EEP590 Spring 2022

Deep Learning for Embedded Real Time Intelligence

Lecture 2: Introduction to Machine Learning and Optimization

Prof. Richard Shi Department of Electrical and Computer Engineering (cjshi@uw.edu)

with the material adopted from Joseph Redmon and Ali Farhadi at UW CSE

ELECTRICAL & COMPUTER ENGINEERING

Logistics

Join the slack group:
https://join.slack.com/t/eep590spr2022-vqi8492/shared_invite/zt-16pvvou7r-guHZf_WPrwFwMDfiKIREOQ

Assignment 1 will be out this weekend, and is due on April 17th 11:59 pm. You will have 1 week to complete this first assignment.

We expect the future assignments to be more involved, and you'll get 2 weeks each to complete them.

We will follow standard UW late policies, i.e 10% penalty per late day.

Logistics (contd.)

Course Grading (subject to change based on the load):

Class Participation - 10% -> Try to engage in discussions in the class and on slack.

Five Assignments - 50% -> Assignments will be a mix of conceptual + programming problems

Project - 40% -> Assemble the different components you learn through the assignments into a project. Be creative and innovative. More guidelines will be provided a few weeks later. .

Breakdown of Assignments:
Assignment 1 (~1 week) -> Setup some basic libraries, intro to DL frameworks, some basic programming & ML.
Assignment 2 (~2 weeks) - Deep learning operators & constructs
Assignment 3 (~2 weeks) - Making DL more efficient (in terms of latency, memory usage, operations)
Assignment 4 (~2 weeks) - Optimized C++ modules for real-time inference
Assignment 5 (~2 weeks) - Some other misc. Topics depending on the coverage.

What is Deep Learning Anyway?

Typical ML pipeline:

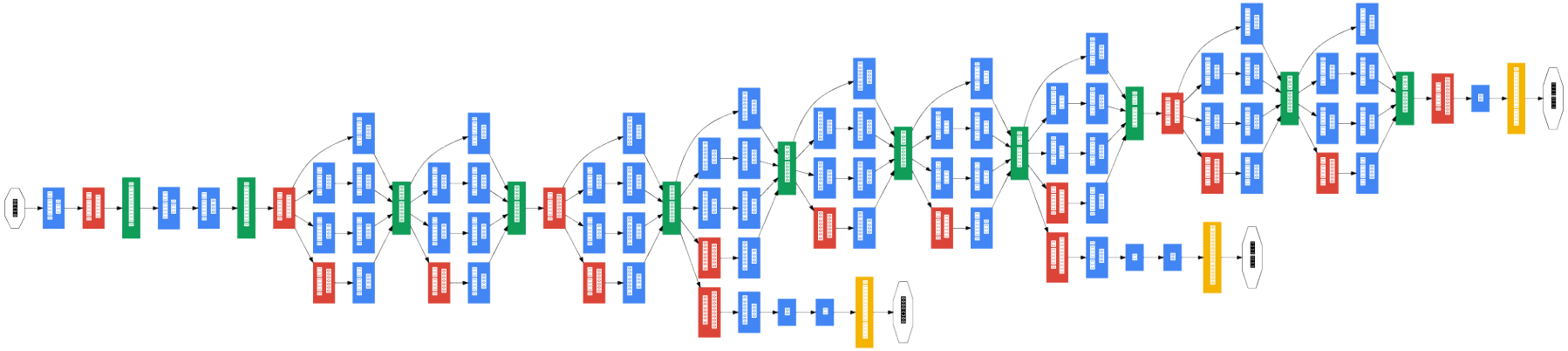
Extract features -> optimize model -> inference

Deep learning:

Optimize the model based on data on GPU -> inference in the cloud or on edges

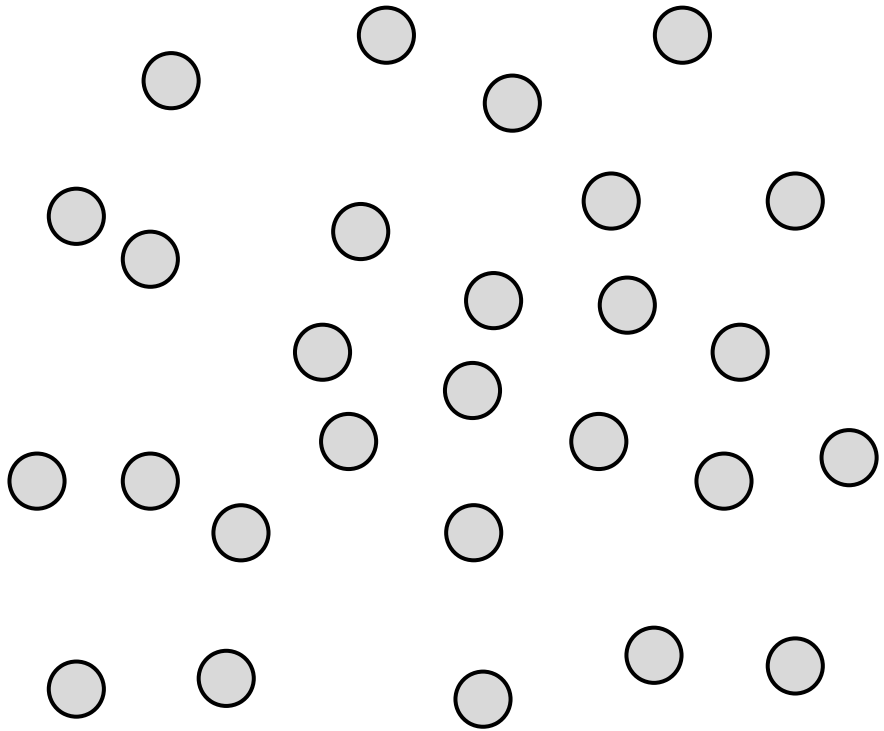
Typically in a *Deep* Learning Network

- Most *feature extraction* is done in the model
- Multiple layers with deeper layers having more semantically meaningful content
- Has figures of architectures that look like:



Mostly Supervised Learning

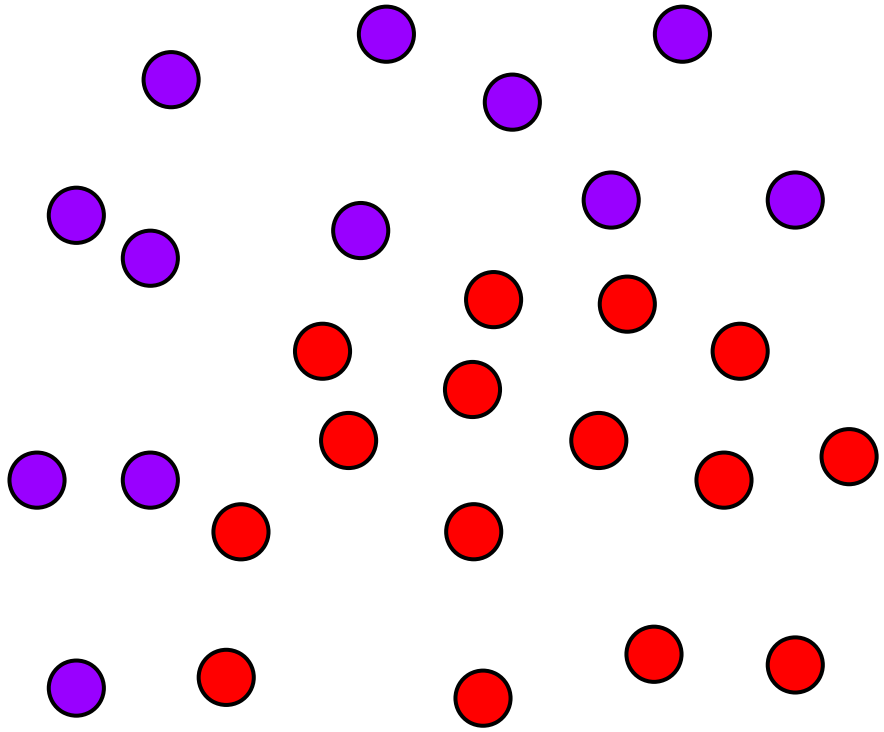
We'll have LOTS of data



Mostly Supervised Learning

We'll have LOTS of data

Our data has labels

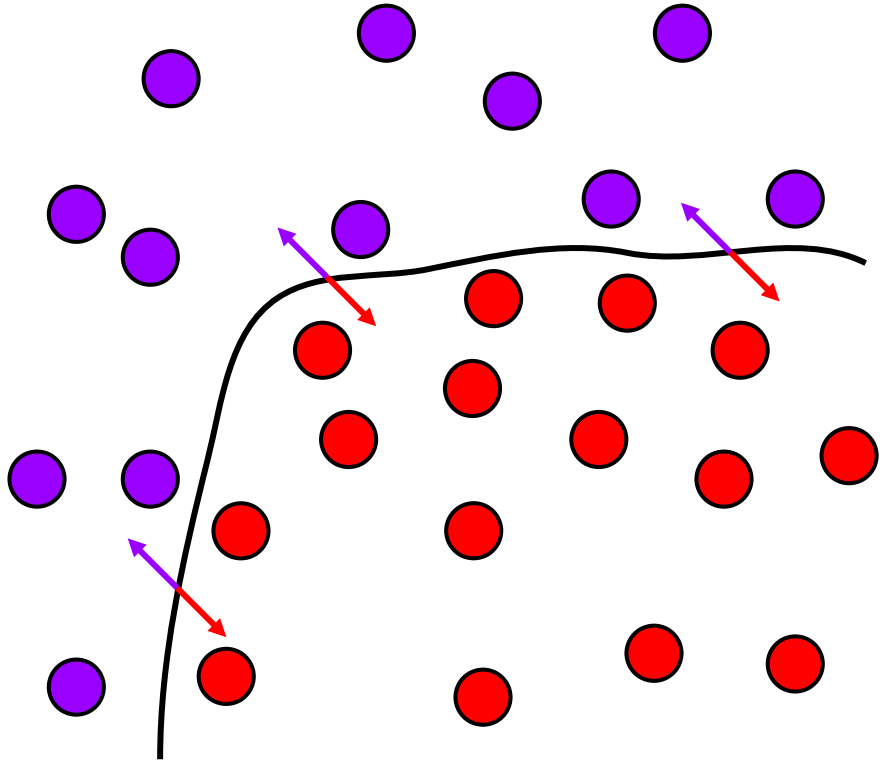


Mostly Supervised Learning

We'll have LOTS of data

Our data has labels

We want to learn a model to predict those labels

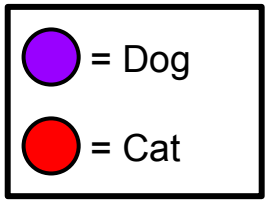


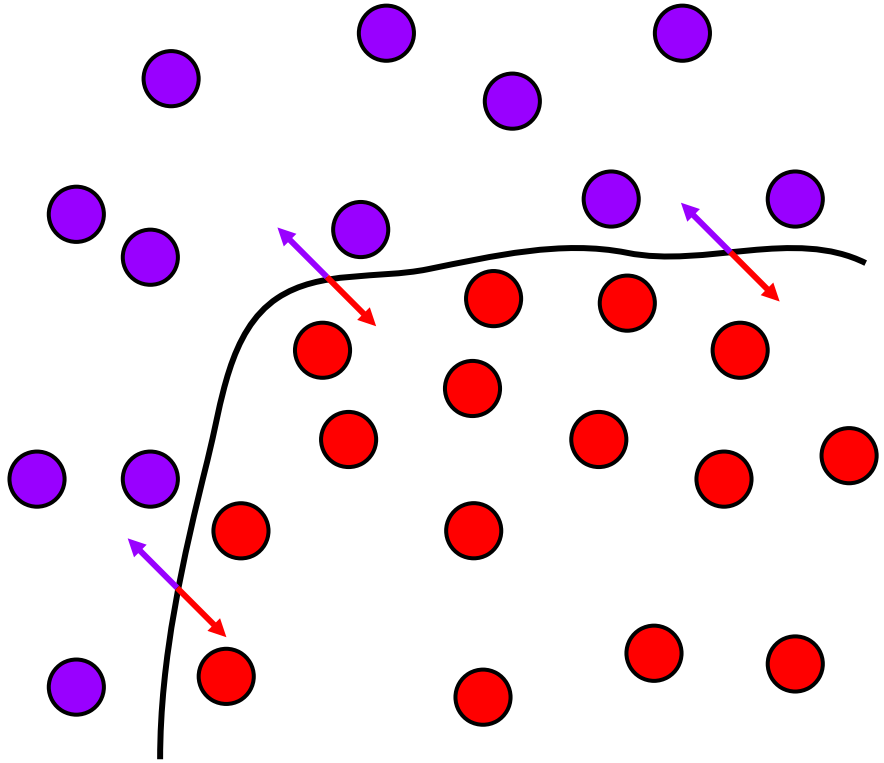
Classification: Class Labeling

Each point belongs to one class

Goal: build a model that separates classes

E.g.: is this an image of a dog or a cat?





Regression: Real-Valued Labels and Functions

Each point has a (or many) real-valued label

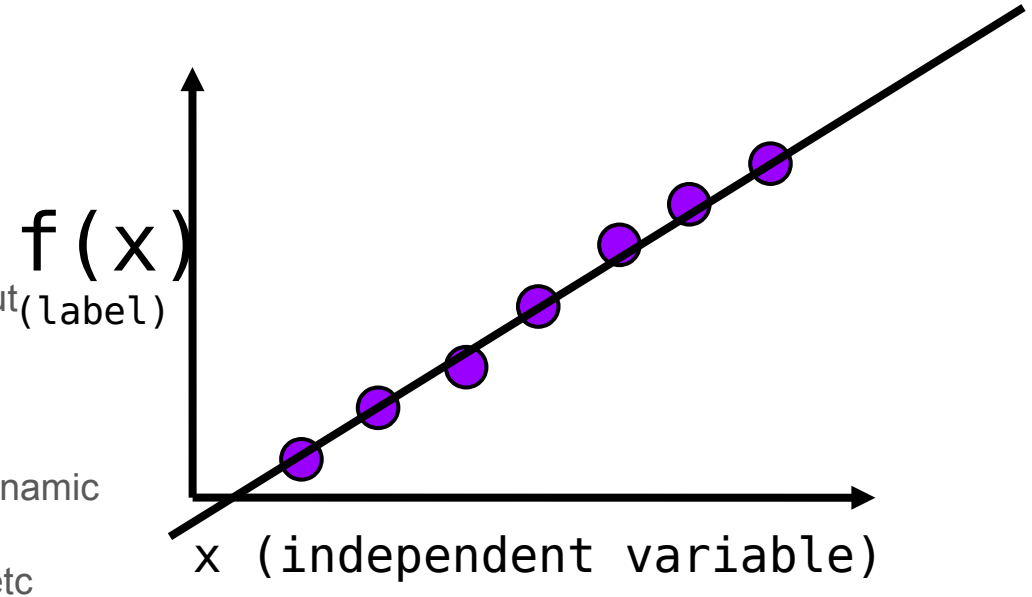
Goal: build model to predict real-values

E.g.: How old is the person in this image?

