Lecture 13: Gradient Descent Again

COMP 411, Fall 2021
Victoria Manfredi

WESLEYAN UNIVERSITY

Acknowledgements: These slides are based primarily on those created by Michael Paul (U of Colorado)

Today's Topics

Prediction functions

Finding maxima and minima

Gradient descent

Revisiting perceptron

Prediction functions

LEARNING PARAMETERS

Prediction functions

Remember: a **prediction function** is the function that predicts what the output should be, given the input

Prediction functions

Linear regression:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

Linear classification (perceptron):

$$f(\mathbf{x}) = \begin{cases} 1, & \mathbf{w}^T\mathbf{x} \geq 0 \\ -1, & \mathbf{w}^T\mathbf{x} < 0 \end{cases}$$

Need to learn what $\mathbf{w}$ should be!

Learning parameters

Goal is to learn to minimize error

- ideally: true error
- instead: training error

The **loss function** gives the training error when using **parameters** $\mathbf{w}$, denoted $L(\mathbf{w})$.

- Also called **cost function**
- More general: **objective function**: in general, objective could be to minimize or maximize; with loss/cost functions, we want to minimize

Learning parameters

Goal is to minimize loss function.

How do we minimize a function?

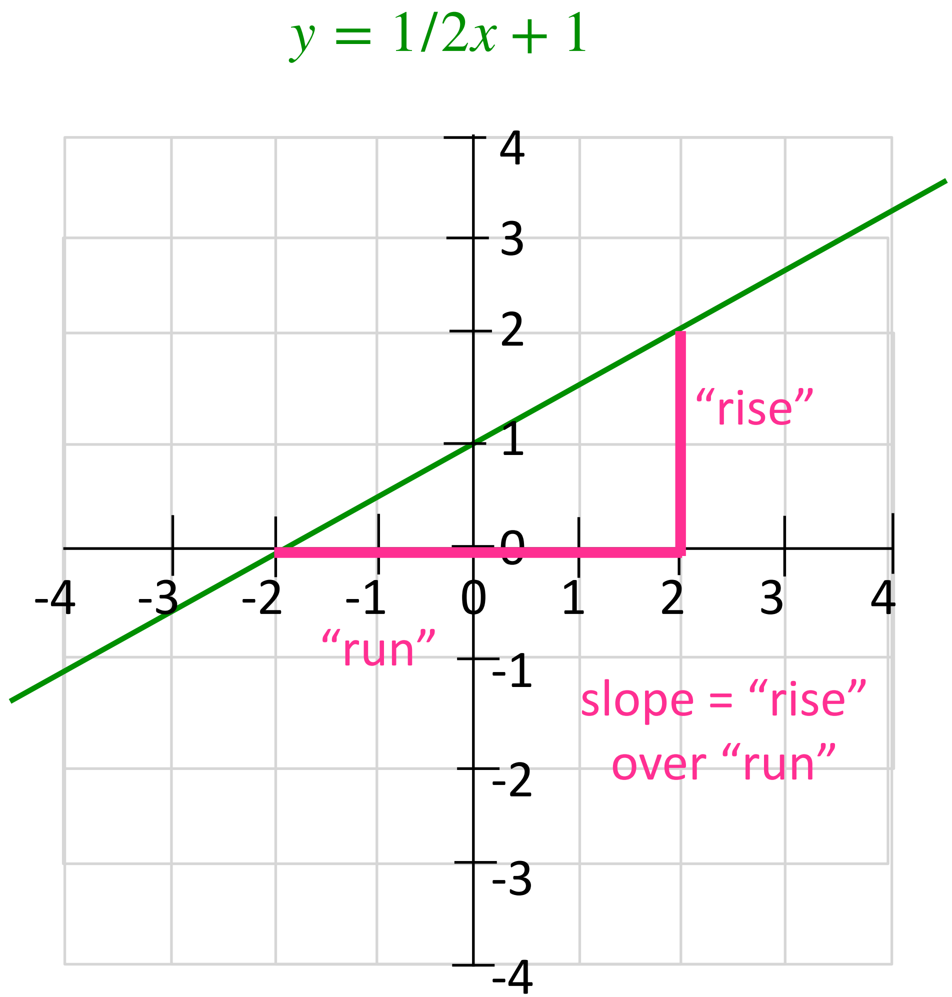
Let’s review some math …

Finding minima and maxima

USING DERIVATIVES

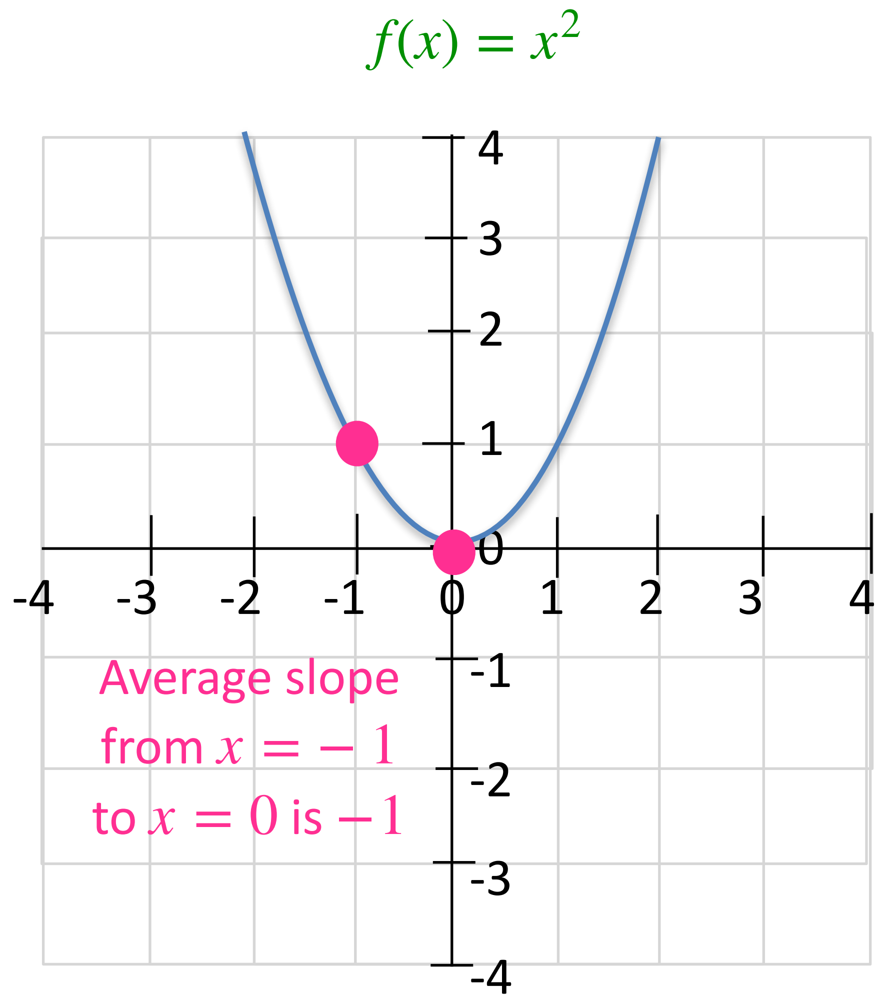
Rate of change

The slope of a line is also called the rate of change of the line



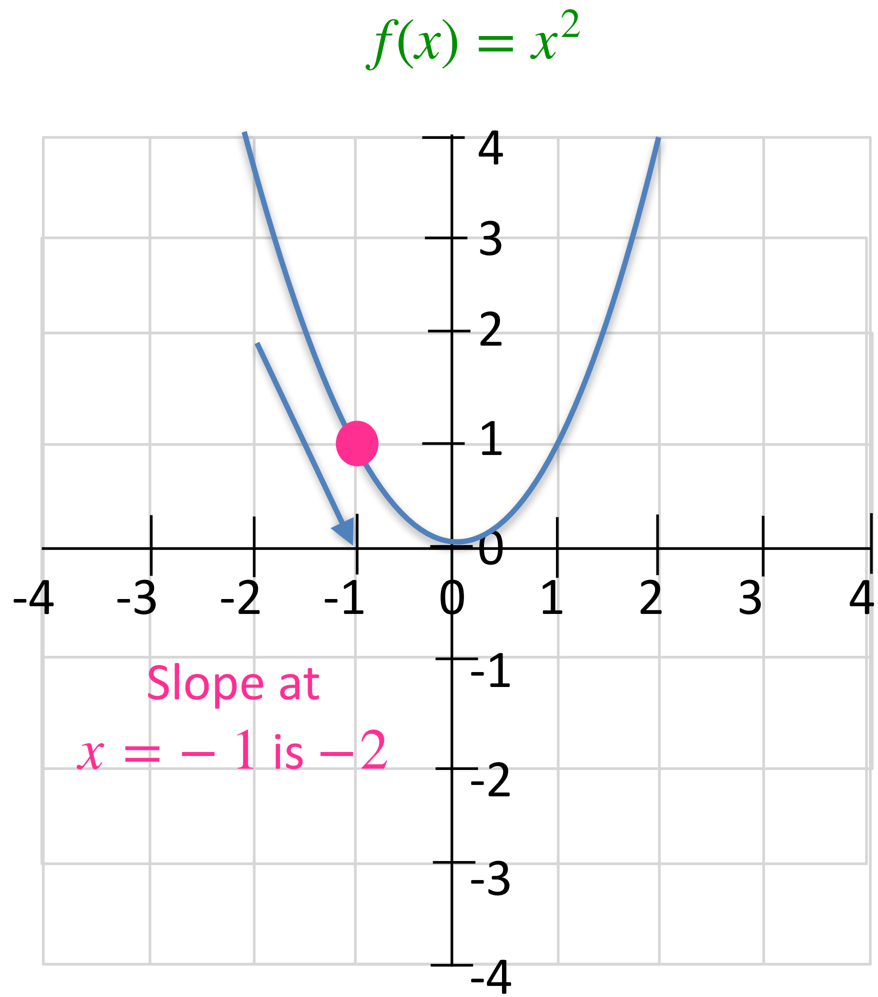
Rate of change

For nonlinear functions, the "rise over run" formula gives you the average rate of change between two points



Rate of change

There is also a concept of rate of change at individual points (rather than two points)



