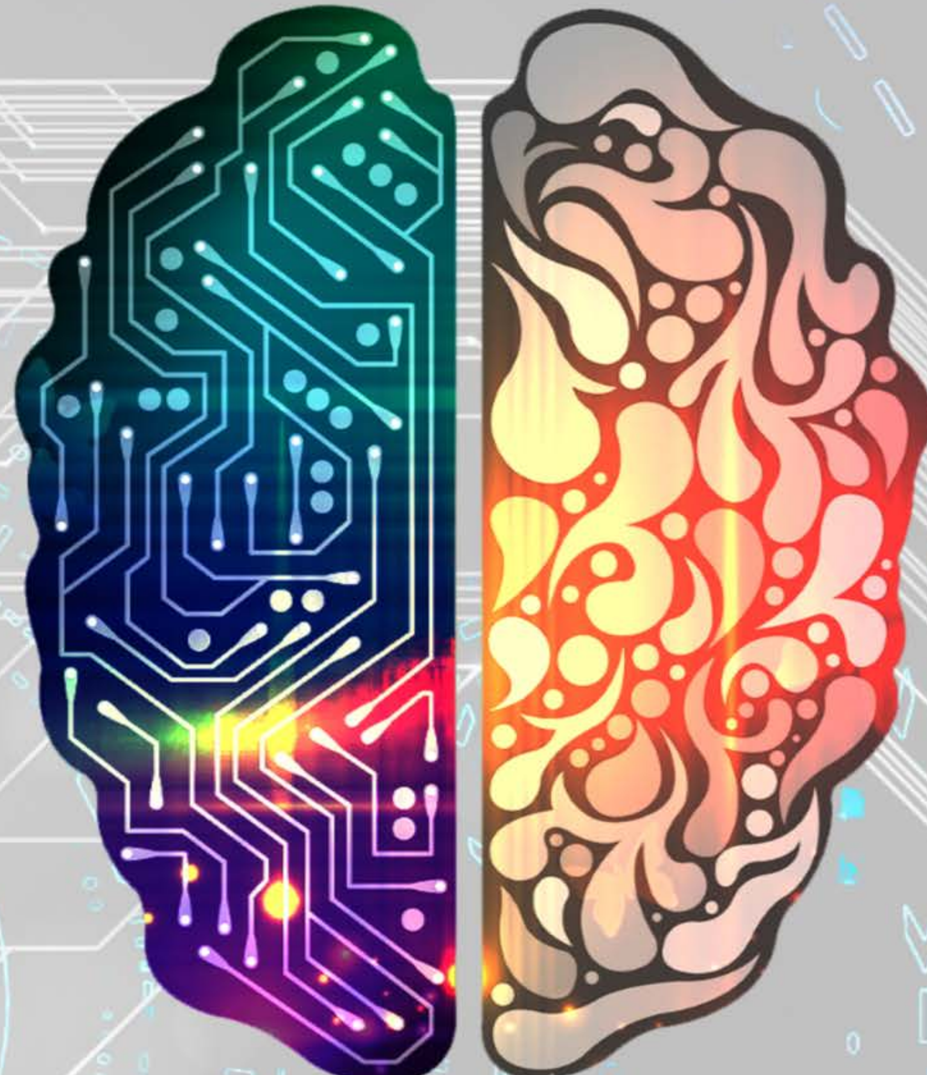


Image hosted by: WittySparks.com | Image source: Pixabay.com

7.c

Linear Models: Gradient Descent

Chapter 7: 7.4

Optimization Framework

Objective function

Loss function measures how well classifier fits training data

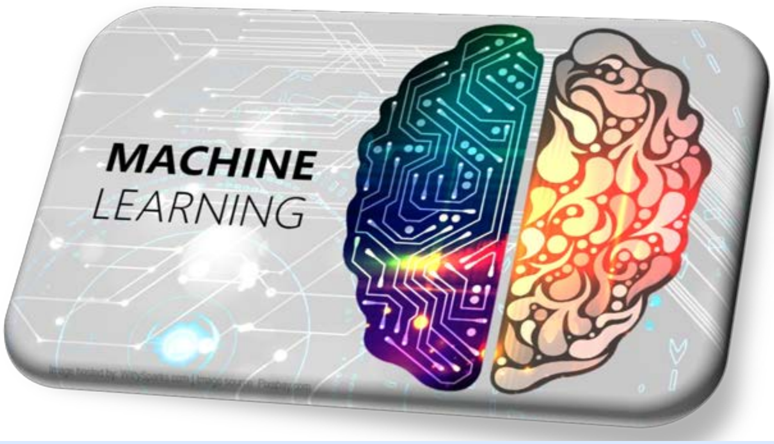
Regularizer prefers solutions that generalize well