Static: multiple-input variables/multiple-output variable functions; curve fitting

Dynamic: has time t (changing with time): dynamic

System modeling, identification, estimation etc

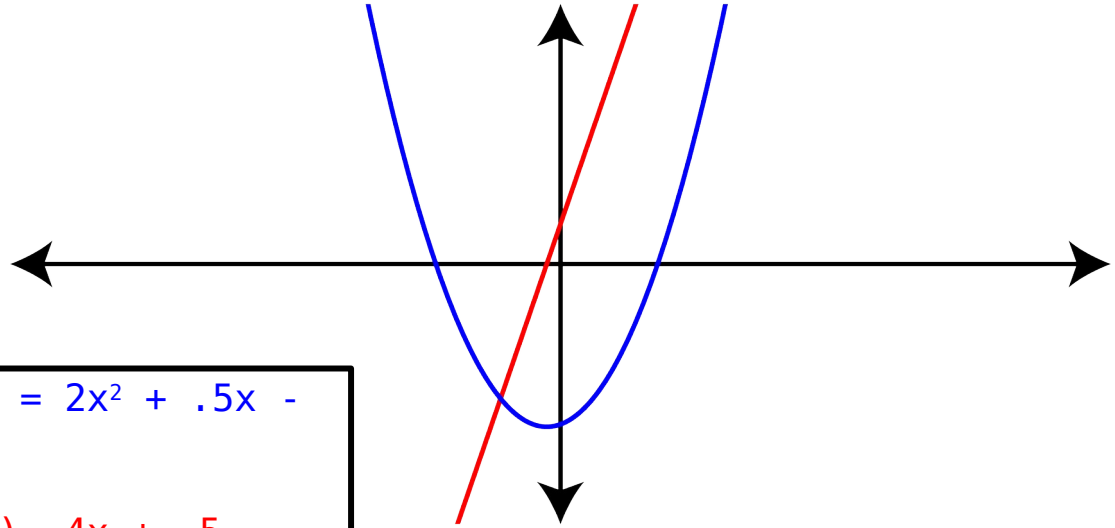


How Do We Get These Models?

Optimization!

Recall: given a function we can find local/global maxs/mins of that function with **calculus**!

Specifically, zero-crossings of the derivative are local extrema



$f(x) = 2x^2 + .5x - 2$

$f'(x) = 4x + .5$

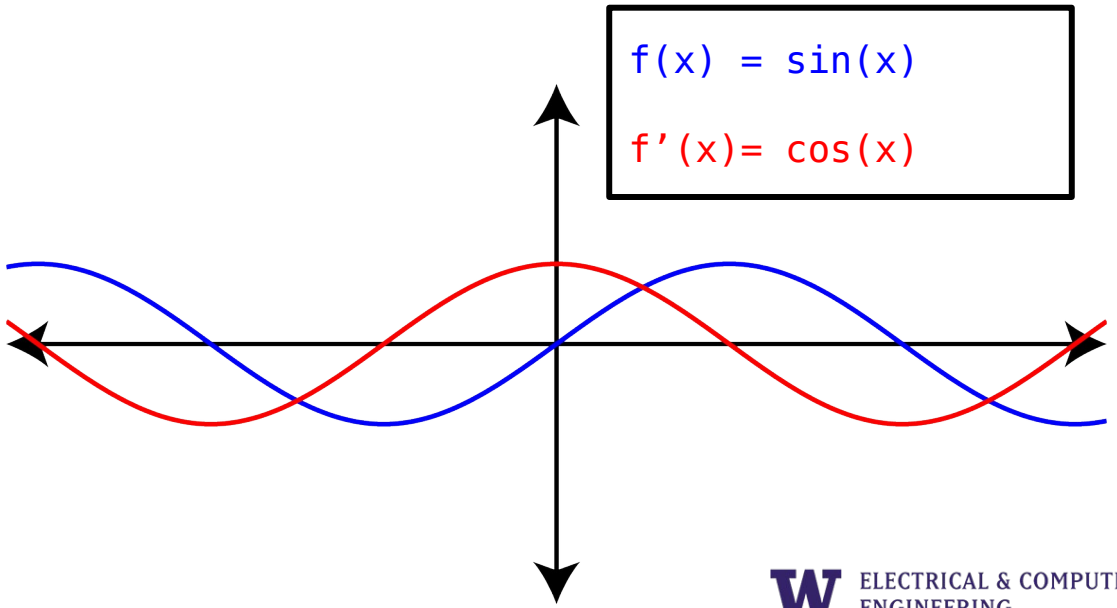
How Do We Get These Models?

Optimization!

Recall: given a function we can find local/global maxs/mins of that function with **calculus**!

Specifically, zero-crossings of the derivative are local extrema

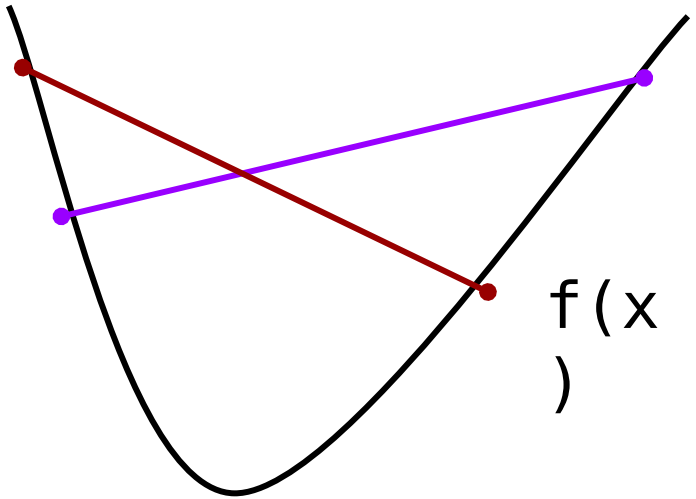
Can be multiple solutions!



Convex Optimization

Convex function: connect any two points on graph with a line, that line lies above function everywhere

Any local extrema is global extrema

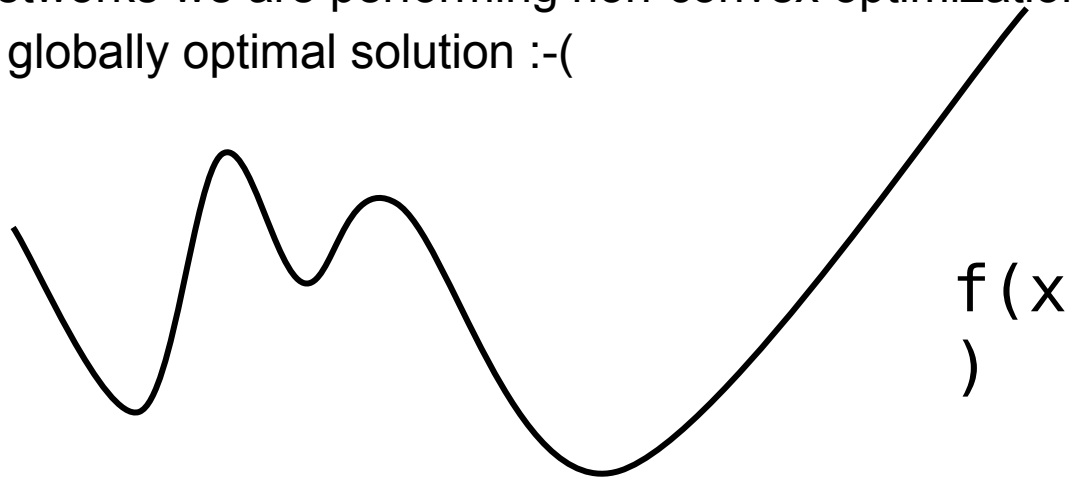


Non-Convex Optimization

Non-convex function: doesn't have those constraints

Extrema may be local or global, don't always know which you have!

With neural networks we are performing non-convex optimization, we aren't guaranteed a globally optimal solution :-(



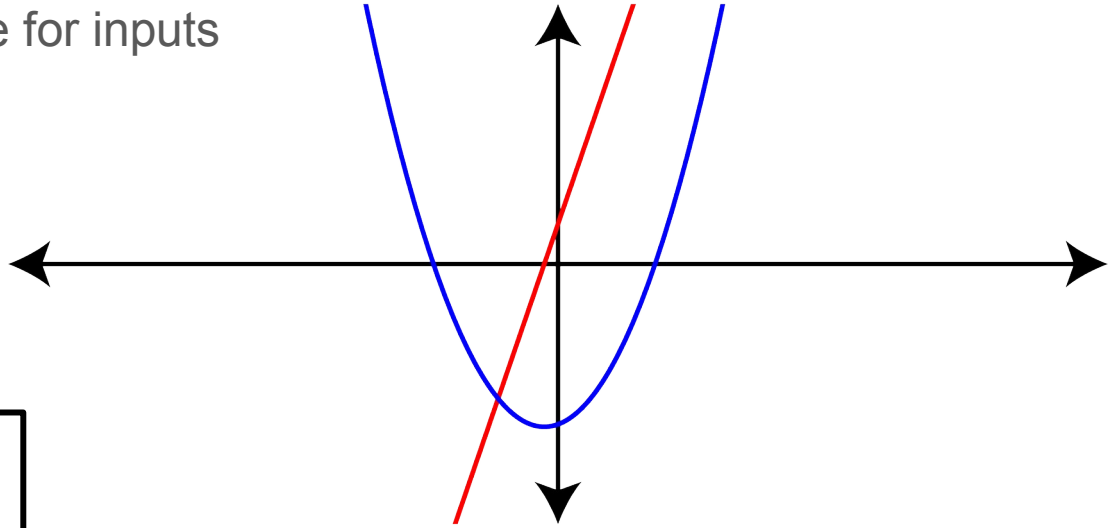
How Do We Optimize?

Sometimes there is a closed form solution

Take derivative, set to 0, solve for inputs

In this case:

$f'(x) = 4x + .5$



$f(x) = 2x^2 + .5x - 2$

$f'(x) = 4x + .5$

How Do We Optimize?

Sometimes there is a closed form solution

Take derivative, set to 0, solve for inputs

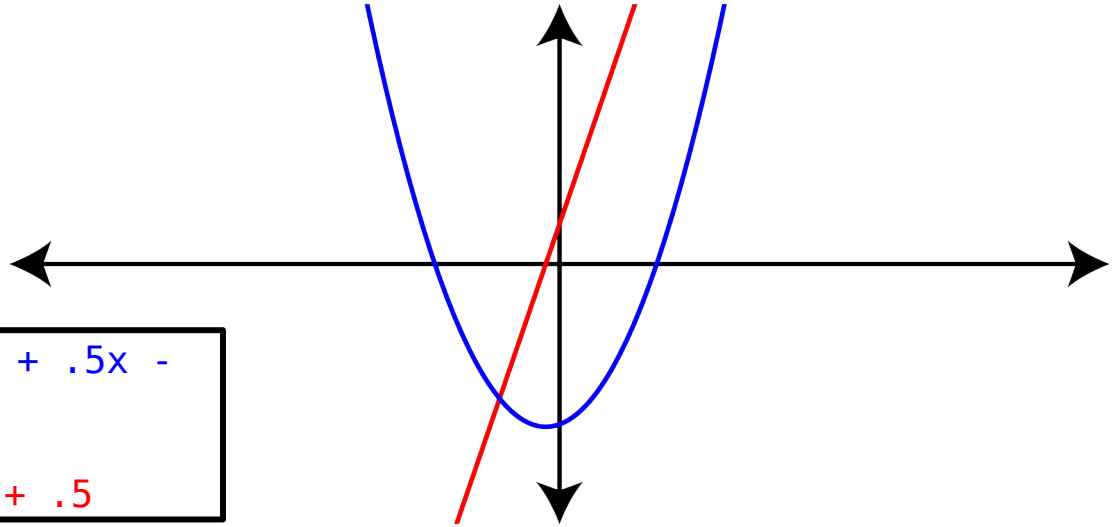
In this case:

$f'(x) = 4x + .5$

$0 = 4x + .5$

$f(x) = 2x^2 + .5x - 2$

$f'(x) = 4x + .5$



How Do We Optimize?

