The Ancient Secrets of Computer Vision

Logistics:

- Homework 2 due today
  - Homework 3 out soon
- Will release to solutions if you email me

Chapter Ten

Machine Learning

Until now: low/mid level vision

- Low level
    - Pixel manipulations
    - Image corrections
    - Feature extraction
- Mid level
    - Images <-> images
        - Panorama
    - Image <-> world
        - Stereo
    - Images <-> time
        - Optical flow

Why did we extract these features anyway?

- High level vision!
  - Image <-> semantics, meaning
- Make predictions about images, what’s in them
  - How do we make predictions?
  - Hard coded rules?
  - Learn things from data!

Time to learn all of machine learning in one class

(haha not really)

What is machine learning?

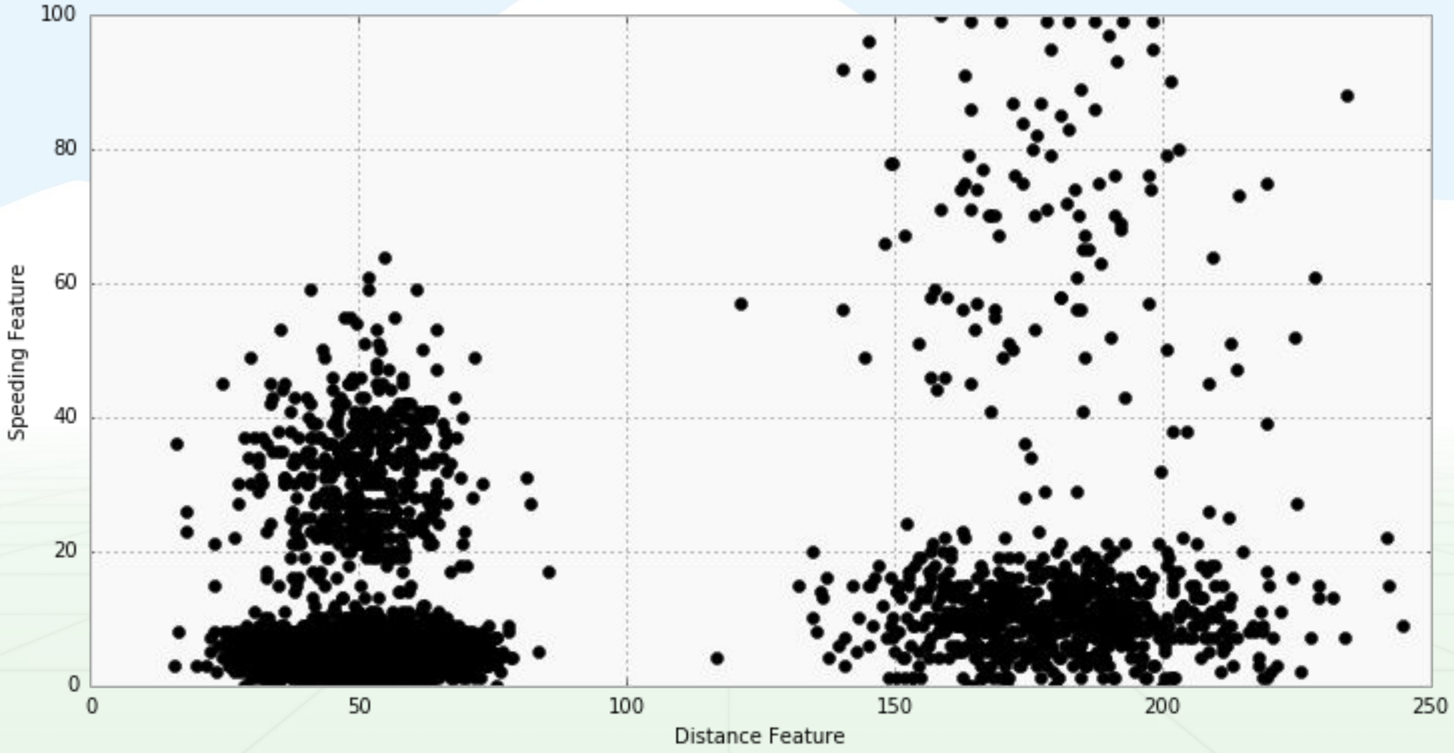
- Algorithms to approximate functions
  - Usually use lots of statistics
  - Usually minimize some form of *loss function*
- Supervised learning
  - Given inputs to a function, try to predict the output
  - Have lots of labelled examples
- Semi-supervised learning
  - Same but number of labelled examples < number of examples
- Unsupervised learning
  - Want to model unlabelled data
  - Find similarities and differences between subgroups of data
  - Learn functions to generate new data