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \cancel{\mathbb{I}(y_n(\mathbf{w}^T \mathbf{x}_n + b) < 0)} + \lambda R(\mathbf{w}, b)$$

$$l(y_n, \hat{y}_n)$$

λ: parameter that controls the importance of the regularization term

- Different loss function approximations
  - easier to optimize
- Regularizer
  - prevents overfitting/prefers simple models.



Optimization with Gradient Descent

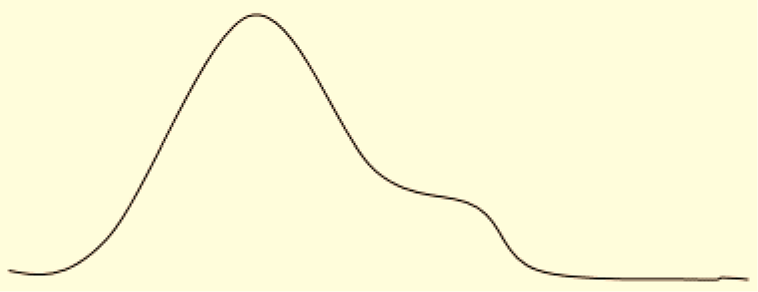
Gradient Descent

- A general solution for our optimization problem

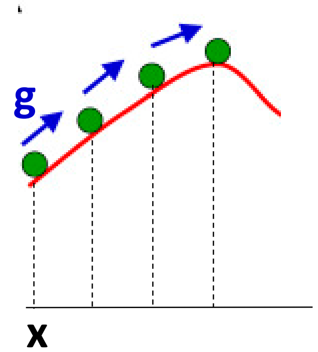
$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T \cancel{\mathbf{x}_n} + b) < 0) + \lambda R(\mathbf{w}, b)$$

$$l(y_n, \hat{y}_n)$$

Climbing a hill!



Gradient-based Optimization



- Suppose the goal is to find maximum of f(**x**).

1. The optimizer maintains a current estimate of the parameter of interest, **x**.
2. At each step, measure the **gradient g** of f(**x**) at the current location, **x**.
3. Then take a step in the direction of the gradient, where the size of the step is controlled by η.

$$\mathbf{x} \leftarrow \mathbf{x} + \eta\, \mathbf{g}$$ **(Gradient Ascent)**

- **Gradient Descent**: opposite of gradient ascent.

$$\mathbf{x} \leftarrow \mathbf{x} - \eta\, \mathbf{g}$$ **(Gradient Descent)**

Gradient Descent

- A general solution for our optimization problem

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T \cancel{x_n} + b) < 0) + \lambda R(\mathbf{w}, b)$$