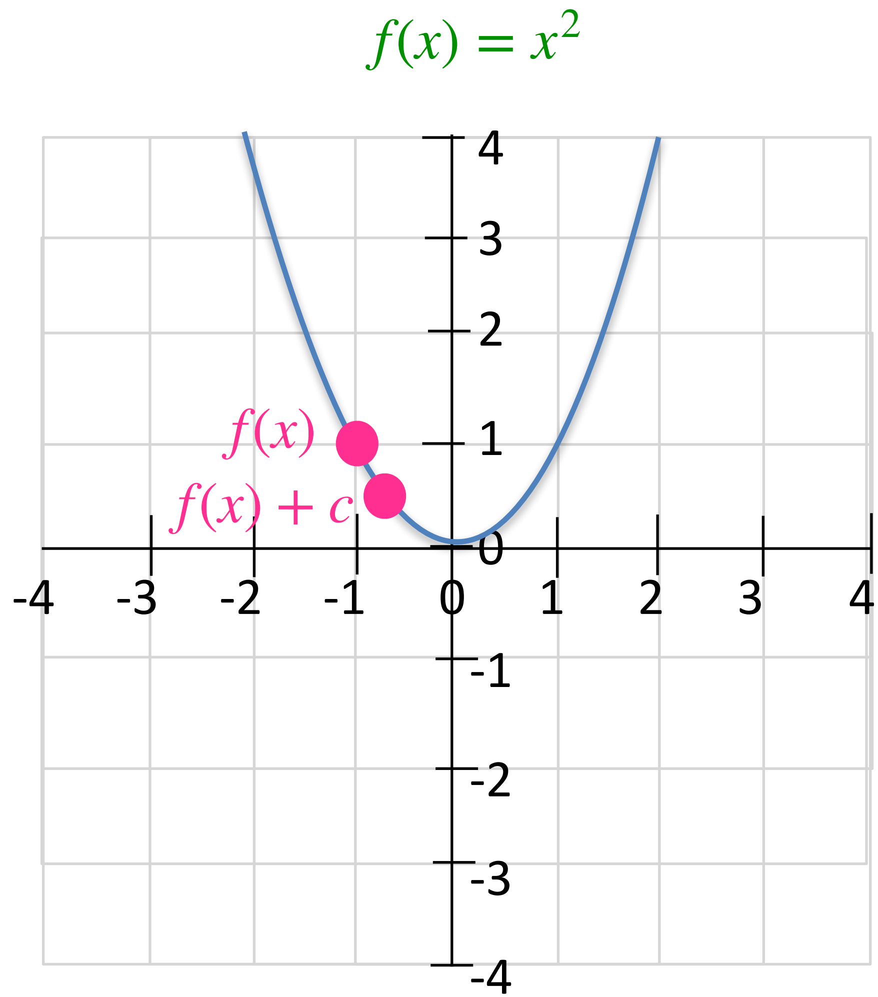
Rate of change

The slope at a point is called the **derivative** at that point

Intuition: measure the slope between two points that are really close together

$$\frac{f(x+c)-f(x)}{c}$$

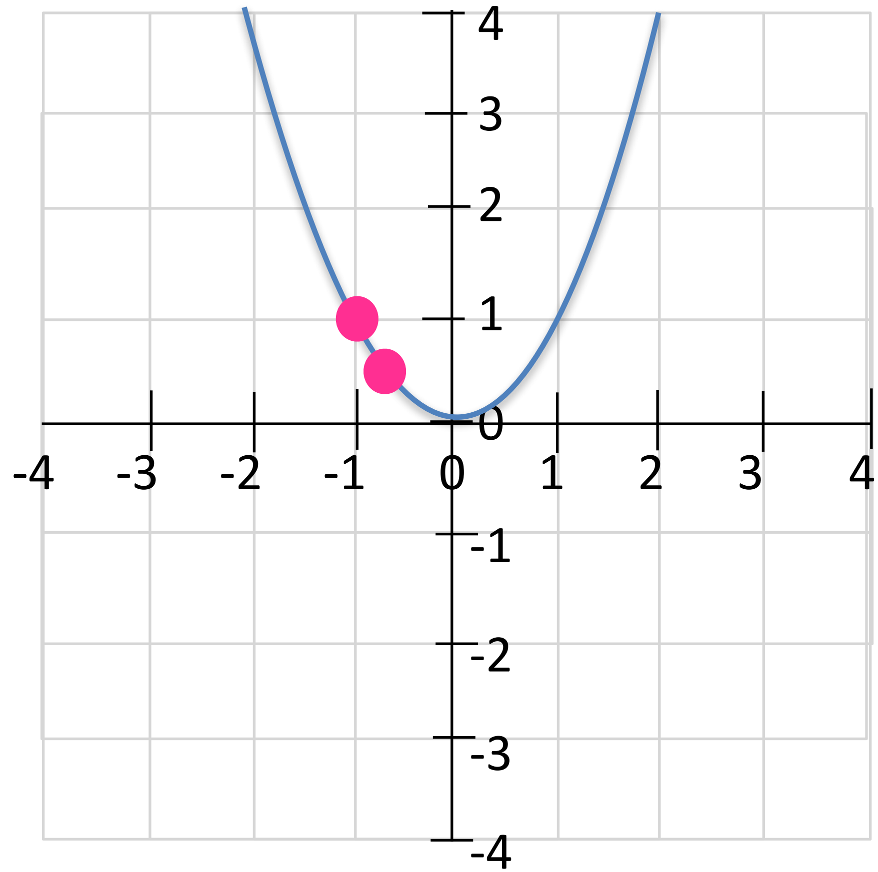
Limit as c goes to zero



Rate of change

The slope at a point is called the **derivative** at that point

$f(x) = x^2$



Intuition: measure the slope between two points that are really close together

Maxima and minima

Whenever there is a peak in the data, this is a **maximum**

The **global** maximum is the highest peak in the entire data set, or the largest $f(x)$ value the function can output

A **local** maximum is any peak, when the rate change switches from positive to negative

Maxima and minima

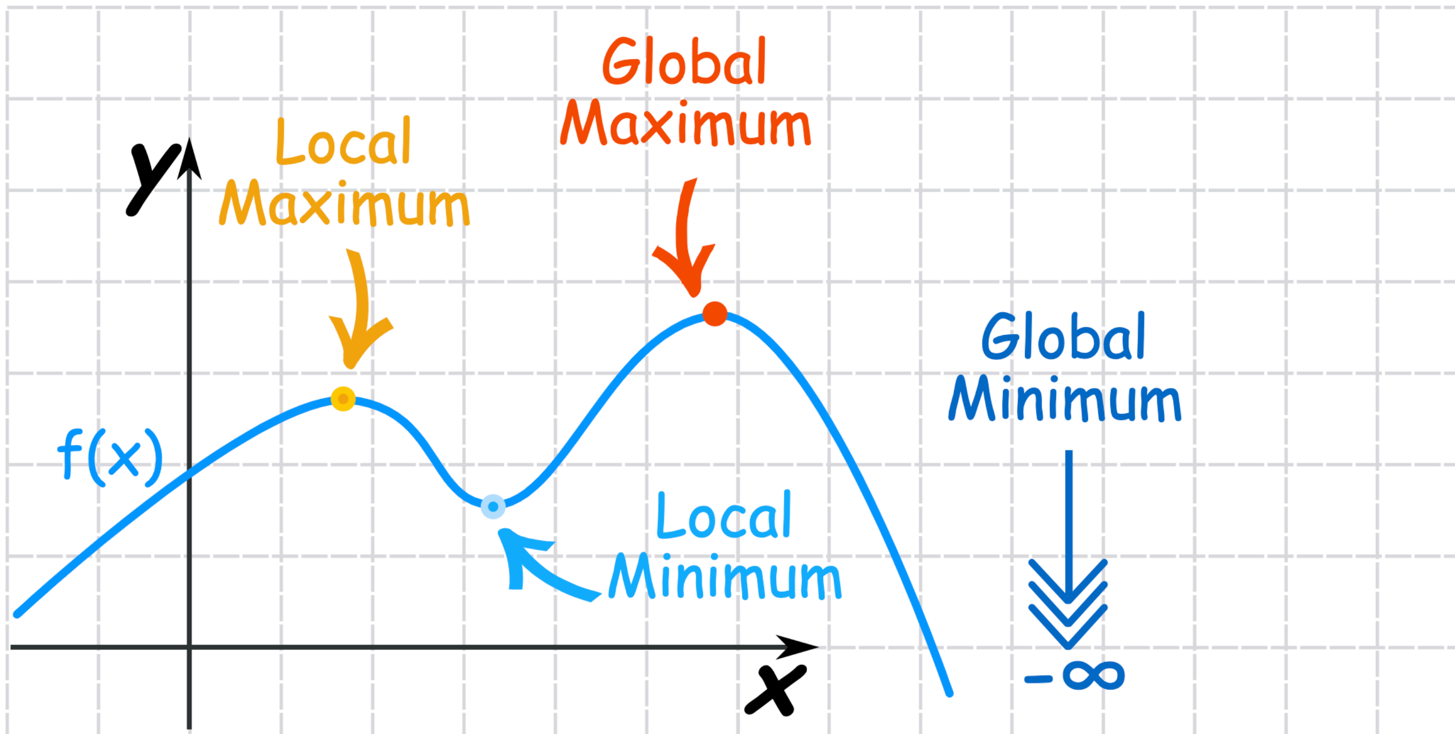
Whenever there is a trough in the data, this is a **minimum**