Sometimes there is a closed form solution

Take derivative, set to 0, solve for inputs

In this case:

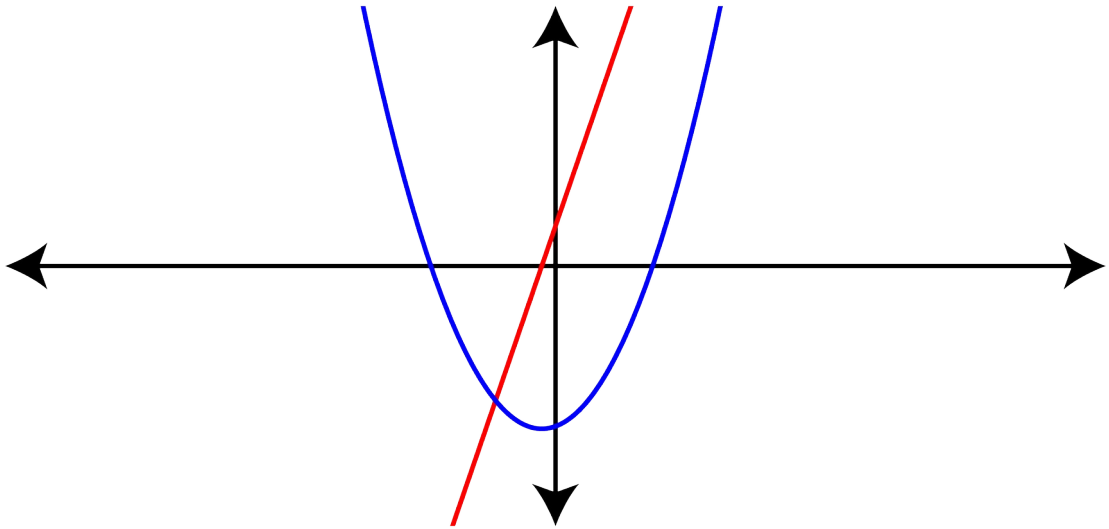
$f'(x) = 4x + .5$

$0 = 4x + .5$

$-4x = .5$

$f(x) = 2x^2 + .5x - 2$

$f'(x) = 4x + .5$



Closed-Form Minimization

Sometimes there is a closed form solution

Take derivative, set to 0, solve for inputs

In this case:

$f'(x) = 4x + .5$

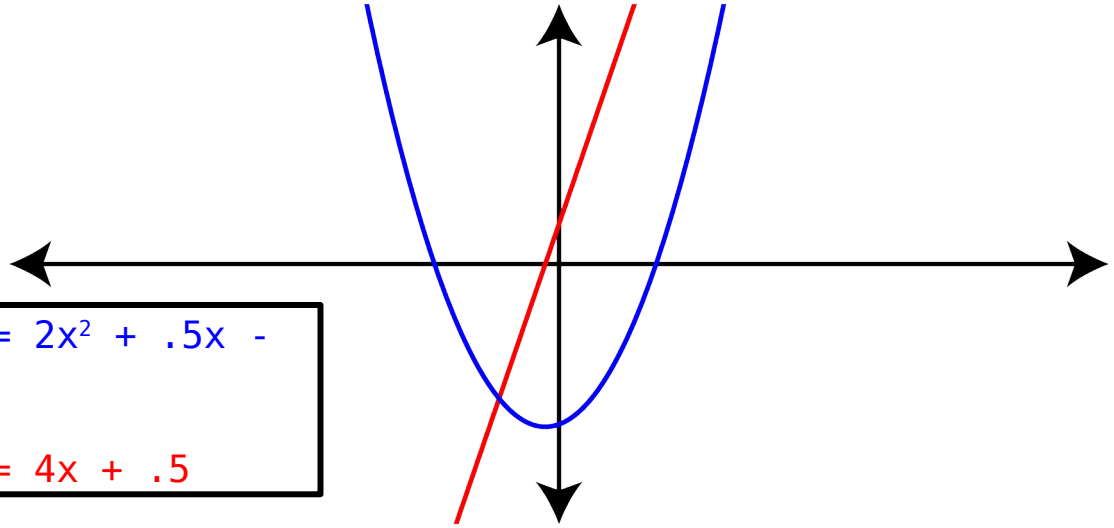
$0 = 4x + .5$

$-4x = .5$

$x = -1/8$

$f(x) = 2x^2 + .5x - 2$

$f'(x) = 4x + .5$

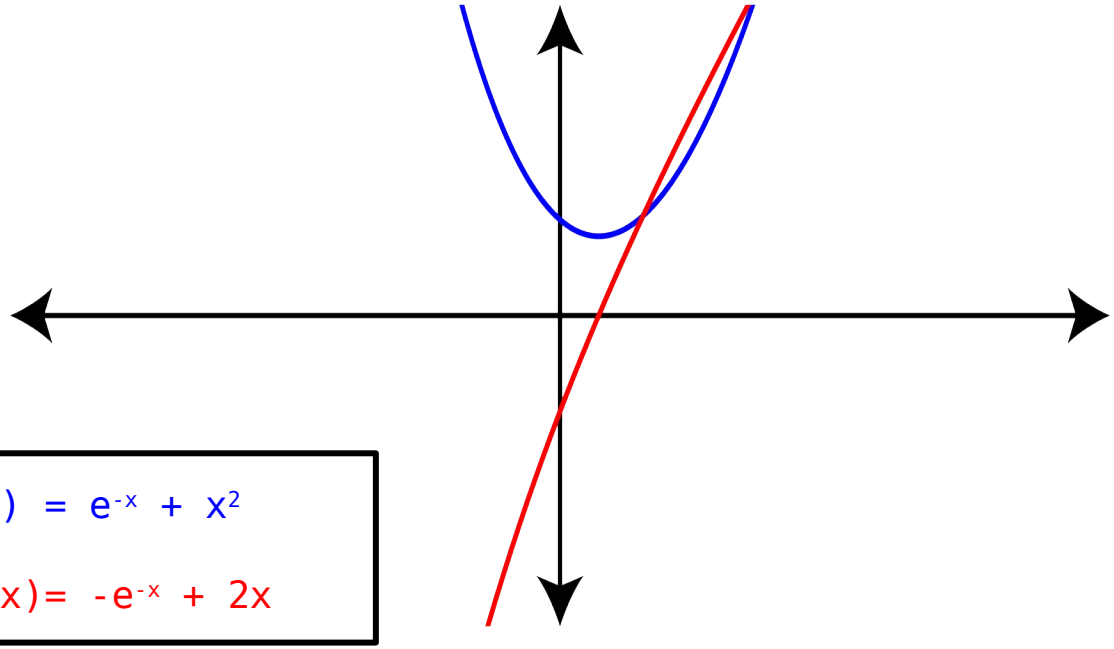


How Do We Optimize?

Sometimes it's harder

In this case:

$f'(x) = -e^{-x} + 2x$



$f(x) = e^{-x} + x^2$

$f'(x) = -e^{-x} + 2x$

How Do We Optimize?

Sometimes it's harder

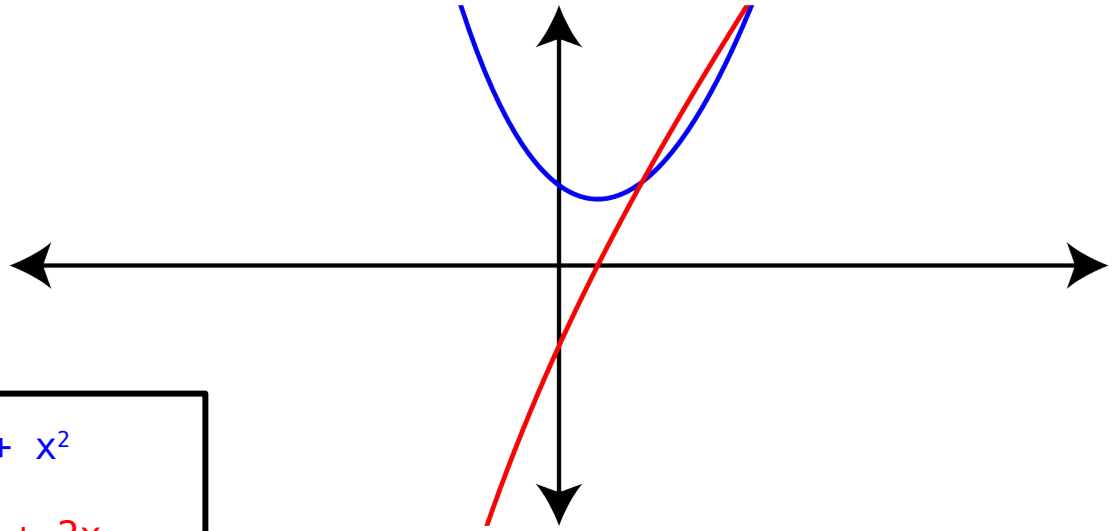
In this case:

$f'(x) = -e^{-x} + 2x$

$0 = -e^{-x} + 2x$

$f(x) = e^{-x} + x^2$

$f'(x) = -e^{-x} + 2x$



How Do We Optimize?

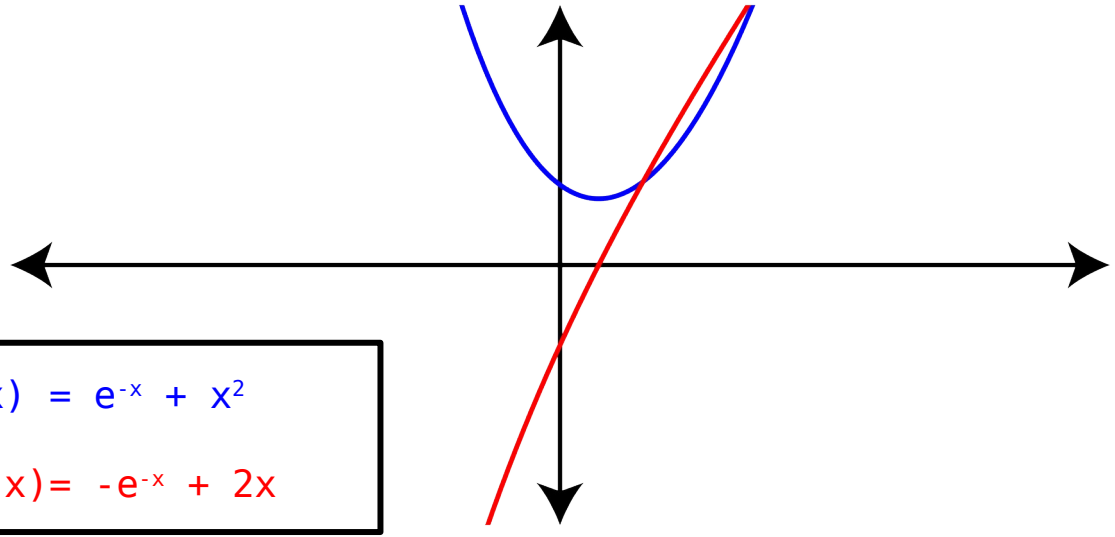
Sometimes it's harder

In this case:

$f'(x) = -e^{-x} + 2x$

$0 = -e^{-x} + 2x$

$e^{-x} = 2x$



$f(x) = e^{-x} + x^2$

$f'(x) = -e^{-x} + 2x$

How Do We Optimize?

Sometimes it's harder

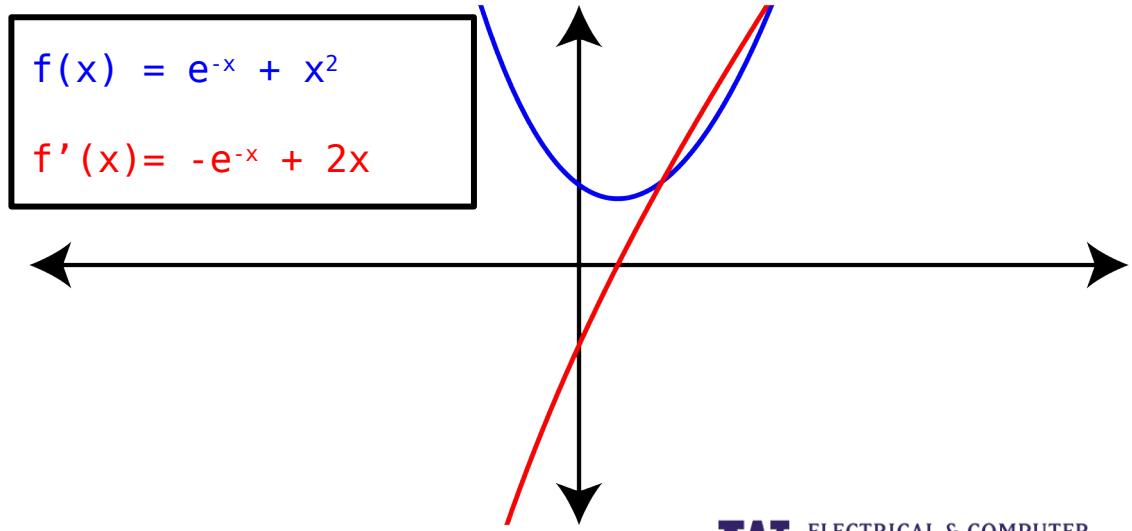
In this case:

$f'(x) = -e^{-x} + 2x$

$0 = -e^{-x} + 2x$

$e^{-x} = 2x$

Now what??



How Do We Optimize?

Sometimes it's harder

In this case:

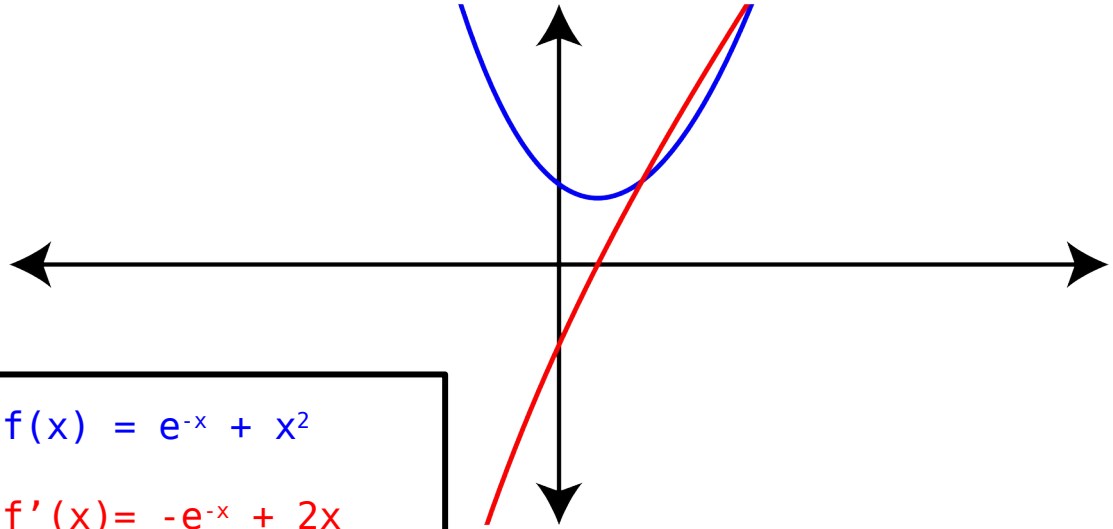
$f'(x) = -e^{-x} + 2x$

$0 = -e^{-x} + 2x$

$e^{-x} = 2x$

Now what??

Transcendental, no closed-form solution! Uh-oh....



$f(x) = e^{-x} + x^2$

$f'(x) = -e^{-x} + 2x$

How Do We Optimize?