$$l(y_n, \hat{y}_n)$$

***Idea*: take iterative steps to update parameters in the direction of the gradient**

Gradient Descent in 1-Dim

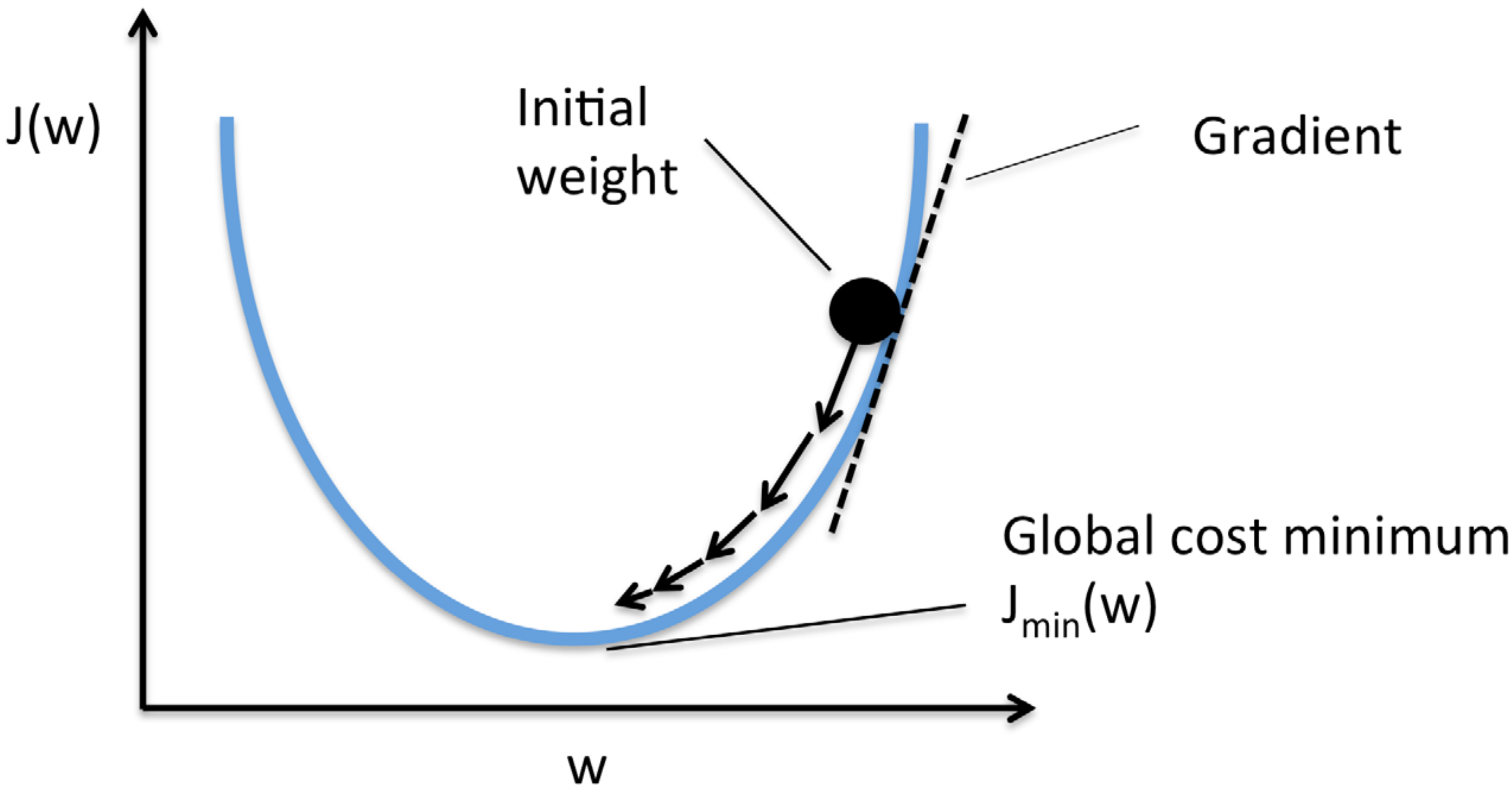


Image source: https://hackernoon.com/gradient-descent-aynk-7cbe95a778da

Gradient Descent in 2-Dim