The **global** minimum is the lowest trough in the entire data set, or the smallest $f(x)$ value the function can output

A **local** minimum is any trough, when the rate change switches from negative to positive

Maxima and minima

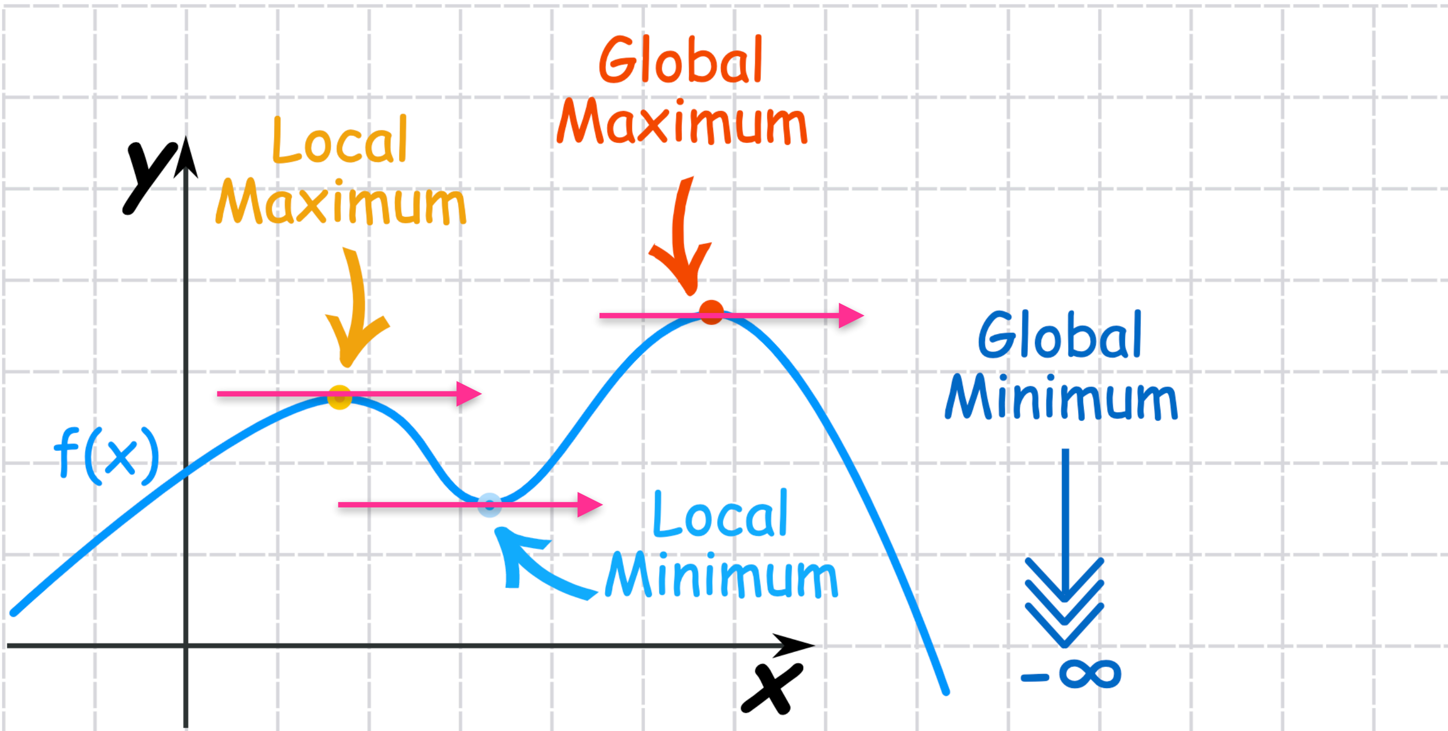
All global maxima and minima are also local maxima and minima



https://www.mathsisfun.com/algebra/functions-maxima-minima.html

Finding minima

The derivative is zero at any local maximum or minimum



https://www.mathsisfun.com/algebra/functions-maxima-minima.html

Finding minima

The derivative is zero at any local maximum or minimum

One way to find a minimum: set $f'(x) = 0$ and solve for x

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

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

$f'(x) = 0$ when $x = 0$ so minimum at $x = 0$

Finding minima

The derivative is zero at any local maximum or minimum

One way to find a minimum: set $f'(x) = 0$ and solve for x

- For most functions, there isn't a way to solve this
- Instead: algorithmically search different values of x until you find one that results in a gradient near 0