Sometimes it's harder

In this case:

$f'(x) = -e^{-x} + 2x$

$0 = -e^{-x} + 2x$

$e^{-x} = 2x$

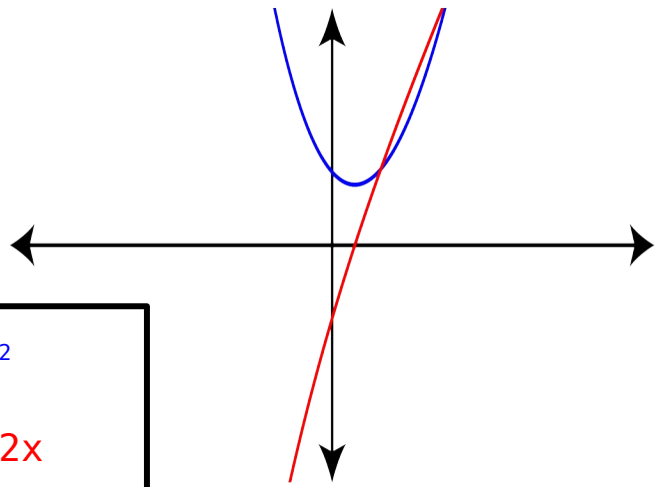
Now what??

Transcendental, no closed-form solution! Uh-oh....

But there is a min! x ~ .351734...

$f(x) = e^{-x} + x^2$

$f'(x) = -e^{-x} + 2x$



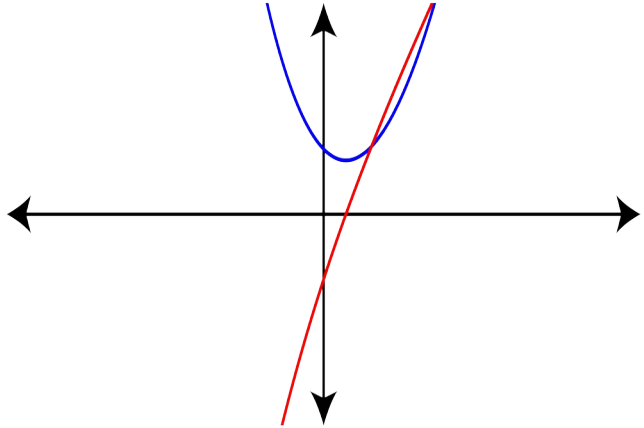
Gradient Descent

Gradient points away from the minimum!

df/dx is positive when ↑ x = ↑ f(x)

df/dx is negative when ↑ x = ↓ f(x)

To minimize f(x), when gradient is positive make x smaller, when negative make x larger!



$f(x) = e^{-x} + x^2$

$f'(x) = -e^{-x} + 2x$

Gradient Descent

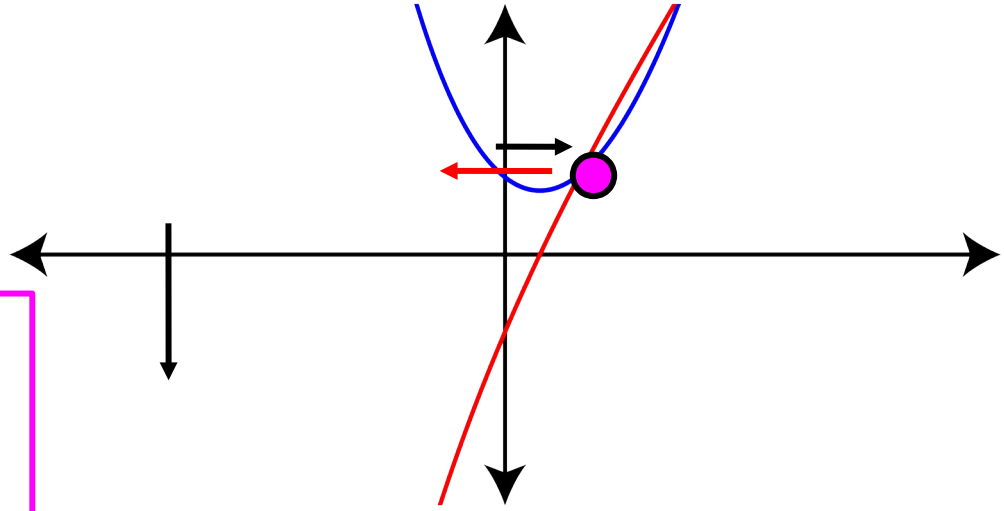
To minimize f(x), when gradient is positive make x smaller, when negative make x larger!

$f(x) = e^{-x} + x^2$

$f'(x) = -e^{-x} + 2x$

$f(-.5) = 1.8987$
$f'(-.5) = -2.64872$

Gradient is neg, make x bigger!

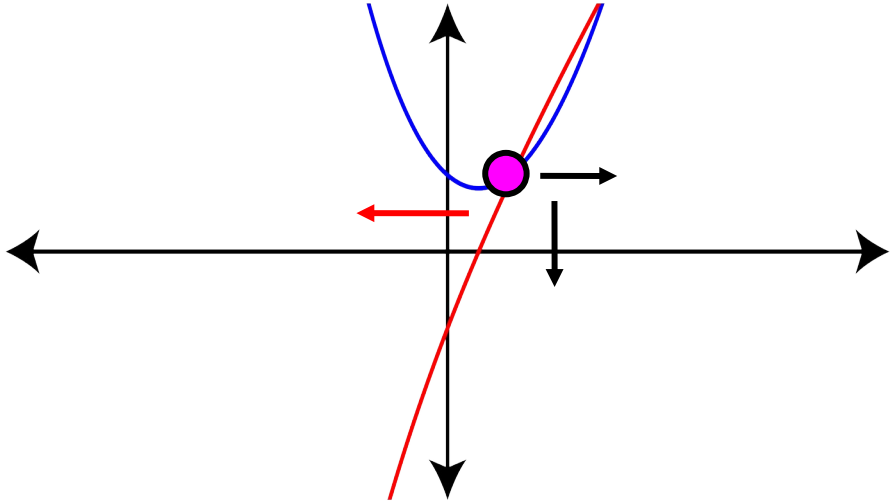


Gradient Descent

To minimize f(x), when gradient is positive make x smaller, when negative make x larger!

$f(x) = e^{-x} + x^2$

$f'(x) = -e^{-x} + 2x$



f (-.25) = 1.34653
f'(-.25)= -1.78403

Gradient is neg, make x bigger!

Gradient Descent

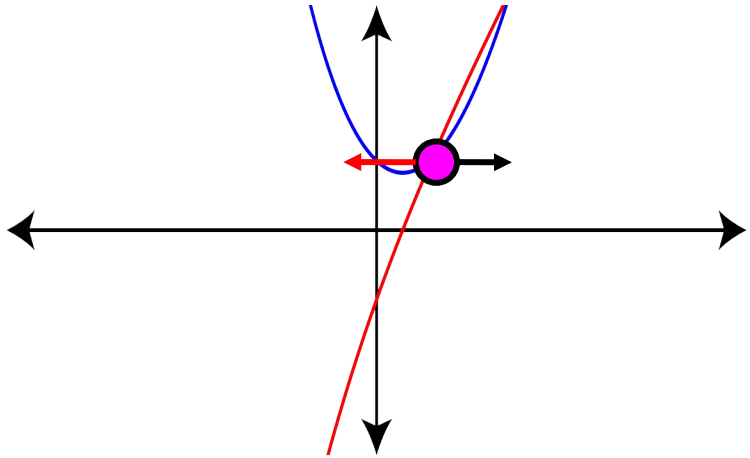
To minimize f(x), when gradient is positive make x smaller, when negative make x larger!

$f(x) = e^{-x} + x^2$

$f'(x) = -e^{-x} + 2x$

$f(0) = 1$
$f'(0) = -1$

Gradient is neg, make x bigger!



Gradient Descent

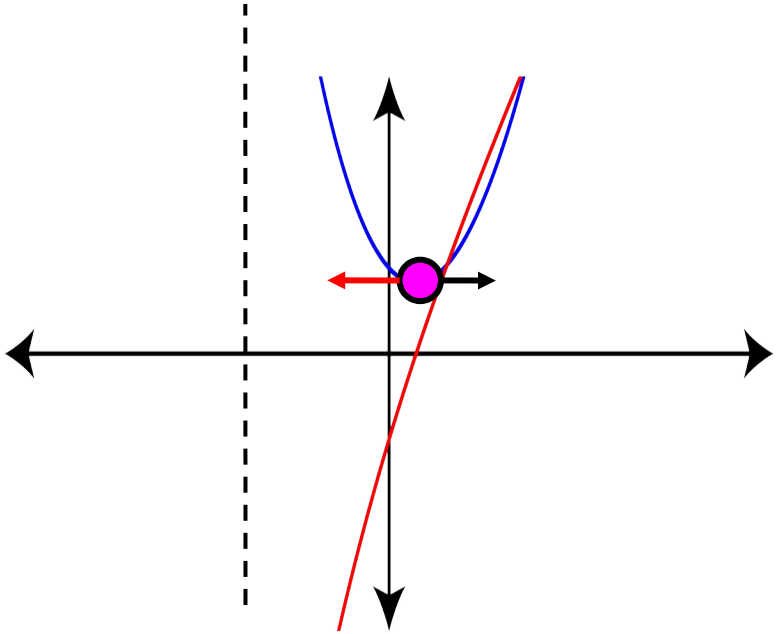
To minimize f(x), when gradient is positive make x smaller, when negative make x larger!

$f(x) = e^{-x} + x^2$

$f'(x) = -e^{-x} + 2x$

$f(.25) = .841301$
$f'(.25) = -.278801$

Gradient is neg, make x bigger!

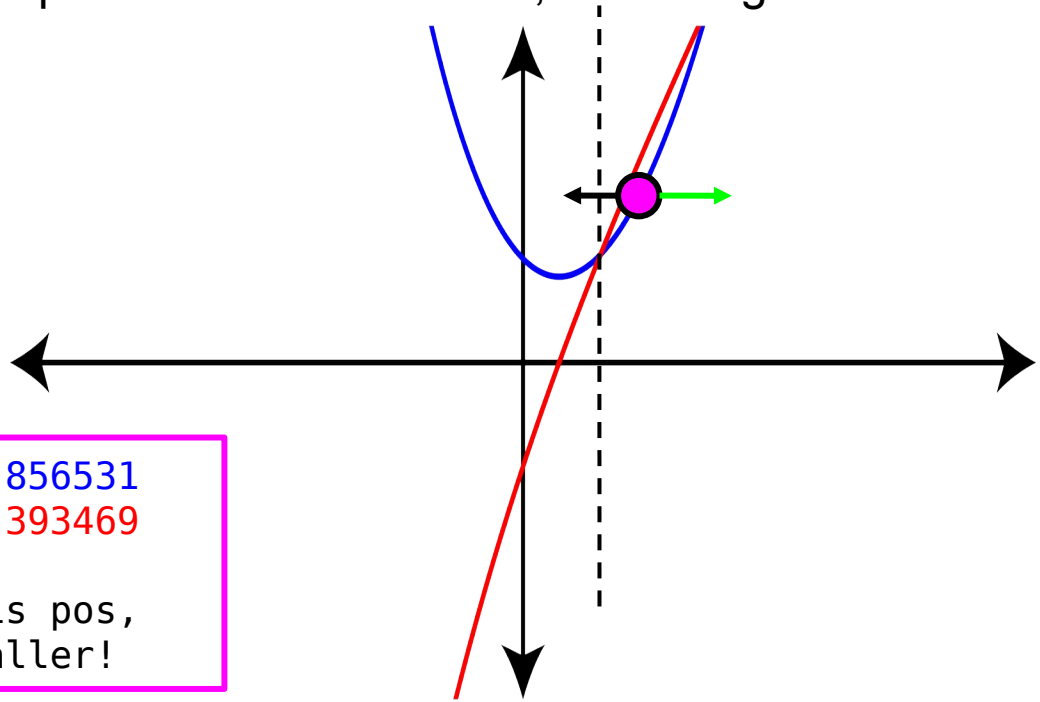


Gradient Descent

To minimize f(x), when gradient is positive make x smaller, when negative make x larger!

$f(x) = e^{-x} + x^2$

$f'(x) = -e^{-x} + 2x$



$f(.5) = .856531$
$f'(.5) = .393469$

Gradient is pos, make x smaller!

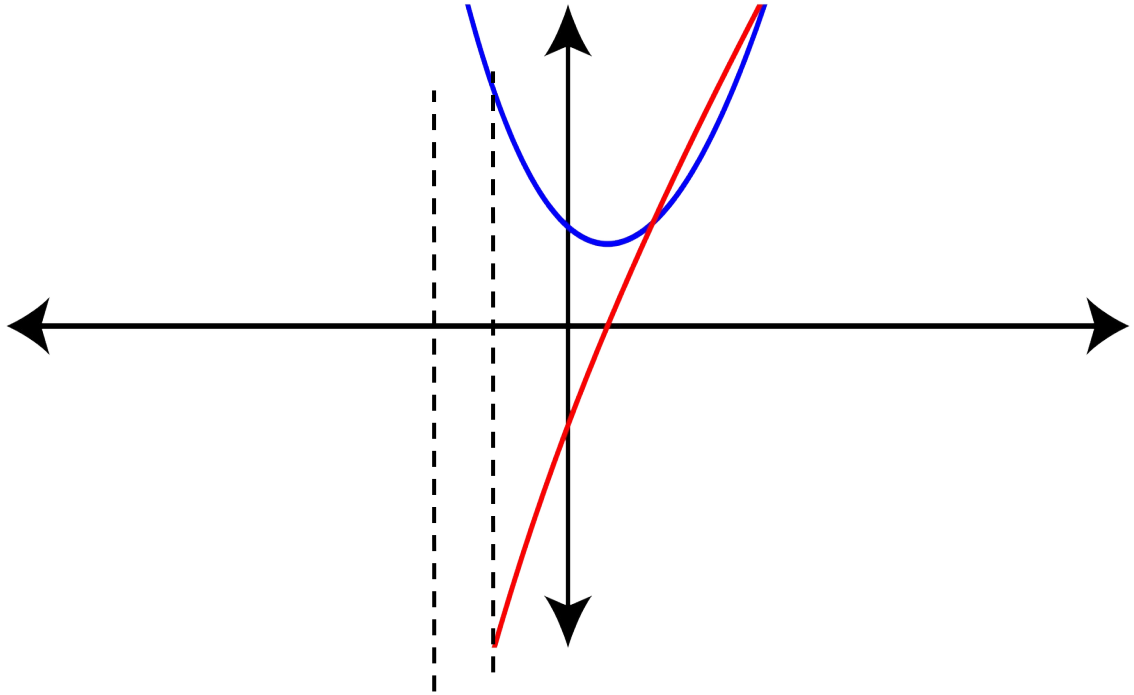
Gradient Descent

To minimize f(x), when gradient is positive make x smaller, when negative make x larger!