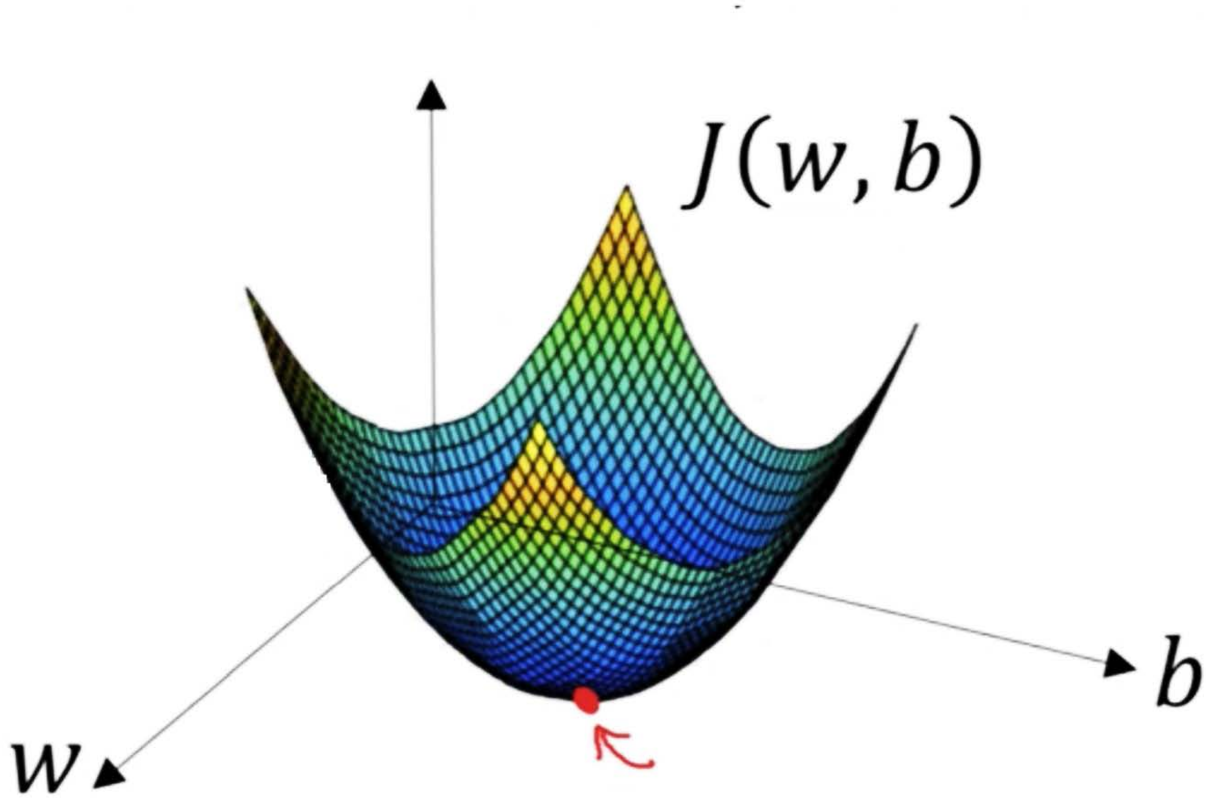


Image source: https://towardsdatascience.com/gradient- descent-in-a-nutshell-eaf8c18212f0

Gradient Descent Algorithm

Objective function to minimize

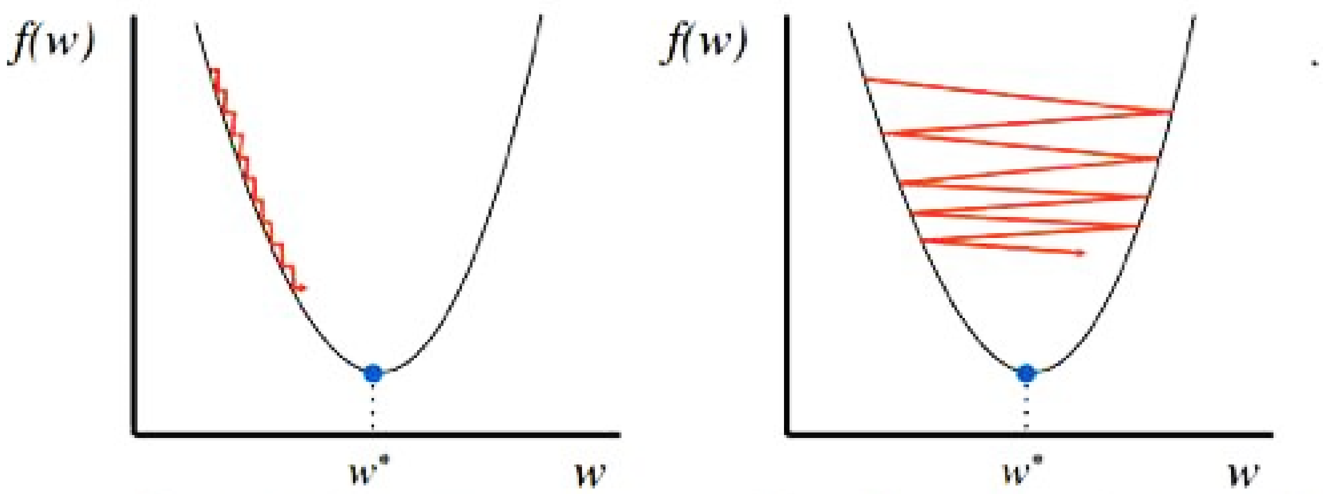
Number of steps

Step size

Algorithm 22 GradientDescent($\mathcal{F}$, K, η_1, ...)