Finding minima

If the derivative is positive, the function is **increasing**

- don't move in that direction, because you'll be moving away from a trough

If the derivative is negative, the function is **decreasing**

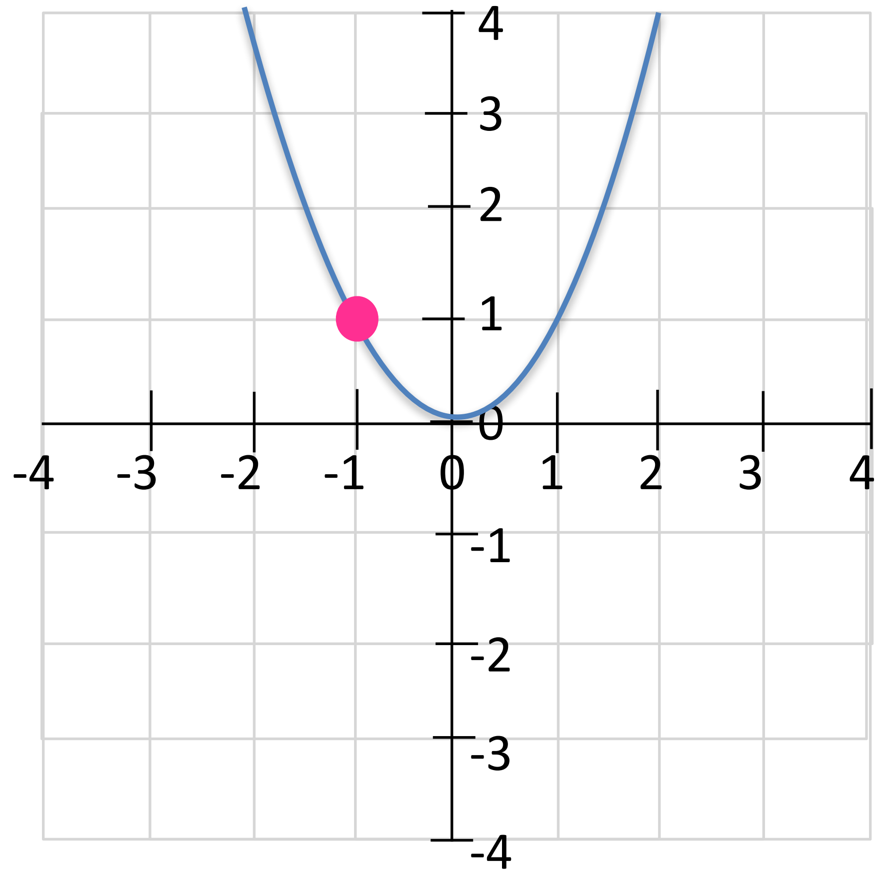
- Keep going, since you're getting closer to a trough

Finding minima

$f(x) = x^2$

$f'(-1) = -2$

For nonlinear functions, the “rise over run” formula gives you the average rate of change between two points



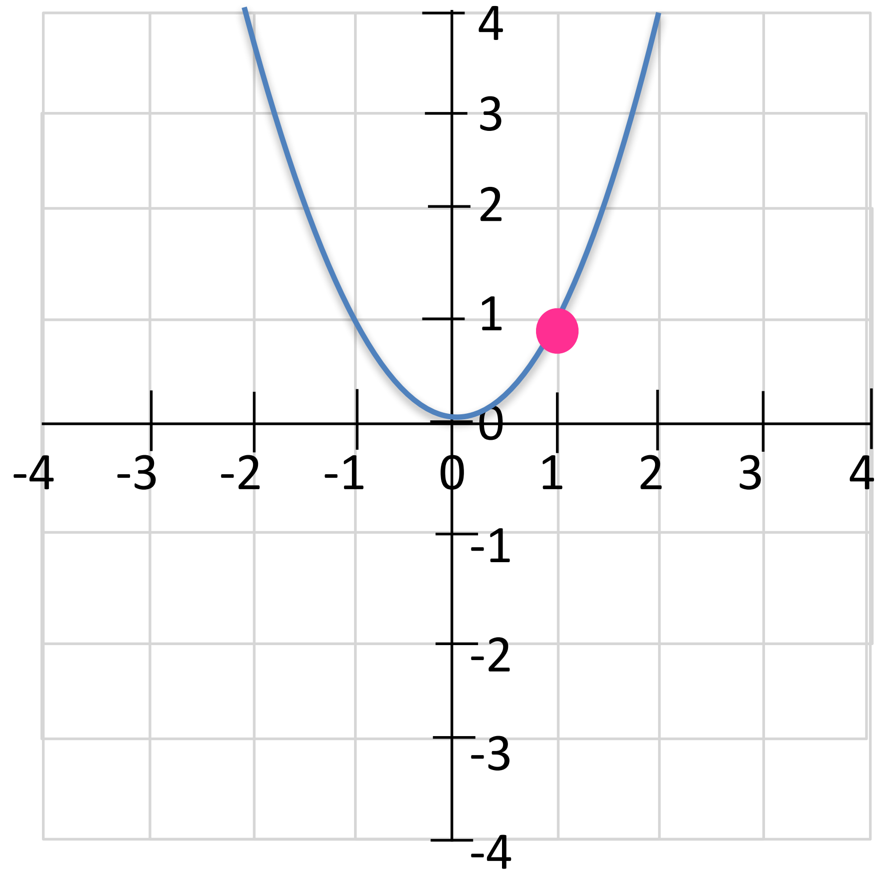
At $x = -1$ the function is decreasing as x gets larger. This is what we want. So let’s make x larger. Increase x by the size of the gradient: $-1 + 2 = 1$

Finding minima

For nonlinear functions, the "rise over run" formula gives you the average rate of change between two points

$f(x) = x^2$

$f'(1) = 2$



At $x = 1$ the function is increasing as x gets larger. This is not what we want. So let's make x smaller. Decrease x by the size of the gradient: $1 - 2 = -1$