$f(x) = e^{-x} + x^2$

$f'(x) = -e^{-x} + 2x$

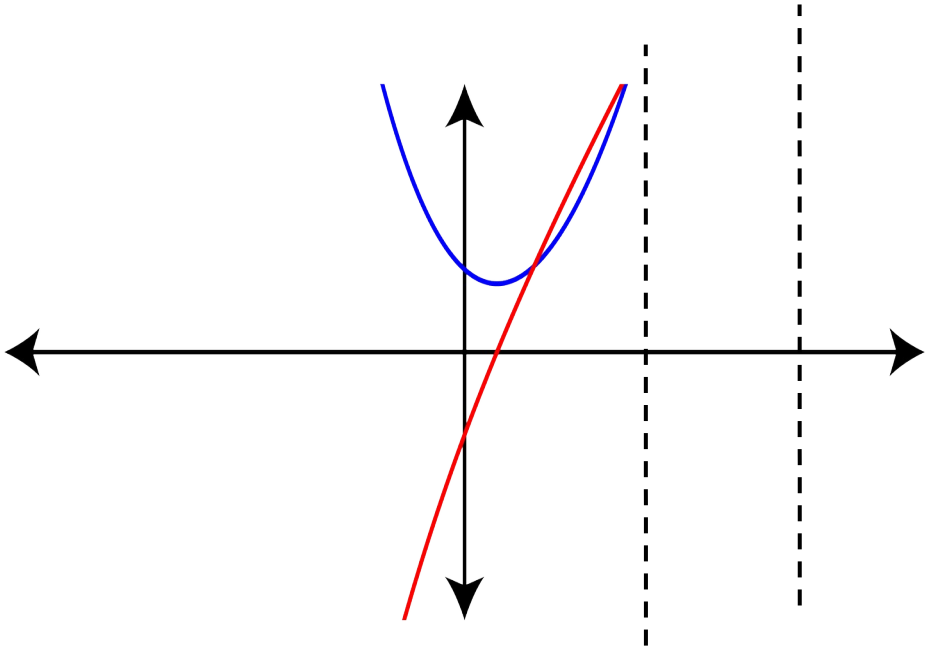
Know min is between 0.25 and 0.5, how can we get more exact??



Gradient Descent

To minimize f(x), when gradient is positive make x smaller, when negative make x larger!

We were moving around by .25 every time, instead we could move based off the gradients

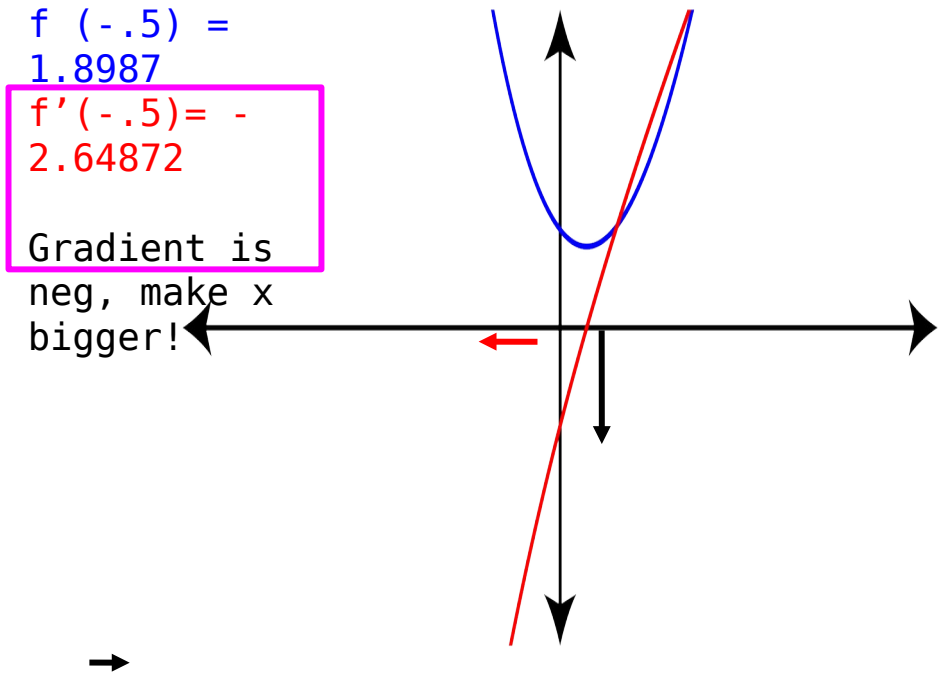


Gradient Descent

To minimize f(x), when gradient is positive make x smaller, when negative make x larger!

We were moving around by .25 every time, instead we could move based off the gradients

At -.5, gradient mag is large, move a lot!

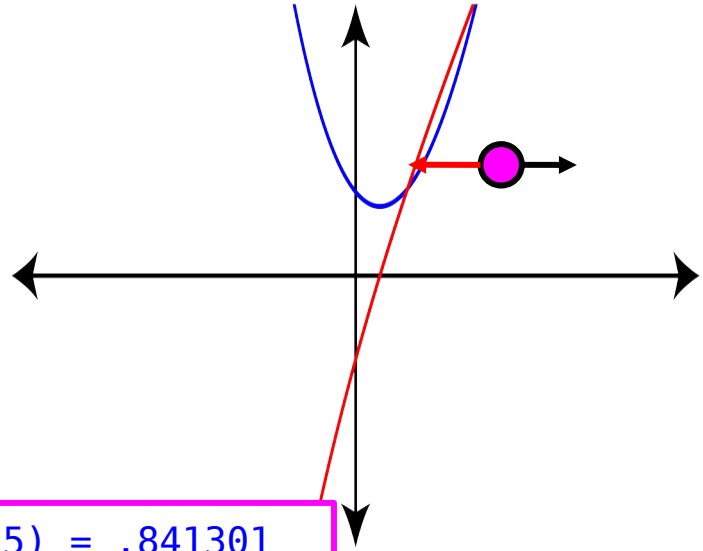


Gradient Descent

To minimize f(x), when gradient is positive make x smaller, when negative make x larger!

We were moving around by .25 every time, instead we could move based off the gradients

At .25, gradient mag is small, move a little!



```
f (.25) = .841301
f'(.25)= -.278801

Gradient is neg,
make x bigger!
```

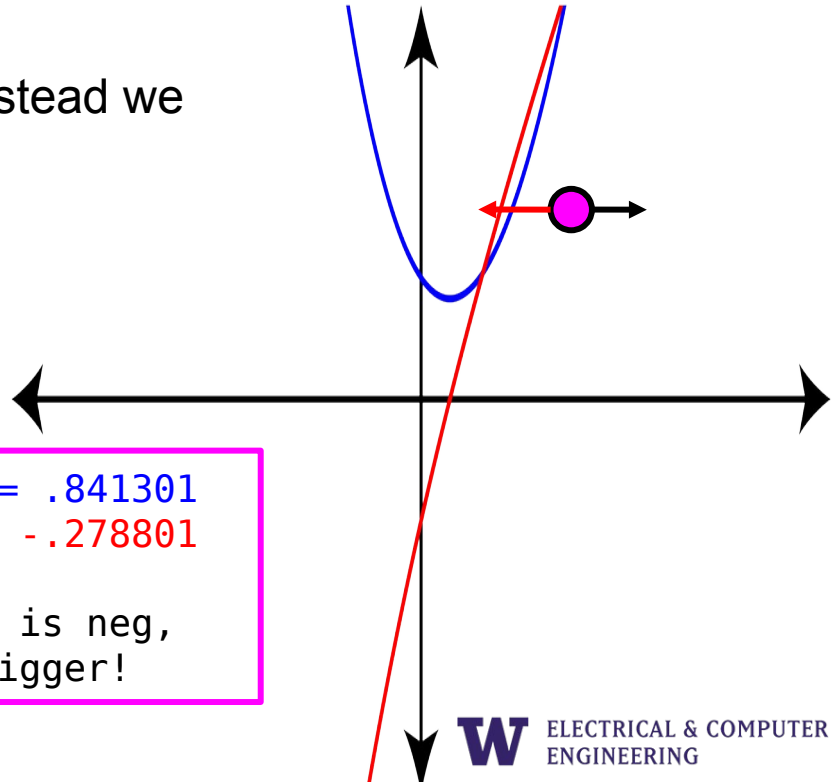
Gradient Descent

To minimize f(x), when gradient is positive make x smaller, when negative make x larger!

We were moving around by .25 every time, instead we could move based off the gradients

At .25, gradient mag is small, move a little!

Move by gradient?
$x = x - \nabla f$



```
f (.25) = .841301
f'(.25)= -.278801

Gradient is neg,
make x bigger!
```

Gradient Descent

To minimize f(x), when gradient is positive make x smaller, when negative make x larger!

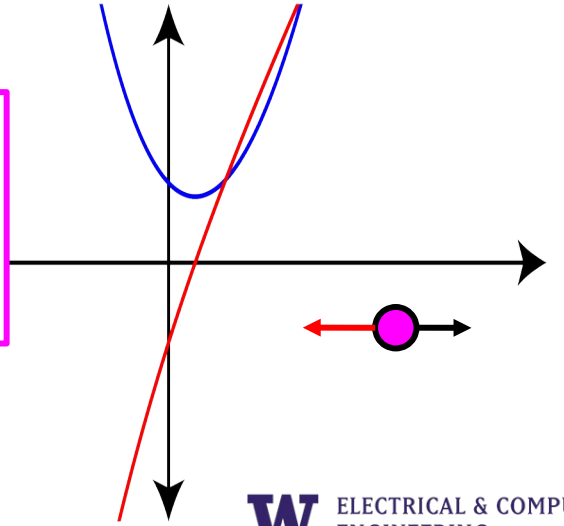
We were moving around by .25 every time, instead we could move based off the gradients

At .25, gradient mag is small, move a little!

Move by scaled gradient
$x = x - \eta \nabla f$

```
f (.25) = .841301
f'(.25)= -.278801

Gradient is neg,
make x bigger!
```



Gradient Descent Algorithm

To find $\text{argmin}_x f(x)$

Initialize x somehow

Gradient Descent Algorithm

To find $\text{argmin}_x f(x)$

Initialize x somehow
Until converged:
    Computer gradient $\nabla f(x)$

Gradient Descent Algorithm

To find $\text{argmin}_x f(x)$

Initialize x somehow

Until converged:

Computer gradient $\nabla f(x)$

$x = x - \eta \nabla f$

Gradient Descent Algorithm

To find $\text{argmin}_x f(x)$

Initialize x somehow
Until converged:
    Computer gradient $\nabla f(x)$
    $x = x - \eta \nabla f$

η is *learning rate*

Gradient Descent Algorithm

To find $\text{argmin}_x f(x)$

Initialize x somehow (how?)
Until converged (when?):
    Computer gradient $\nabla f(x)$
    $x = x - \eta \nabla f$

η is *learning rate* (how do we pick?)

What Does this Have to Do with ML?

Remember, we wanted to optimize our models to fit the data. First we need a measure of “goodness-of-fit”:

Likelihood function - how likely our model thinks our data is

Loss function - how wrong is our model

Want to find parameters that maximize likelihood or minimize loss!

Case Study: Linear Regression

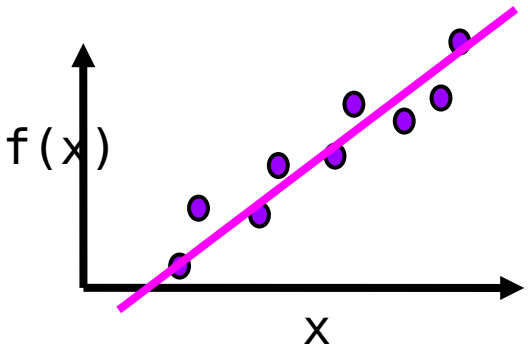
Model: $f^*(x) = ax + b$

Loss function: Sum squared error

$L(f^*) = \Sigma_i \, ||f(x_i) - f^*(x_i)||^2$

Penalizes larger errors more than small errors

Optimization: want to find $\text{argmin}_{a,b} L(f^*)$



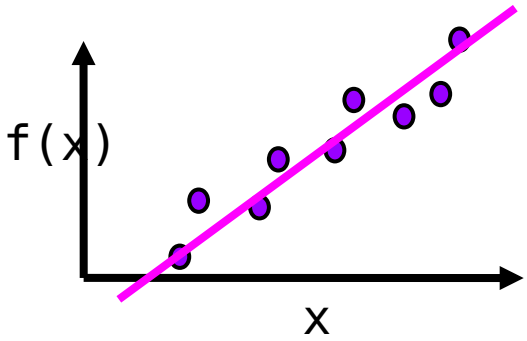
Case Study: Linear Regression

Model: $f^*(x) = ax + b$

Loss function: Sum squared error

$L(f^*) = \Sigma_i \, ||f(x_i) - f^*(x_i)||^2$

Penalizes larger errors more than small errors



Optimization: want to find $\text{argmin}_{a,b} L(f^*)$

$d/da \, L(f^*) = \Sigma_i \, 2[f(x_i) - f^*(x_i)]^*-x$

$d/db \, L(f^*) = \Sigma_i \, 2[f(x_i) - f^*(x_i)]$

Case Study: Linear Regression

Optimization: want to find $\text{argmin}_{a,b} L(f^*)$

$d/da\ L(f^*) = \Sigma_i\ 2[f(x_i) - f^*(x_i)]^*\text{-}x$

$d/db\ L(f^*) = \Sigma_i\ 2[f(x_i) - f^*(x_i)]^*\text{-}1$

How do we change b?