1: $z^{(0)} \leftarrow \langle 0, 0, \dots, 0 \rangle$ // initialize variable we are optimizing
2: **for** $k = 1 \dots K$ **do**
3: $\quad g^{(k)} \leftarrow \nabla_z \mathcal{F}|_{z^{(k-1)}}$ // compute gradient at current location
4: $\quad z^{(k)} \leftarrow z^{(k-1)} - \eta^{(k)} g^{(k)}$ // take a step down the gradient
5: **end for**
6: **return** $z^{(K)}$

Impact of Step Size



Too small: converge very slowly

Too big: overshoot and even diverge

[Image source: https://towardsdatascience.com/gradient- descent-in-a-nutshell-eaf8c18212f0](https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0)

Gradient Descent

When to stop?

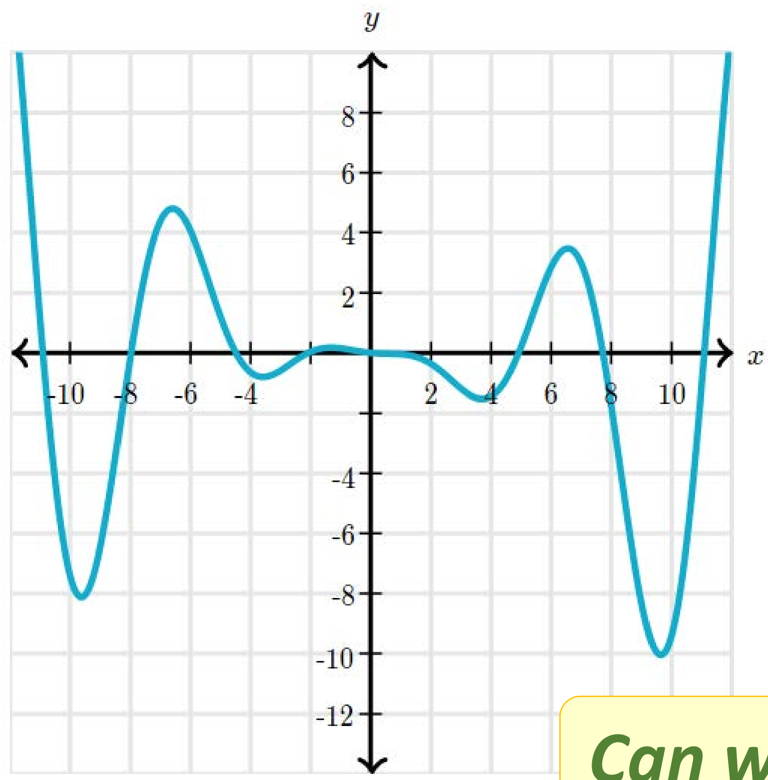
- When the gradient gets close to zero
- When the objective stops changing much
- When the parameters stop changing much
- When performance on held-out dev set plateaus

How to choose the step size?