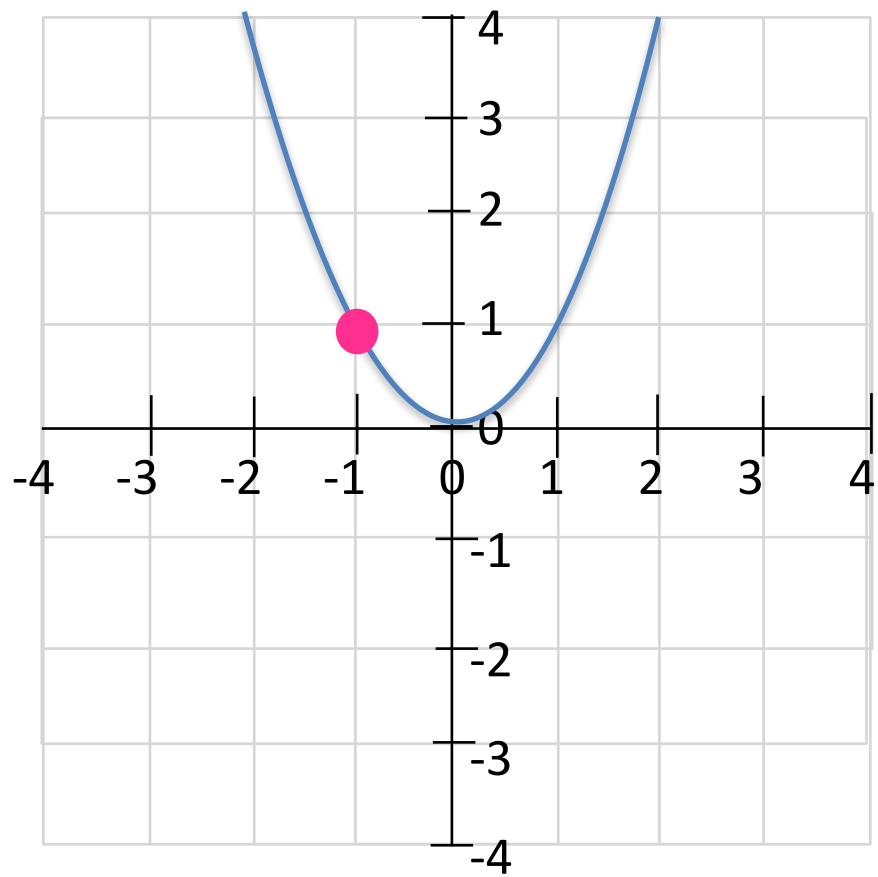
Finding minima

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

$$f'(-1) = -2$$

We will keep jumping between the same two points this way

We can fix this by using a learning rate or step size



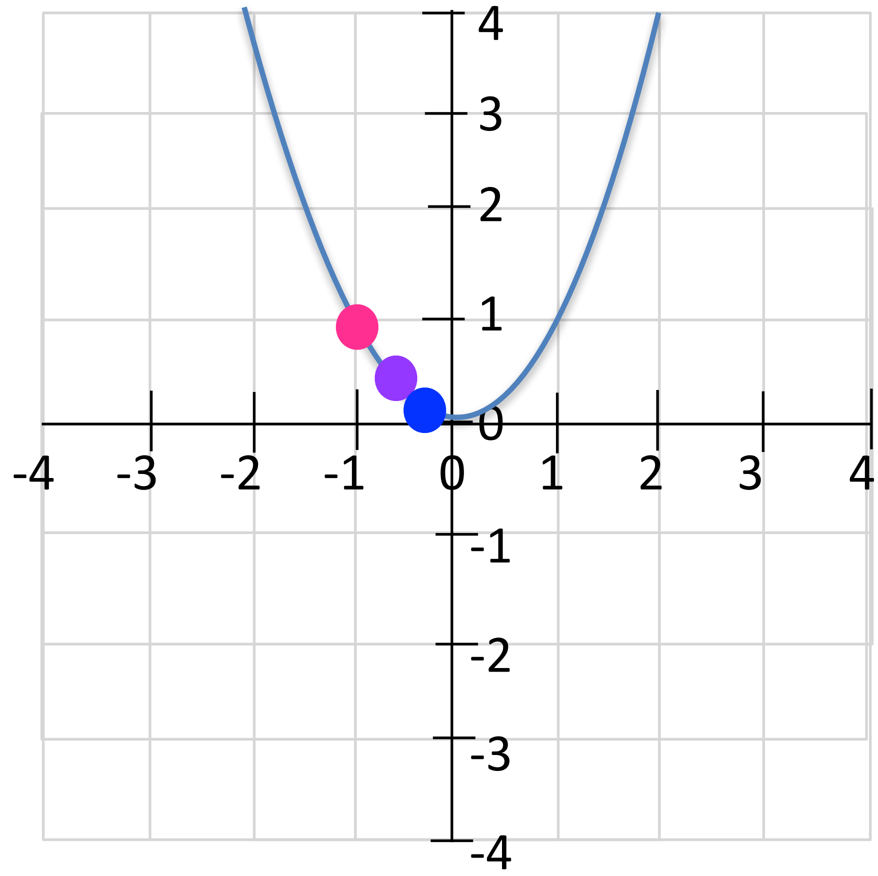
Finding minima

$f(x) = x^2$

$f'(-1) = -2$

$x += 2\eta = ?$

Let's use $\eta = 0.25$



$f'(-1) = -2$

$x = -1 + 2(0.25) = -0.5$

$f'(-0.5) = -1$

$x = -0.5 + 1(0.25) = -0.25$

Eventually we'll reach $x = 0$

Gradient descent

REVIEW

Gradient descent

1. Initialize the parameters $\mathbf{w}$ to some guess (usually all zeroes, or random values)

2. Update the parameters:

   $$\mathbf{w} = \mathbf{w} - \eta \nabla L(\mathbf{w})$$

3. Update the learning rate η

   How? Discuss later …

4. Repeat steps 2-3 until $\nabla L(\mathbf{w})$ is close to zero

Gradient descent

Gradient descent is guaranteed to eventually find a local minimum if

- the learning rate is decreased appropriately
- a finite local minimum exists (i.e., the function doesn't keep decreasing forever)

Gradient ascent

What if we want to find a local maximum?

- Same idea, but the update rule moves the parameters in the opposite direction:

$$\mathbf{w} = \mathbf{w} + \eta \nabla L(\mathbf{w})$$

Learning rate

In order to guarantee that the algorithm will converge, the learning rate should decrease over time. Here is a general formula

- At iteration t

$$\eta_t = c_1/(t^a + c_2) \text{ where}$$

$$0.5 < a < 2$$

$$c_1 > 0$$

$$c_2 \geq 0$$

Stopping criteria

For most functions, you probably won't get the gradient to be exactly equal to $\mathbf{0}$ in a reasonable amount of time

Once the gradient is sufficiently close to $\mathbf{0}$, stop trying to minimize further

How do we measure how close a gradient is to $\mathbf{0}$?

Distance

A special case is the distance between a point and zero (the origin)

$$d(\mathbf{p}, \mathbf{0}) = \sqrt{\sum_{i=1}^{k} p_i^2}$$ also written $||\mathbf{p}||$

This is called the **Euclidean norm** of $\mathbf{p}$

- A norm is a measure of a vector's length
- The Euclidean norm is also called the **L2 norm**

Stopping criteria

Stop when the norm of the gradient is below some threshold, θ:

$$||\nabla L(\mathbf{w})|| < \theta$$

Common values of θ are around .01, but if it is taking too long, you can make the threshold larger

Gradient descent

1. Initialize the parameters $\mathbf{w}$ to some guess (usually all zeroes, or random values)

2. Update the parameters:

$$\mathbf{w} = \mathbf{w} - \eta \nabla L(\mathbf{w})$$

$$\eta = c_1 / (t^a + c_2)$$

3. Repeat step 2 until $||\nabla L(\mathbf{w})|| < \theta$ or until the maximum number of iterations is reached

Stochastic gradient descent

A variant of gradient descent makes updates using an approximate of the gradient that is only based on one instance at a time

$$L_i(\mathbf{w}) = (y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

$$dL_i/dw_j = -\,2x_{ij}(y_i - \mathbf{w}^T\mathbf{x}_i$$

Stochastic gradient descent algorithm

Iterate through the instances in a random order. For each instance x_i, update the weights based on the gradient of the loss for that instance only

$$\mathbf{w} = \mathbf{w} - \eta \nabla L_i(\mathbf{w}; \mathbf{x}_i)$$

The gradient for one instance's loss is an approximation to the true gradient.