Case Study: Linear Regression

Optimization: want to find $\text{argmin}_{a,b} L(f^*)$

$d/da\ L(f^*) = \Sigma_i\ 2[f(x_i) - f^*(x_i)]^*\text{-}x$

$d/db\ L(f^*) = \Sigma_i\ 2[f(x_i) - f^*(x_i)]^*\text{-}1$

How do we change b?

Sum over differences between truth and prediction, if it is positive, make b larger

Case Study: Linear Regression

Optimization: want to find $\text{argmin}_{a,b} L(f^*)$

$d/da\ L(f^*) = \Sigma_i\ 2[f(x_i) - f^*(x_i)]^*\text{-}x$

$d/db\ L(f^*) = \Sigma_i\ 2[f(x_i) - f^*(x_i)]^*\text{-}1$

How do we change b? Sum over differences between truth and prediction, if it is positive, make b larger

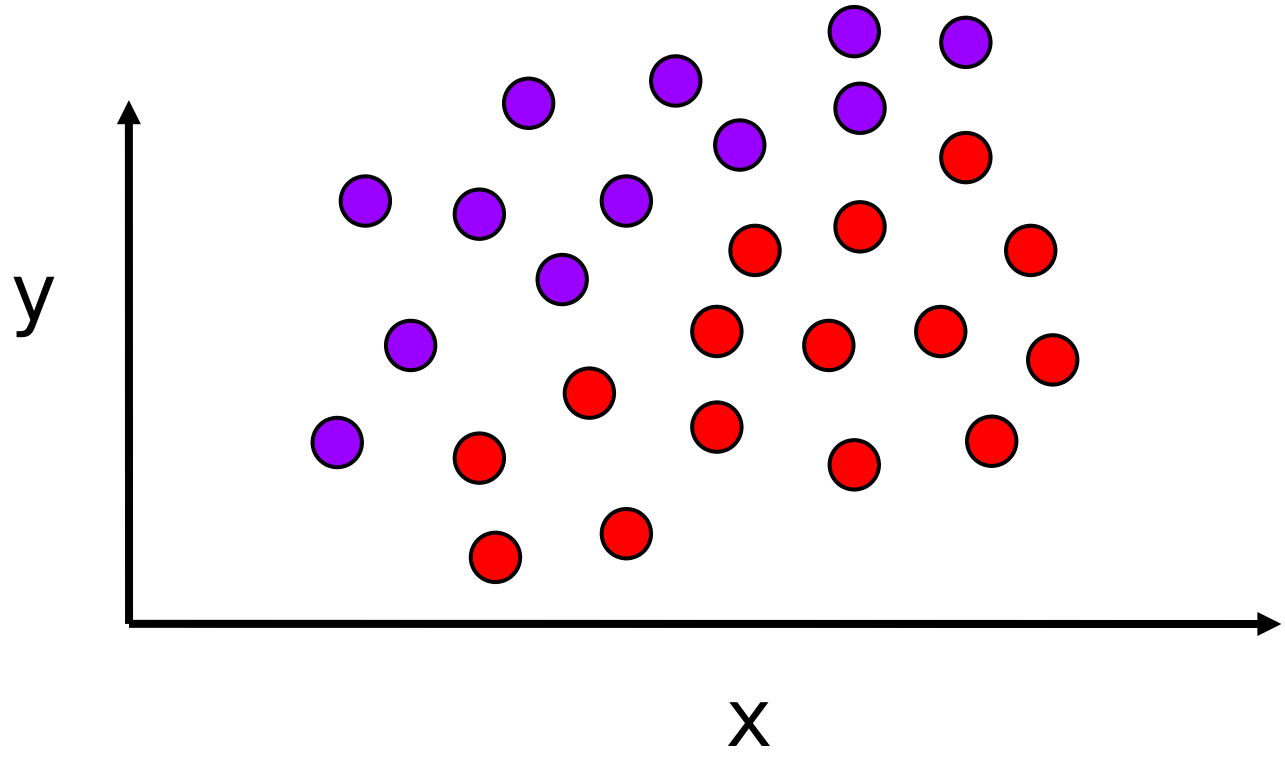
This makes sense, our model is predicting too small for most data, increase the bias!

Loss Functions

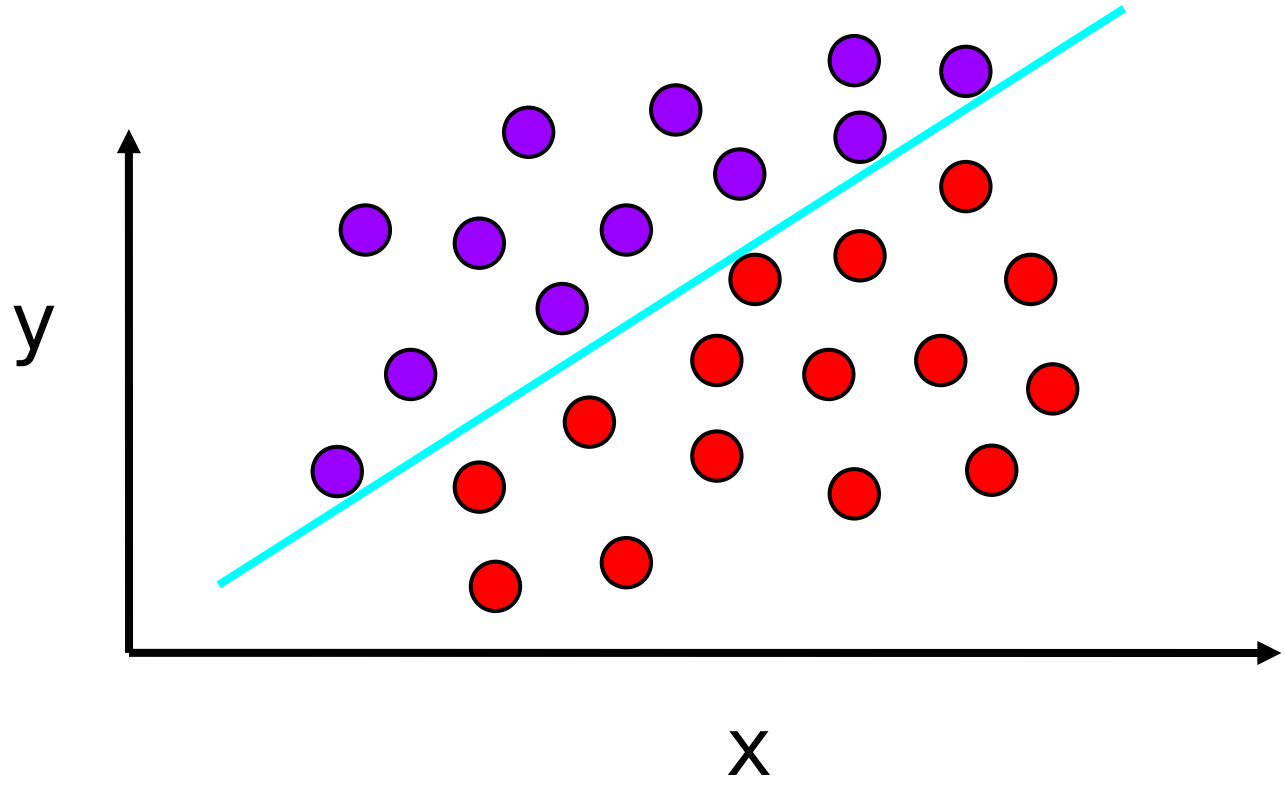
Think of loss functions as functions have the data as givens and take as input the parameters to the model (i.e. model weights, or a and b in linear regression)

Then we want to optimize these parameters to minimize the loss function, which we can do with gradient descent! (or other methods)

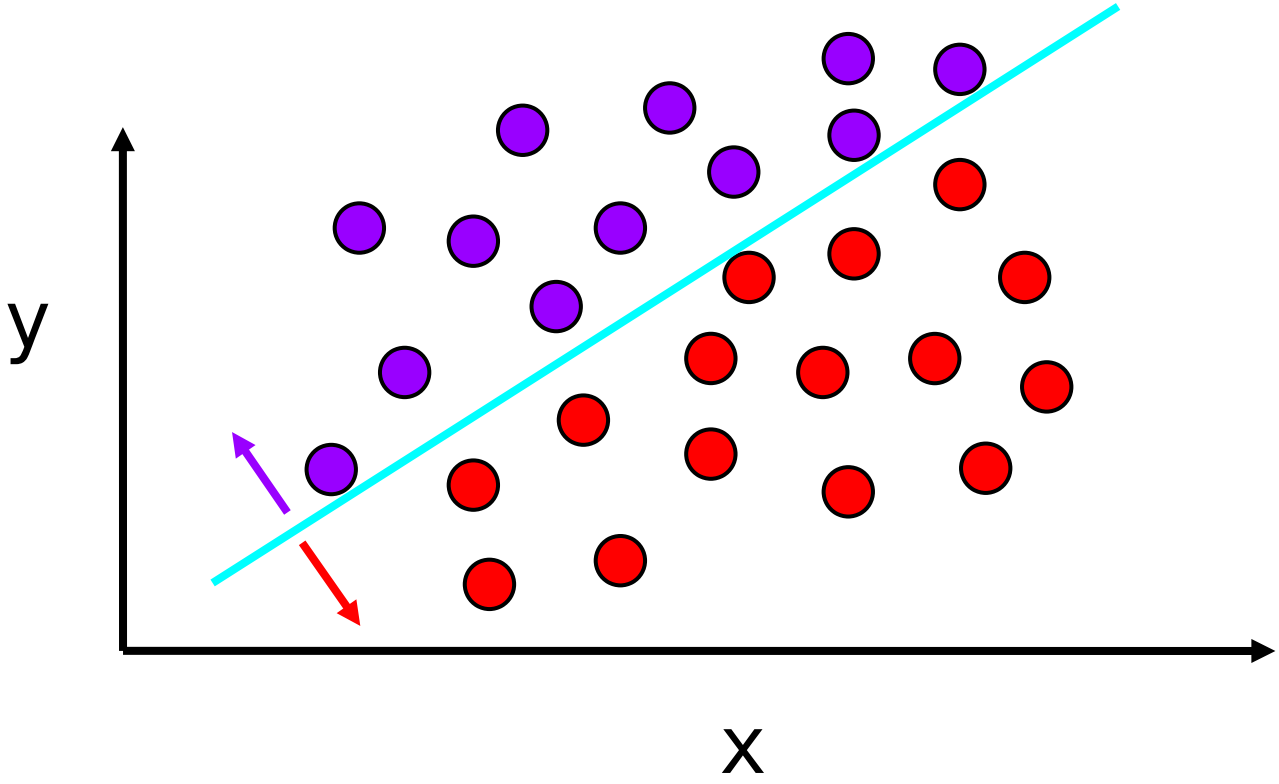
Classification in Two Dimensions



Classification in Two Dimensions

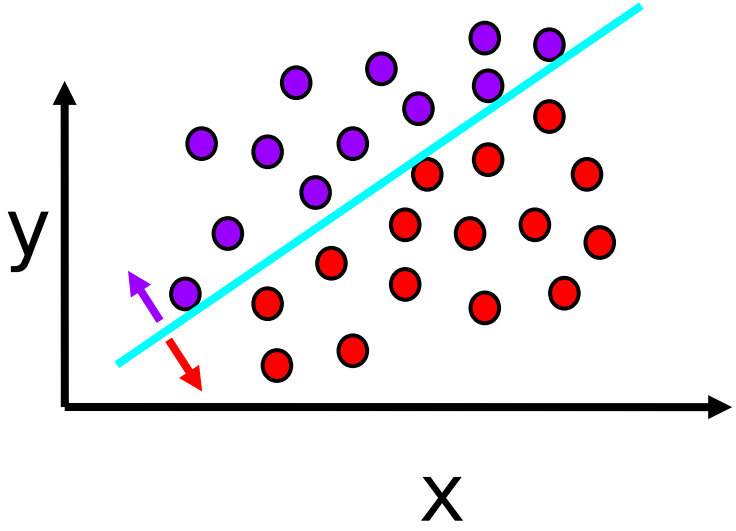


Classification in Two Dimensions



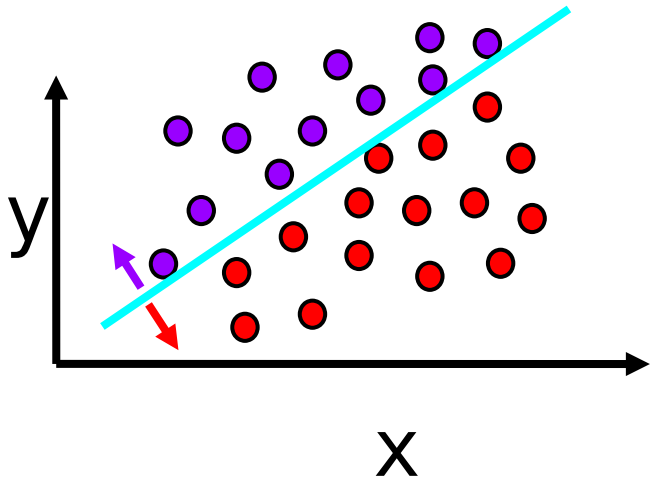
Classification in Two Dimensions

- Linear classifier
  - Given dataset, learn weights **w**
  - Output of model is weighted sum of inputs
  - P(purple | **x**) = f(**w**·**x**) = f($\Sigma_i(w_i x_i)$)
  - Where f is some function (a few options)



Classification in Two Dimensions

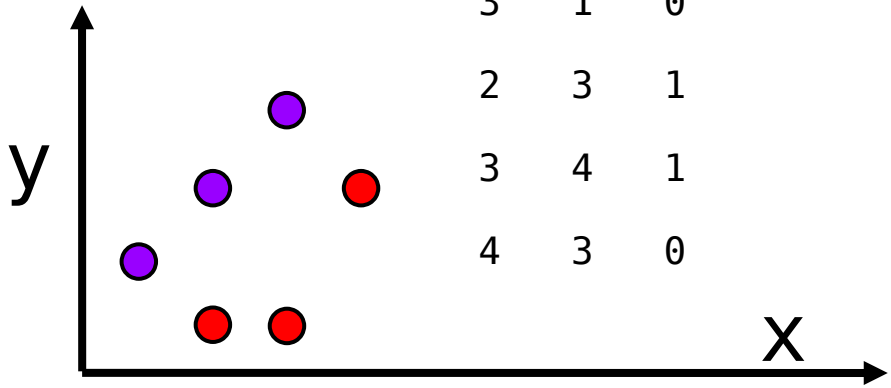
- Linear classifier
  - Given dataset, learn weights **w**
  - Output of model is weighted sum of inputs
  - P(purple | **x**) = f(**w**·**x**) = f($\Sigma_i(w_i x_i)$)
  - Where f is some function (a few options)
  - Typically a bias term:
    - f($\Sigma_i(w_i x_i) + w_{bias}$)