- Constant
- Better: start with large steps, then take smaller steps

Example

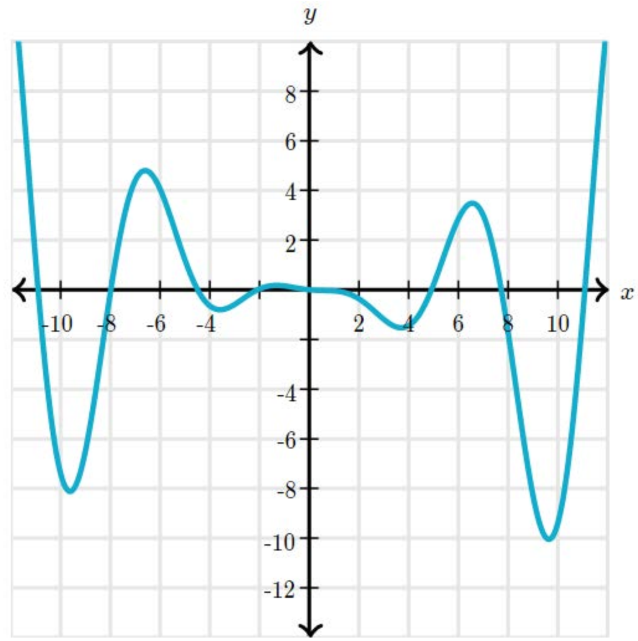
$$f(x) = \frac{x^2 \cos(x) - x}{10}$$



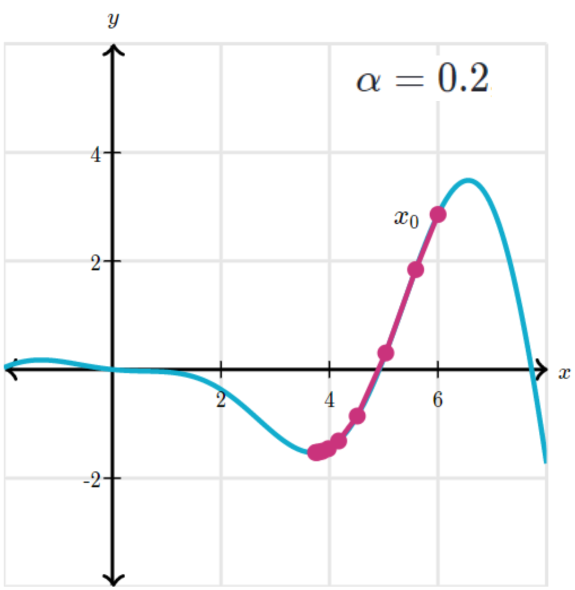
Can we apply GD?

https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/what-is-gradient-descent

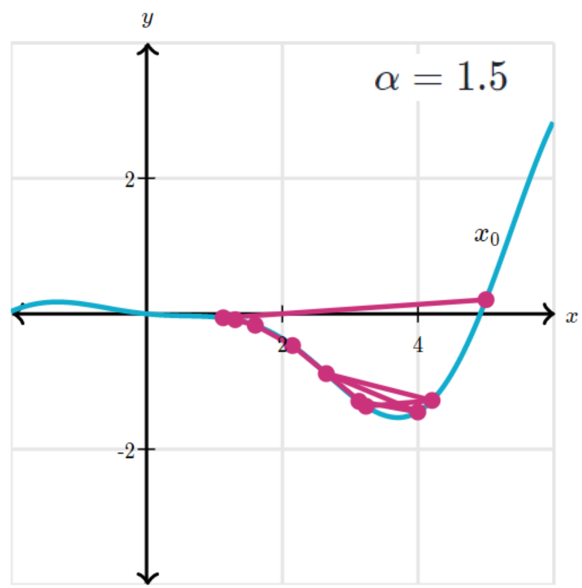
Example



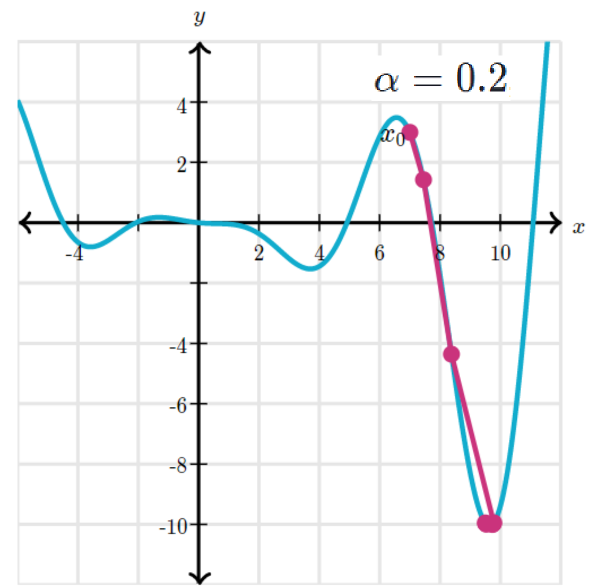
y
x
8
6
4
2
-4
-6
-8
-10
-12
-10
-8
-6
-4
2
4
6
8
10