Stochastic = Random

The expected gradient is the true gradient.

Revisiting Perceptron

Revisiting perceptron

In perceptron, you increase the weights if they were an underestimate and decrease if they were an overestimate

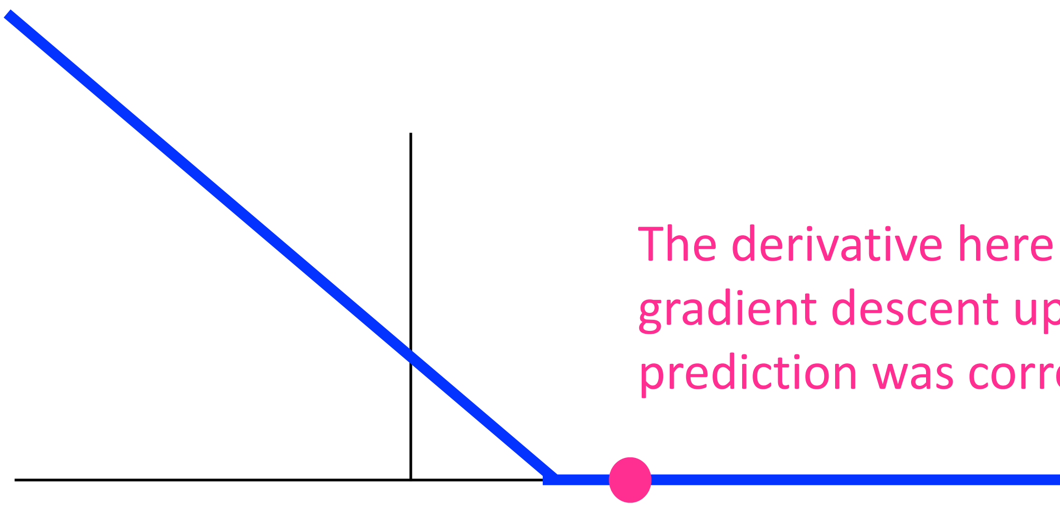
$$w_j = w_j + \eta(y_i - f(x_i))x_{ij}$$

This looks similar to the gradient descent rule

Revisiting perceptron

Perceptron has a different loss function:

$$L_i(\mathbf{w}; \mathbf{x}_i) = \begin{cases} 0 & y_i(\mathbf{w}^T\mathbf{x}_i) \geq 0 \\ -y_i(\mathbf{w}^T\mathbf{x}_i) & \text{otherwise} \end{cases}$$



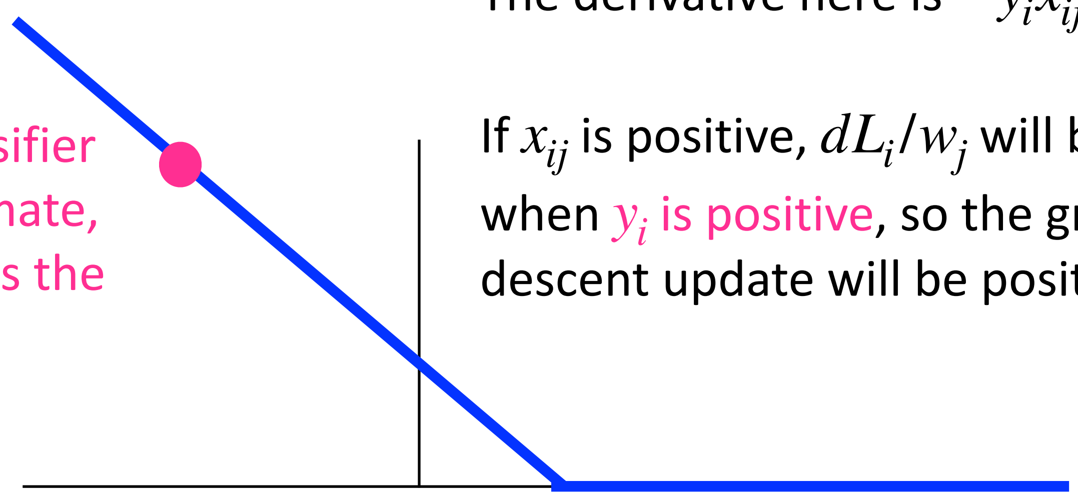
The derivative here is 0. No gradient descent updates if the prediction was correct

Revisiting perceptron

Perceptron has a different loss function:

$$L_i(\mathbf{w}; \mathbf{x}_i) = \begin{cases} 0 & y_i(\mathbf{w}^T\mathbf{x}_i) \geq 0 \\ -y_i(\mathbf{w}^T\mathbf{x}_i) & \text{otherwise} \end{cases}$$

This means the classifier made an underestimate, so perceptron makes the weights larger



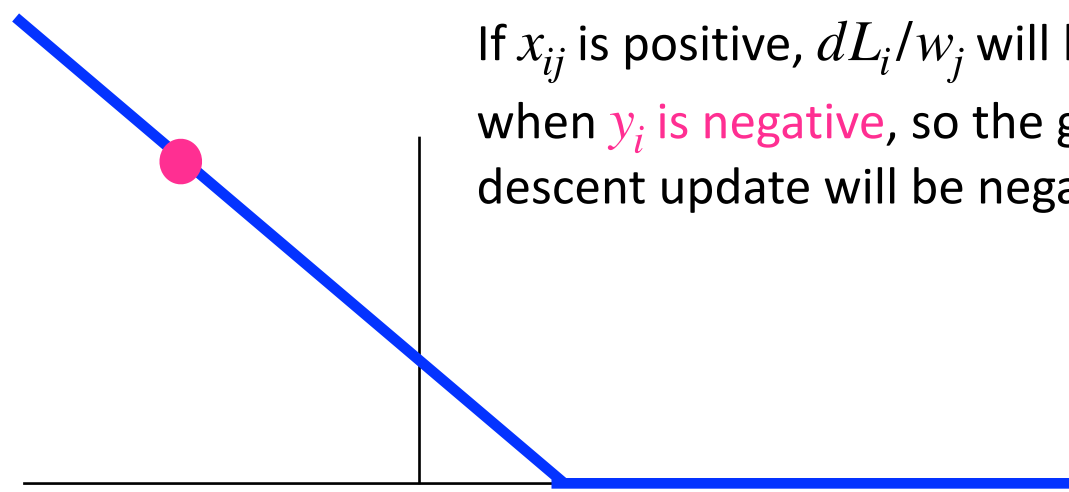
The derivative here is $-y_i x_{ij}$.