Classification in Two Dimensions

- Simple example:
  - Learned weights: [-1, 1]
  - f is threshold at 0

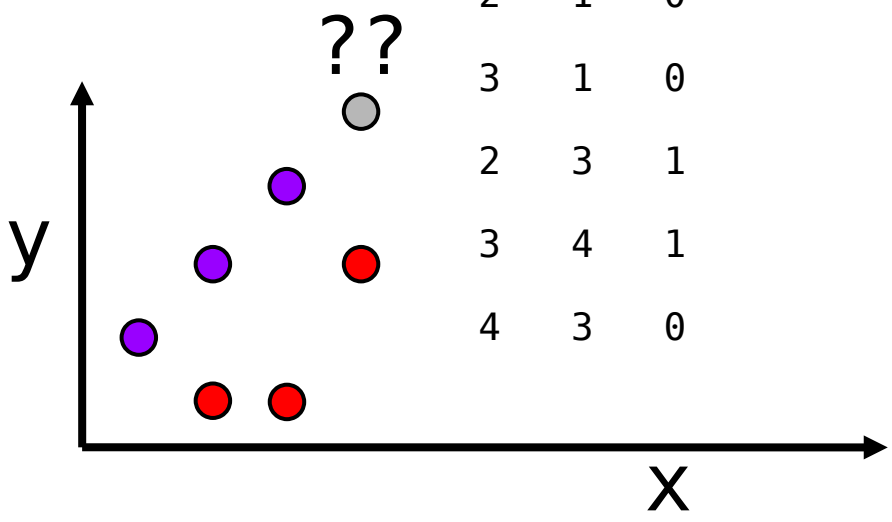
x	y	purple
1	2	1
2	1	0
3	1	0
2	3	1
3	4	1
4	3	0



Classification in Two Dimensions

- Simple example:
  - Learned weights: [-1, 1]
  - f is threshold at 0
- New data point (4, 5)
  - (**w**·**x**) = (4,5)·(-1, 1)
    = 4*-1 + 5*1 = 1

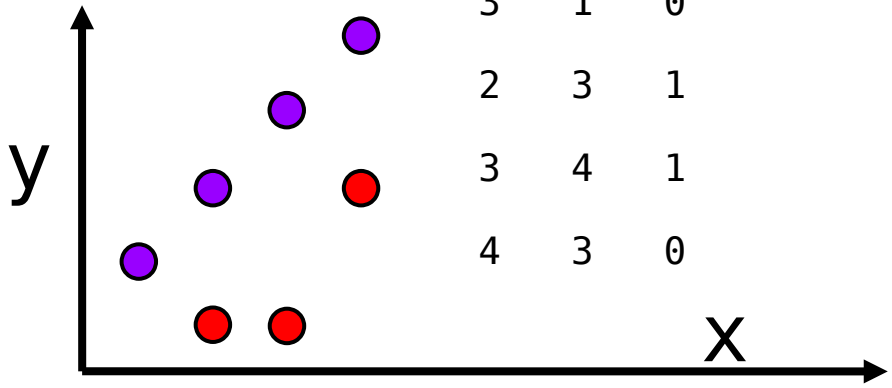
x	y	purple
1	2	1
2	1	0
3	1	0
2	3	1
3	4	1
4	3	0



Classification in Two Dimensions

- Simple example:
  - Learned weights: [-1, 1]
  - f is threshold at 0
- New data point (4, 5)
  - $(\mathbf{w}\cdot\mathbf{x}) = (4,5)\cdot(-1, 1)$
    $= 4*-1 + 5*1 = 1$
  - $f(\mathbf{w}\cdot\mathbf{x}) = f(1) = 1$

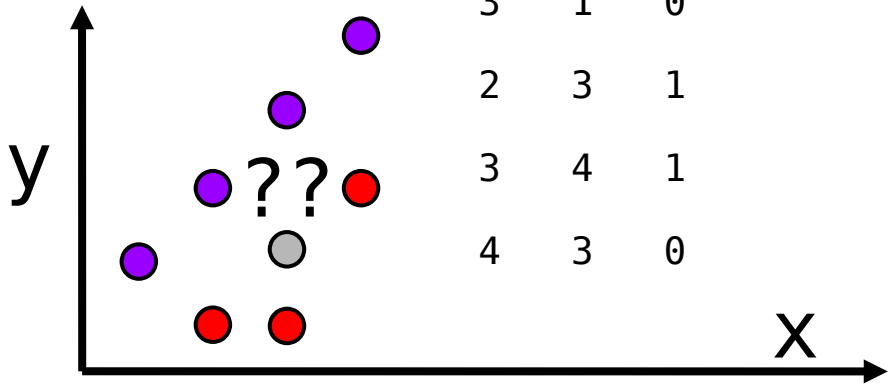
x	y	purple
1	2	1
2	1	0
3	1	0
2	3	1
3	4	1
4	3	0



Classification in Two Dimensions

- Simple example:
  - Learned weights: [-1, 1]
  - f is threshold at 0
- New data point (4, 5)
  - f(**w**·**x**) = 1
- New data point (3, 2)
  - (**w**·**x**) = (3,2)·(-1, 1)
    = 3*-1 + 2*1 = -1
  - f(**w**·**x**) = f(-1) = 0

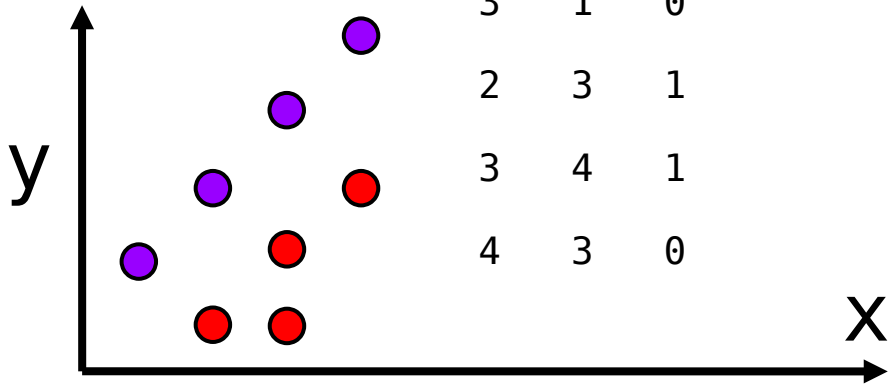
x	y	purple
1	2	1
2	1	0
3	1	0
2	3	1
3	4	1
4	3	0



Classification in Two Dimensions

- Simple example:
  - Learned weights: [-1, 1]
  - f is threshold at 0
- New data point (4, 5)
  - f(**w**·**x**) = 1
- New data point (3, 2)
  - f(**w**·**x**) = 0

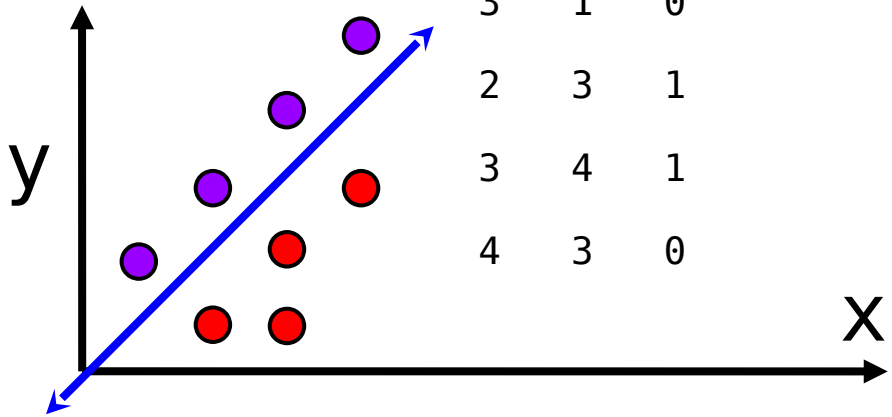
x	y	purple
1	2	1
2	1	0
3	1	0
2	3	1
3	4	1
4	3	0



Classification in Two Dimensions

- Simple example:
  - Learned weights: [-1, 1]
  - f is threshold at 0
- New data point (4, 5)
  - f(**w**·**x**) = 1
- New data point (3, 2)
  - f(**w**·**x**) = 0
- Decision boundary: x=y

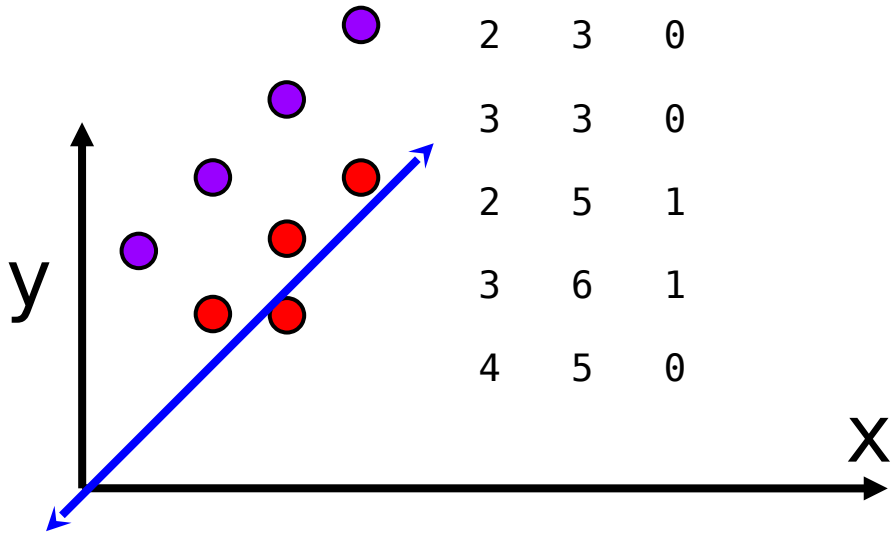
x	y	purple
1	2	1
2	1	0
3	1	0
2	3	1
3	4	1
4	3	0



What if Data is Shifted up by Two?

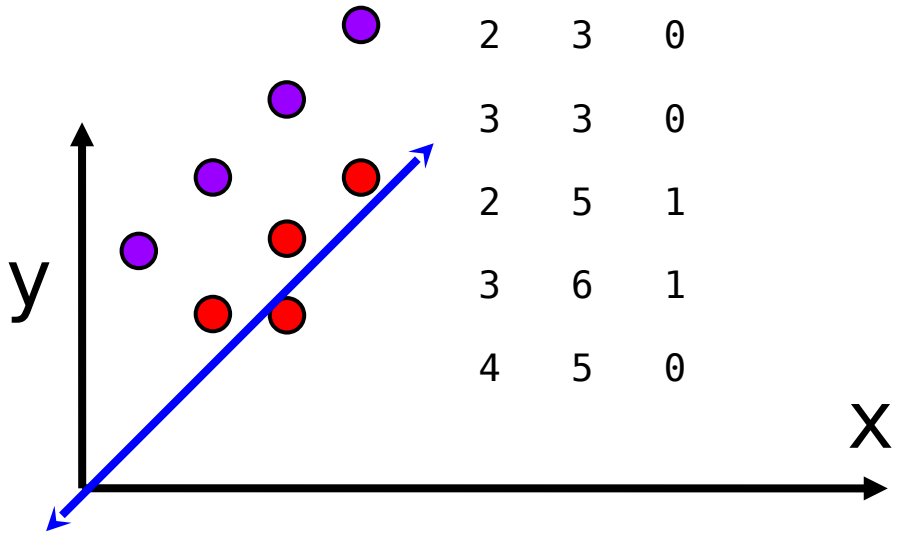
- Need a bias!:
  - Learned weights: [-1, 1]
  - f is threshold at 0

x	y	purple
1	4	1
2	3	0
3	3	0
2	5	1
3	6	1
4	5	0



What if data is shifted up by two?

- Need a bias!:
  - Learned weights: [-1, 1, **-2**]
  - f is threshold at 0

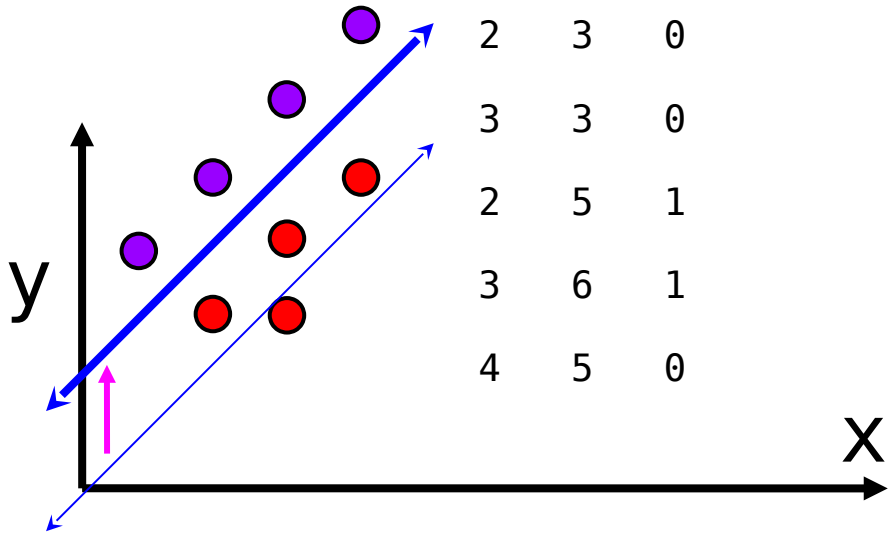


x	y	purple
1	4	1
2	3	0
3	3	0
2	5	1
3	6	1
4	5	0

What if Data is Shifted up by Two?