$\alpha = 0.2$
x_0
y
x
4
2
-2
2
4
6



$\alpha = 1.5$
x_0
y
x
2
-2
2
4



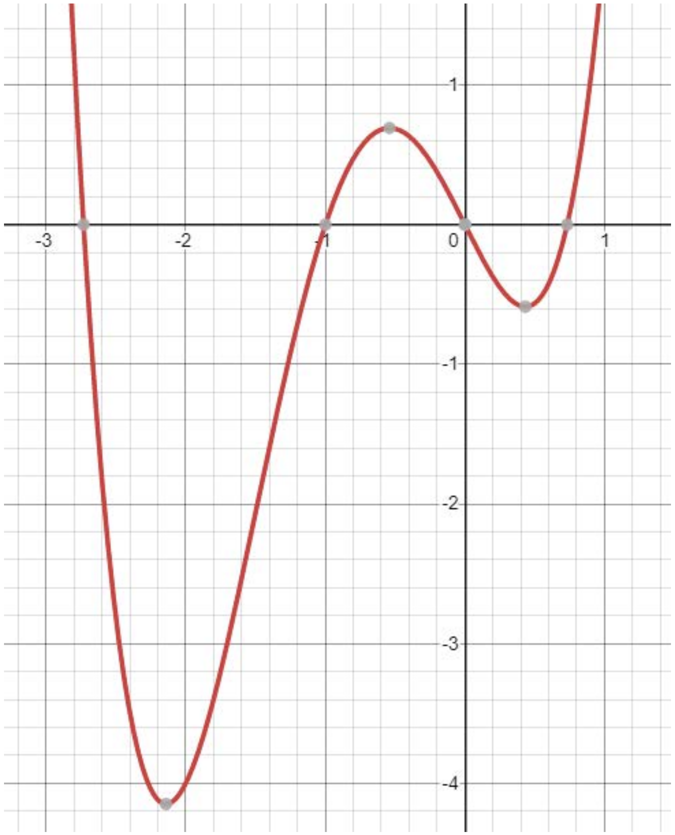
$\alpha = 0.2$
x_0
y
x
4
2
-4
-6
-8
-10
-4
2
4
6
8
10

Exercise

$$x^4 + 3x^3 - 2x$$

$x_0 = 1, \propto = 0.2$

$x_0 = 1, \propto = 1.5$



Gradients for Multivariate Objectives

- Consider the following learning objective

$$\mathcal{L}(\boldsymbol{w}, b) = \sum_n \exp\left[-y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right] + \frac{\lambda}{2} ||\boldsymbol{w}||^2$$

What do we need to do to run gradient descent?

(1) Derivative with respect to b

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial}{\partial b} \sum_n \exp\left[-y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right] + \frac{\partial}{\partial b} \frac{\lambda}{2} ||\boldsymbol{w}||^2 \tag{6.12}$$

$$= \sum_n \frac{\partial}{\partial b} \exp\left[-y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right] + 0 \tag{6.13}$$

$$= \sum_n \left(\frac{\partial}{\partial b} - y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right) \exp\left[-y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right] \tag{6.14}$$

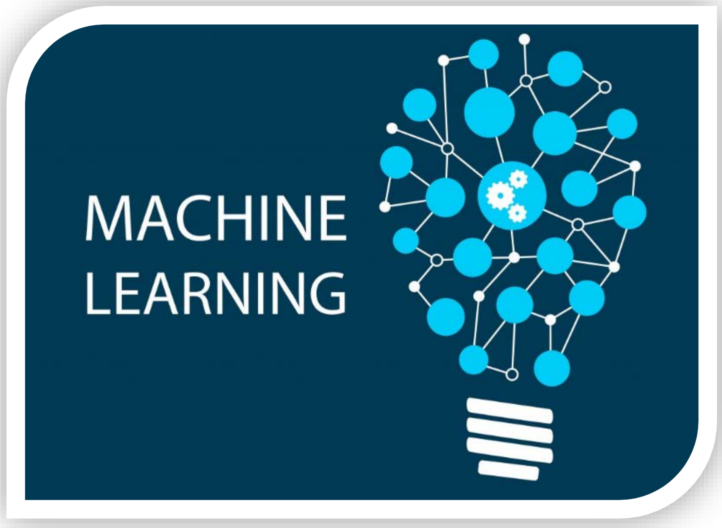
$$= -\sum_n y_n \exp\left[-y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right] \tag{6.15}$$

(2) Gradient with respect to *w*

$$\nabla_{\boldsymbol{w}}\mathcal{L} = \nabla_{\boldsymbol{w}}\sum_n \exp\left[-y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\right] + \nabla_{\boldsymbol{w}}\frac{\lambda}{2}\left|\left|\boldsymbol{w}\right|\right|^2 \tag{6.16}$$

$$= \sum_n \left(\nabla_{\boldsymbol{w}} - y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\right)\exp\left[-y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\right] + \lambda\boldsymbol{w} \tag{6.17}$$

$$= -\sum_n y_n\boldsymbol{x}_n \exp\left[-y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\right] + \lambda\boldsymbol{w} \tag{6.18}$$