If x_{ij} is positive, dL_i/w_j will be negative when y_i is positive, so the gradient descent update will be positive.

Revisiting perceptron

Perceptron has a different loss function:

$$L_i(\mathbf{w}; \mathbf{x}_i) = \begin{cases} 0 & y_i(\mathbf{w}^T\mathbf{x}_i) \geq 0 \\ -y_i(\mathbf{w}^T\mathbf{x}_i) & \text{otherwise} \end{cases}$$

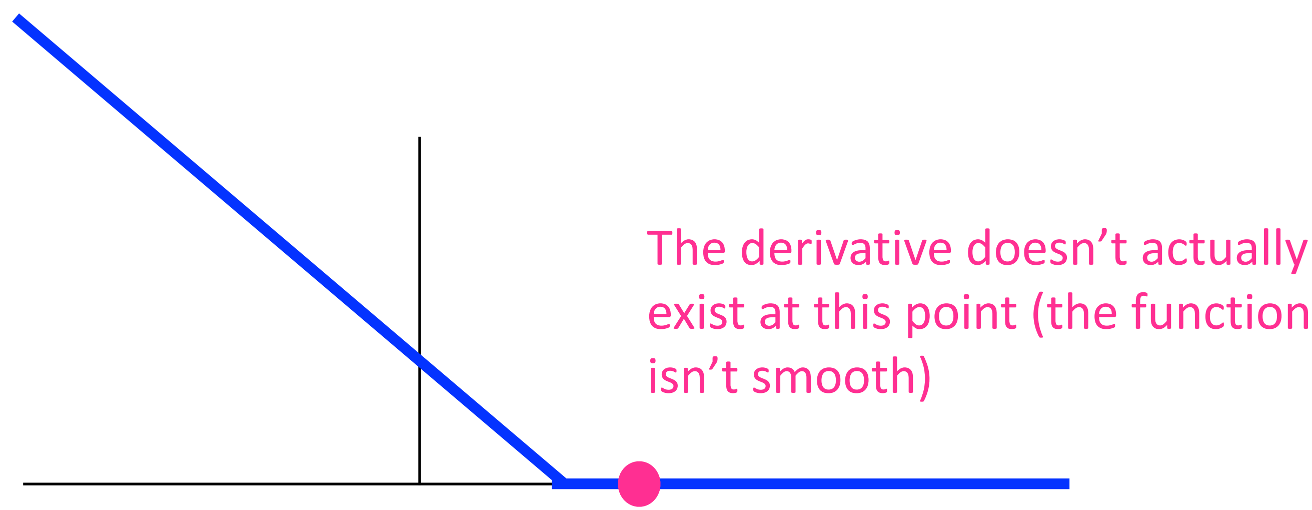


If x_{ij} is positive, dL_i/w_j will be positive when y_i is negative, so the gradient descent update will be negative

Revisiting perceptron

Perceptron has a different loss function:

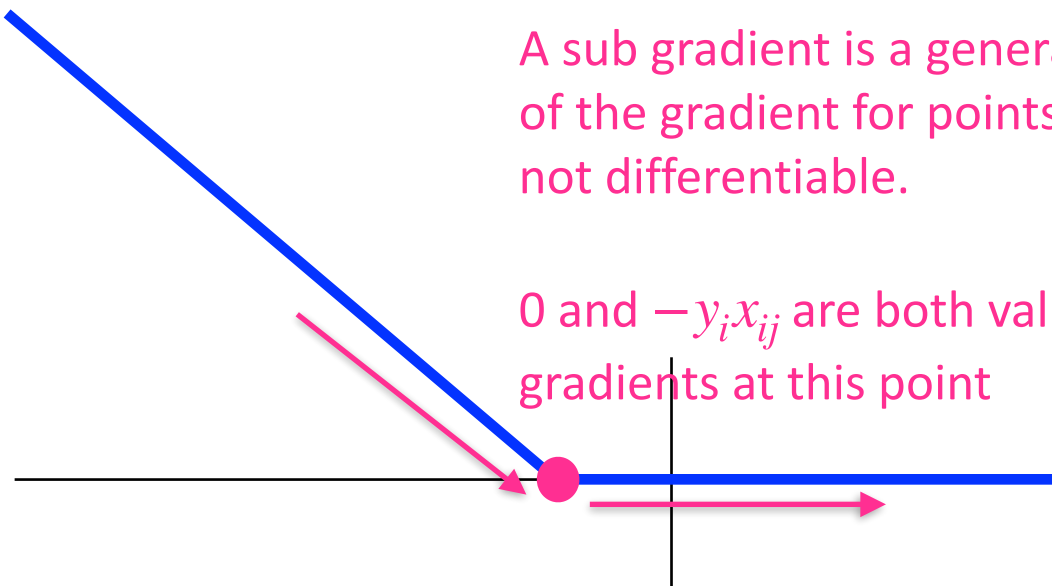
$$L_i(\mathbf{w}; \mathbf{x}_i) = \begin{cases} 0 & y_i(\mathbf{w}^T \mathbf{x}_i) \geq 0 \\ -y_i(\mathbf{w}^T \mathbf{x}_i) & \text{otherwise} \end{cases}$$



Revisiting perceptron

Perceptron has a different loss function:

$$L_i(\mathbf{w}; \mathbf{x}_i) = \begin{cases} 0 & y_i(\mathbf{w}^T\mathbf{x}_i) \geq 0 \\ -y_i(\mathbf{w}^T\mathbf{x}_i) & \text{otherwise} \end{cases}$$



A sub gradient is a generalization of the gradient for points that are not differentiable.

0 and $-y_i x_{ij}$ are both valid sub gradients at this point

Revisiting perceptron

Perceptron has a different loss function:

$$L_i(\mathbf{w}; \mathbf{x}_i) = \begin{cases} 0 & y_i(\mathbf{w}^T\mathbf{x}_i) \geq 0 \\ -y_i(\mathbf{w}^T\mathbf{x}_i) & \text{otherwise} \end{cases}$$

Perceptron is a stochastic gradient descent algorithm using this loss function (and using the sub gradient instead of gradient)