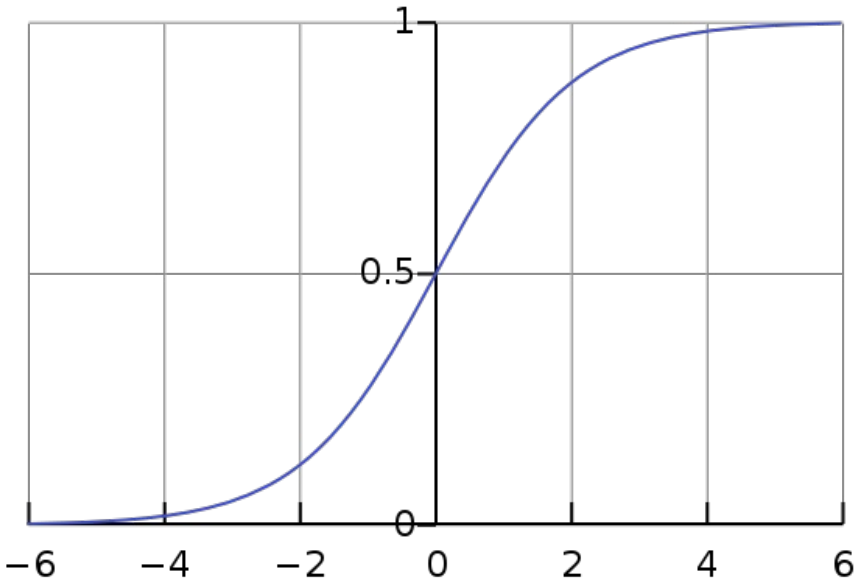
- Need a bias!:
  - Learned weights: [-1, 1, **-2**]
  - f is threshold at 0
- New data point (4, 7, **1**)
  - (**w**·**x**) = (4,7,1)·(-1,1,-2)
    = 4*-1 + 7*1 + 1*-2 = 1
  - f(**w**·**x**) = f(1) = 1

x	y	purple
1	4	1
2	3	0
3	3	0
2	5	1
3	6	1
4	5	0



Case Study: Logistic Regression

- Linear classifier, f is logistic function
  - $\sigma(x) = 1/(1 + e^{-x}) = e^{x}/(1 + e^{x})$
  - Maps all reals -> [0,1], probabilities!



Logistic Regression

- Linear classifier, f is logistic function
  - $\sigma(x) = 1/(1 + e^{-x}) = e^{x}/(1 + e^{x})$
- Want something to optimize!
  - Good choice: how well our model fits the data, likelihood

Logistic Regression

- Linear classifier, f is logistic function
  - $\sigma(x) = 1/(1 + e^{-x}) = e^{x}/(1 + e^{x})$
- Want something to optimize!
  - Good choice: how well our model fits the data, likelihood
  - **X**: training input variables, **Y**: training dependant variables
  - Want to learn **w** that models training data well

Logistic Regression

- Linear classifier, f is logistic function
  - $\sigma(x) = 1/(1 + e^{-x}) = e^{x}/(1 + e^{x})$
- Want something to optimize!
  - Good choice: how well our model fits the data, likelihood
  - **X**: training input variables, **Y**: training dependant variables
  - Want to learn **w** that models training data well
  - $L(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = Pr(\mathbf{Y} \mid \mathbf{X}, \mathbf{w}) = \mathbf{\Pi}_i Pr(\mathbf{Y}_i \mid \mathbf{X}_i, \mathbf{w})$

Logistic Regression

- Linear classifier, f is logistic function
  - $\sigma(x) = 1/(1 + e^{-x}) = e^{x}/(1 + e^{x})$
- Want something to optimize!
  - Good choice: how well our model fits the data, likelihood
  - **X**: training input variables, **Y**: training dependant variables
  - Want to learn **w** that models training data well
  - $L(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = Pr(\mathbf{Y} \mid \mathbf{X},\mathbf{w}) = \mathbf{\Pi}_i Pr(\mathbf{Y}_i \mid \mathbf{X}_i,\mathbf{w})$
  - $Pr(\mathbf{Y}_i \mid \mathbf{X}_i,\mathbf{w}) =$
    - If $\mathbf{Y}_i = 1$, $\sigma(\mathbf{w}\cdot\mathbf{X}_i)$

Logistic Regression

- Linear classifier, f is logistic function
  - $\sigma(x) = 1/(1 + e^{-x}) = e^{x}/(1 + e^{x})$
- Want something to optimize!
  - Good choice: how well our model fits the data, likelihood
  - **X**: training input variables, **Y**: training dependant variables
  - Want to learn **w** that models training data well
  - $L(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = Pr(\mathbf{Y} \mid \mathbf{X},\mathbf{w}) = \mathbf{\Pi}_i Pr(\mathbf{Y}_i \mid \mathbf{X}_i,\mathbf{w})$
  - $Pr(\mathbf{Y}_i \mid \mathbf{X}_i,\mathbf{w}) =$
    - If $\mathbf{Y}_i = 1$, $\sigma(\mathbf{w}\cdot\mathbf{X}_i)$
    - If $\mathbf{Y}_i = 0$, $1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i)$

Logistic Regression

- Linear classifier, f is logistic function
  - $\sigma(x) = 1/(1 + e^{-x}) = e^{x}/(1 + e^{x})$
- Want something to optimize!
  - Good choice: how well our model fits the data, likelihood
  - **X**: training input variables, **Y**: training dependant variables
  - Want to learn **w** that models training data well
  - $L(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = Pr(\mathbf{Y} \mid \mathbf{X},\mathbf{w}) = \mathbf{\Pi}_i Pr(\mathbf{Y}_i \mid \mathbf{X}_i,\mathbf{w})$
  - $Pr(\mathbf{Y}_i \mid \mathbf{X}_i,\mathbf{w}) =$
    - If $\mathbf{Y}_i = 1$, $\sigma(\mathbf{w}\cdot\mathbf{X}_i)$
    - If $\mathbf{Y}_i = 0$, $1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i)$
  - $L(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = \mathbf{\Pi}_i[(\sigma(\mathbf{w}\cdot\mathbf{X}_i))^{Yi} * (1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))^{(1-Yi)}]$

Logistic Regression

- Linear classifier, f is logistic function
  - $\sigma(x) = 1/(1 + e^{-x}) = e^{x}/(1 + e^{x})$
- Want something to optimize!
  - Good choice: how well our model fits the data, likelihood
  - **X**: training input variables, **Y**: training dependant variables
  - Want to learn **w** that models training data well
  - $L(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = Pr(\mathbf{Y} \mid \mathbf{X},\mathbf{w}) = \mathbf{\Pi}_i Pr(\mathbf{Y}_i \mid \mathbf{X}_i,\mathbf{w})$
  - $Pr(\mathbf{Y}_i \mid \mathbf{X}_i,\mathbf{w}) =$
    - If $\mathbf{Y}_i = 1$, $\sigma(\mathbf{w}\cdot\mathbf{X}_i)$
    - If $\mathbf{Y}_i = 0$, $1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i)$
  - $L(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = \mathbf{\Pi}_i[(\sigma(\mathbf{w}\cdot\mathbf{X}_i))^{Yi} * (1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))^{(1-Yi)}]$
  - In practice we use log likelihood, it's simpler later!!

Logistic Regression

- Linear classifier, f is logistic function
  - $\sigma(x) = 1/(1 + e^{-x}) = e^{x}/(1 + e^{x})$
- Want something to optimize!
  - $\log L(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = \log \prod_i [(\sigma(\mathbf{w}\cdot\mathbf{X}_i))^{Y_i} * (1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))^{(1-Y_i)}]$
  - $= \Sigma_i \log[(\sigma(\mathbf{w}\cdot\mathbf{X}_i))^{Y_i} * (1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))^{(1-Y_i)}]$
  - $= \Sigma_i [\mathbf{Y}_i \log(\sigma(\mathbf{w}\cdot\mathbf{X}_i)) + (1-\mathbf{Y}_i)\log(1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))]$

Logistic Regression

- Linear classifier, f is logistic function
  - $\sigma(x) = 1/(1 + e^{-x}) = e^{x}/(1 + e^{x})$
- Want something to optimize!
  - $\log L(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = \log \prod_i [(\sigma(\mathbf{w}\cdot\mathbf{X}_i))^{Y_i} * (1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))^{(1-Y_i)}]$
  - $= \Sigma_i \log[(\sigma(\mathbf{w}\cdot\mathbf{X}_i))^{Y_i} * (1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))^{(1-Y_i)}]$
  - $= \Sigma_i [\mathbf{Y}_i \log(\sigma(\mathbf{w}\cdot\mathbf{X}_i)) + (1-\mathbf{Y}_i)\log(1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))]$
- Can we take derivative and set to 0?
  - No! :-(no closed form solution
  - BUT! We can still optimize

Gradient Descent

Calculating $\nabla L(\mathbf{w})$ can be hard, especially for big data, |data| very large.

What if we estimate it it instead?

How do we estimate things?

Stochastic Gradient Descent (SGD)

Estimate $\nabla L(\mathbf{w})$ with only some of the data

Which part of data?

Random sample! (it's unbiased)

How many points in the sample?

Who knows! It's up to you

Stochastic Gradient Descent (SGD)

Estimate $\nabla L(\mathbf{w})$ with only some of the data; Before:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \Sigma_i \nabla L_i(\mathbf{w}), \text{ for all i in |data|}$$

Now:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \Sigma_j \nabla L_j(\mathbf{w}), \text{ for some subset j}$$

Maybe even:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla L_k(\mathbf{w}), \text{ for some random k}$$

# of points used for update is called *batch size*

SGD with Logistic Regression

Say we want to do SGD with batch size = 1

Want to maximize $\log L(\mathbf{w} \mid \mathbf{X}_i, \mathbf{Y}_i)$ for random i

$$= \mathbf{Y}_i \log(\sigma(\mathbf{w} \cdot \mathbf{X}_i)) + (1-\mathbf{Y}_i)\log(1 - \sigma(\mathbf{w} \cdot \mathbf{X}_i))$$

https://math.stackexchange.com/questions/477207/derivative-of-cost-function-for-logistic-regression

SGD with Logistic Regression

Say we want to do SGD with batch size = 1

Want to maximize log $L(\mathbf{w} \mid \mathbf{X}_i, \mathbf{Y}_i)$ for random i

$$= \mathbf{Y}_i \log(\sigma(\mathbf{w} \cdot \mathbf{X}_i)) + (1-\mathbf{Y}_i)\log(1 - \sigma(\mathbf{w} \cdot \mathbf{X}_i))$$

But wait!

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

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

https://math.stackexchange.com/questions/477207/derivative-of-cost-function-for-logistic-regression

SGD with logistic regression

Say we want to do SGD with batch size = 1

Want to maximize log $L(\mathbf{w} \mid \mathbf{X}_i, \mathbf{Y}_i)$ for random i

$$= \mathbf{Y}_i \log(\sigma(\mathbf{w}\cdot\mathbf{X}_i)) + (1-\mathbf{Y}_i)\log(1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))$$

But wait!

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

$\log(1-\sigma(x)) = \log(1 - 1/(1 + e^{-x})) = \log(e^{-x}/(1 + e^{-x}))= -x-\log(1+e^{-x})$ So:

$\log L_i(\mathbf{w}) = -\mathbf{Y}_i \log(1+e^{-(\mathbf{w}\cdot\mathbf{X}_i)}) + (1-\mathbf{Y}_i)[-(\mathbf{w}\cdot\mathbf{X}_i) - \log(1+e^{-(\mathbf{w}\cdot\mathbf{X}_i)})]$

$= \mathbf{Y}_i(\mathbf{w}\cdot\mathbf{X}_i) - (\mathbf{w}\cdot\mathbf{X}_i) - \log(1+e^{-(\mathbf{w}\cdot\mathbf{X}_i)})$

https://math.stackexchange.com/questions/477207/derivative-of-cost-function-for-logistic-regression

SGD with logistic regression

$$\log L_i(\mathbf{w}) = -\mathbf{Y}_i \log(1+e^{-(\mathbf{w}\cdot\mathbf{X}_i)}) + (1-\mathbf{Y}_i)[-(\mathbf{w}\cdot\mathbf{X}_i) - \log(1+e^{-(\mathbf{w}\cdot\mathbf{X}_i)})]$$

$$= \mathbf{Y}_i(\mathbf{w}\cdot\mathbf{X}_i) - (\mathbf{w}\cdot\mathbf{X}_i) - \log(1+e^{-(\mathbf{w}\cdot\mathbf{X}_i)})$$ But wait again:

$$-x - \log(1 + e^{-x}) = -[\log(e^x) + \log(1 + e^{-x})] = -\log(e^x + 1)$$

So:

$$\mathbf{Y}_i(\mathbf{w}\cdot\mathbf{X}_i) - (\mathbf{w}\cdot\mathbf{X}_i) - \log(1+e^{-(\mathbf{w}\cdot\mathbf{X}_i)}) = \mathbf{Y}_i(\mathbf{w}\cdot\mathbf{X}_i) - \log(1+e^{(\mathbf{w}\cdot\mathbf{X}_i)})$$

Need to take derivatives:

$$\nabla \log L_i(\mathbf{w}) = \nabla \mathbf{Y}_i(\mathbf{w}\cdot\mathbf{X}_i) - \nabla \log(1+e^{(\mathbf{w}\cdot\mathbf{X}_i)})$$

$$= \mathbf{Y}_i\mathbf{X}_i - 1/(1+e^{(\mathbf{w}\cdot\mathbf{X}_i)}) * e^{(\mathbf{w}\cdot\mathbf{X}_i)} * \mathbf{X}_i = \mathbf{X}_i[\mathbf{Y}_i - e^{(\mathbf{w}\cdot\mathbf{X}_i)}/(1+e^{(\mathbf{w}\cdot\mathbf{X}_i)})]$$

$$= \mathbf{X}_i[\mathbf{Y}_i - \sigma(\mathbf{w}\cdot\mathbf{X}_i)]$$

https://math.stackexchange.com/questions/477207/derivative-of-cost-function-for-logistic-regression

SGD with Logistic Regression

$\nabla \log L_i(\mathbf{w}) = \mathbf{X}_i[\mathbf{Y}_i - \sigma(\mathbf{w} \cdot \mathbf{X}_i)]$

So our weight update rule is:

$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta\, \mathbf{X}_i[\mathbf{Y}_i - \sigma(\mathbf{w}_t \cdot \mathbf{X}_i)]$

Why plus?

Doing gradient ascent up the likelihood function