Subgradients

Subgradients

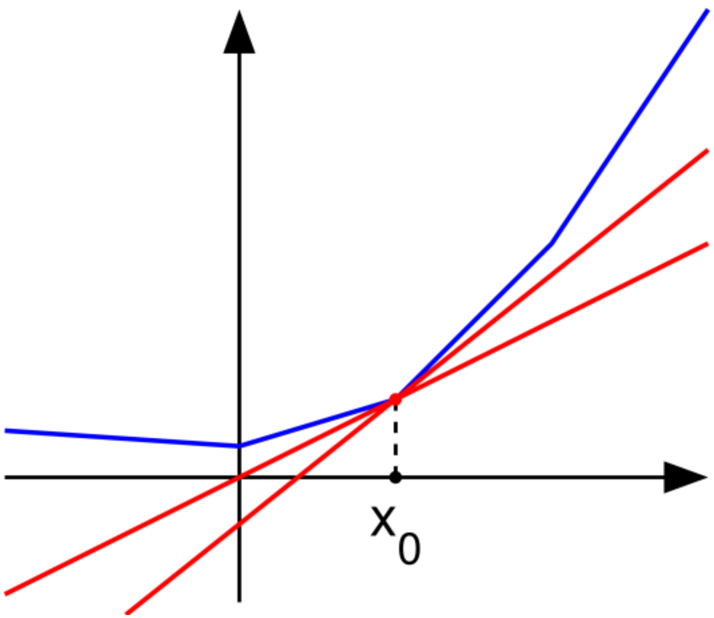
- Problem: some objective functions are not differentiable everywhere
  - e.g., Hinge loss, l_1 norm

Solution: Subgradient Optimization

- Let's ignore the problem, and just try to apply gradient descent anyway!!
- We will just differentiate by parts…

Subgradients

- Generalizes the derivative to functions which are not differentiable.
- For any x_0 in the domain of the function, one can draw a line which goes through the point $(x_0, f(x_0))$ and which is everywhere either touching or below the graph of f.
- Set-valued



Example: Subgradient of Hinge Loss

- For a given example n

$$\partial_{\boldsymbol{w}} \max\{0, 1 - y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\} \tag{7.24}$$

$$= \partial_{\boldsymbol{w}} \begin{cases} 0 & \text{if } y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 1 \\ 1 - y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) & \text{otherwise} \end{cases} \tag{7.25}$$

$$= \begin{cases} \partial_{\boldsymbol{w}} 0 & \text{if } y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 1 \\ \partial_{\boldsymbol{w}} 1 - y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) & \text{otherwise} \end{cases} \tag{7.26}$$

$$= \begin{cases} \boldsymbol{0} & \text{if } y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 1 \\ -y_n \boldsymbol{x}_n & \text{otherwise} \end{cases} \tag{7.27}$$

Subgradient Descent for Hinge Loss

Algorithm 23 HingeRegularizedGD(**D**, λ, *MaxIter*)

```
w ← ⟨0,0,...0⟩   ,   b ← 0                      // initialize weights and bias
for iter = 1 ... MaxIter do
    g ← ⟨0,0,...0⟩   ,   g ← 0                  // initialize gradient of weights and bias
    for all (x,y) ∈ D do
        if y(w · x + b) ≤ 1 then
            g ← g + y x                         // update weight gradient
            g ← g + y                           // update bias derivative
        end if
    end for
    g ← g − λw                                  // add in regularization term
    w ← w + ηg                                  // update weights
    b ← b + ηg                                  // update bias
end for
return w, b
```

Anything wrong here?

Perceptron …

Algorithm 5 $\textsc{PerceptronTrain}(\mathbf{D}, \mathit{MaxIter})$

```
w_d ← 0, for all d = 1 ... D                         // initialize weights
b ← 0                                                // initialize bias
for iter = 1 ... MaxIter do
    for all (x,y) ∈ D do
        a ← Σ_{d=1}^{D} w_d x_d + b                  // compute activation for this example
        if ya ≤ 0 then
            w_d ← w_d + y x_d, for all d = 1 ... D   // update weights
            b ← b + y                                // update bias
        end if
    end for
end for
return w_0, w_1, ..., w_D, b
```

What is it optimizing?

Summary: Gradient Descent

- A generic algorithm to minimize objective functions
- Works well as long as functions are well behaved (i.e., convex)
- Subgradient descent can be used at points where derivative is not defined.

- Can we do better?
  - For some objectives, we can find closed form solutions (see CIML 7.6)