Unsupervised learning

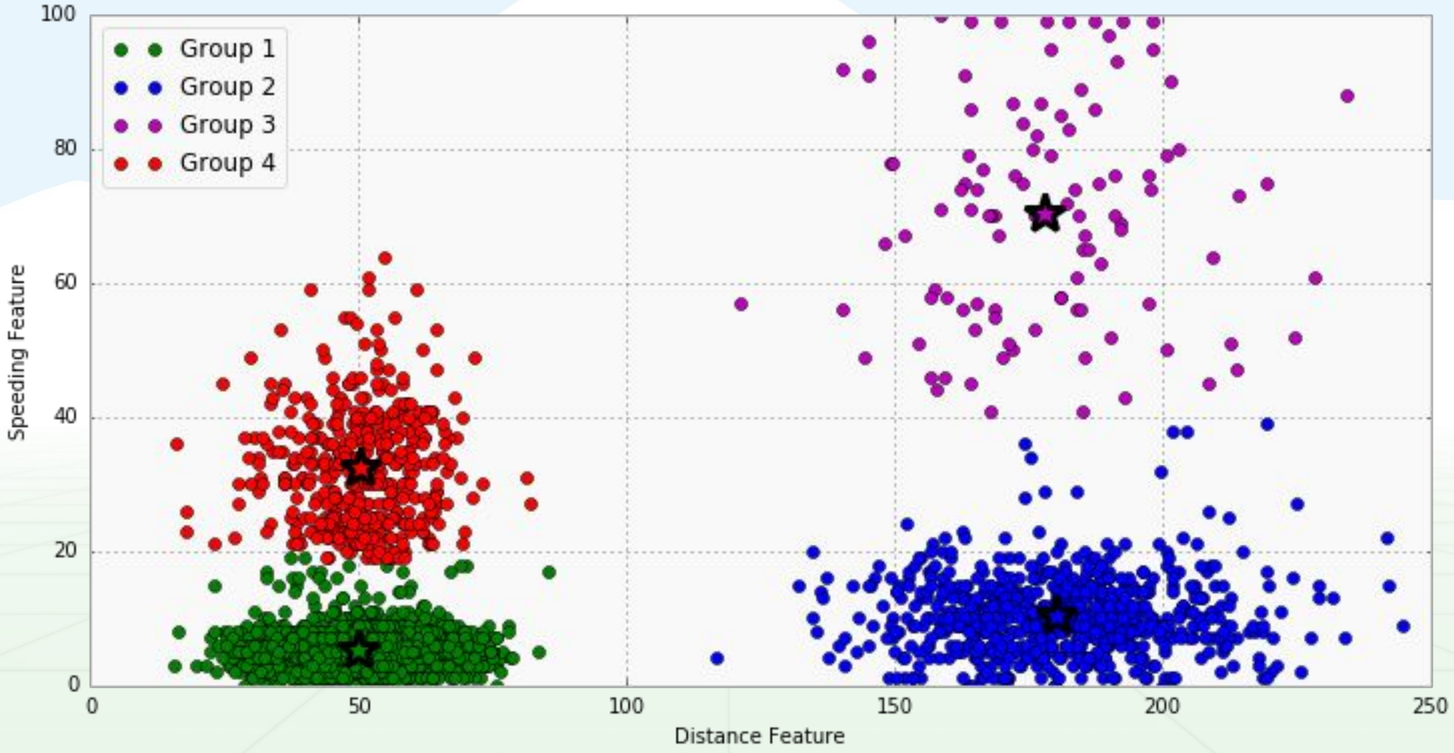
No labels, just looking for patterns in data

E.g. clustering, in data with multiple clusters, what are they, how big, etc.

Clustering: finding groups in data



Clustering: finding groups in data



K-means clustering

Assume points are close to other points in group, far from points out of group

Algorithm:

Randomly initialize cluster centers

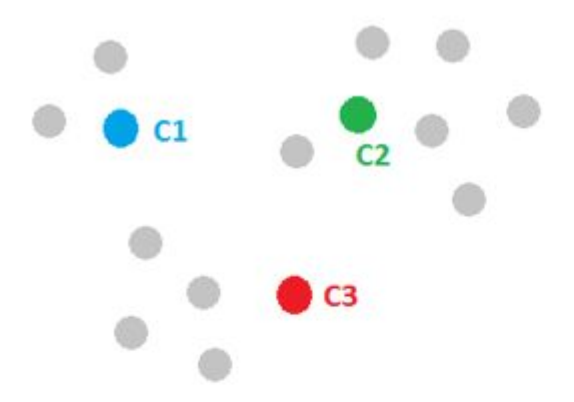
https://healthcare.ai/step-step-k-means-clustering/

K-means clustering

Assume points are close to other points in group, far from points out of group

Algorithm:

Randomly initialize cluster centers



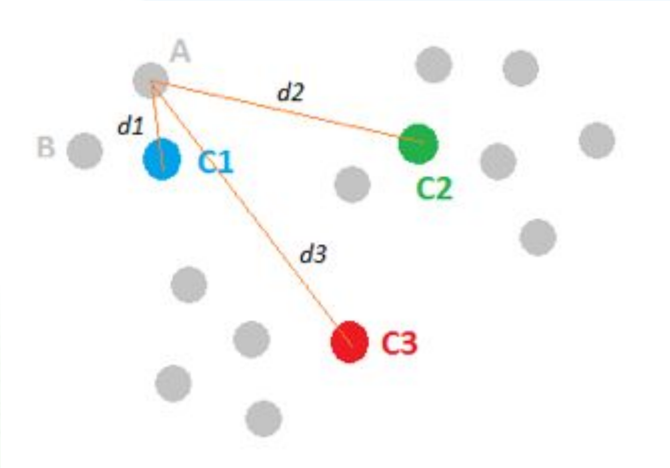
https://healthcare.ai/step-step-k-means-clustering/

K-means clustering

Assume points are close to other points in group, far from points out of group

Algorithm:

Randomly initialize cluster centers
Calculate distance points <-> centers



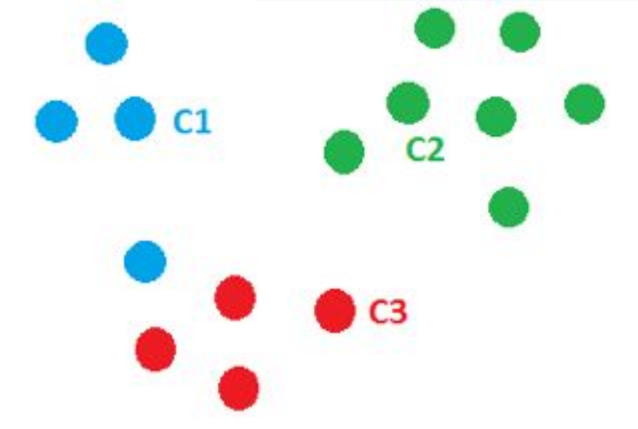
https://healthcare.ai/step-step-k-means-clustering/

K-means clustering

Assume points are close to other points in group, far from points out of group

Algorithm:

Randomly initialize cluster centers
Calculate distance points <-> centers
Assign each point to closest cluster center



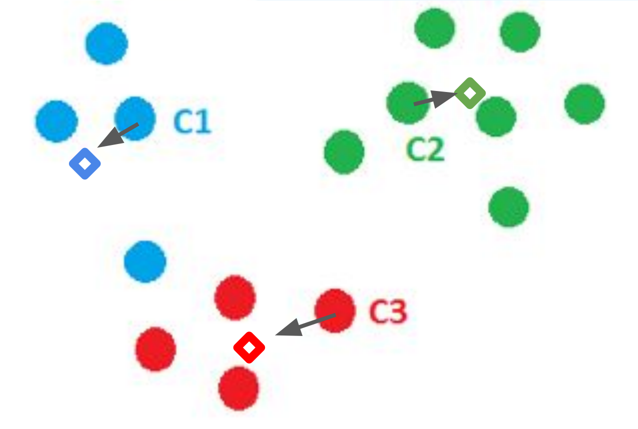
https://healthcare.ai/step-step-k-means-clustering/

K-means clustering

Assume points are close to other points in group, far from points out of group

Algorithm:

Randomly initialize cluster centers
Calculate distance points <-> centers
Assign each point to closest cluster center
Update cluster centers: avg of points



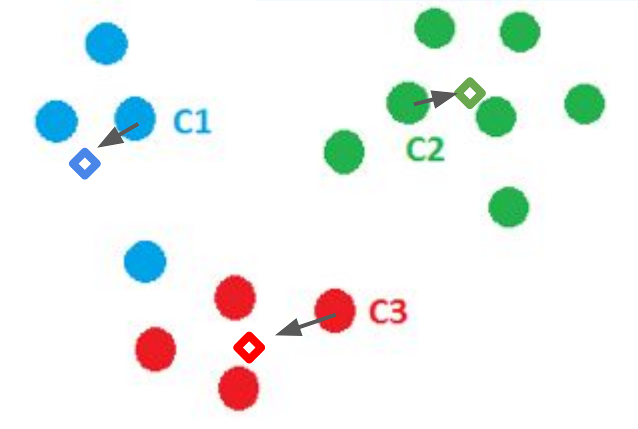
https://healthcare.ai/step-step-k-means-clustering/

K-means clustering

Assume points are close to other points in group, far from points out of group

Algorithm:

Randomly initialize cluster centers
Calculate distance points <-> centers
Assign each point to closest cluster center
Update cluster centers: avg of points
Repeat!



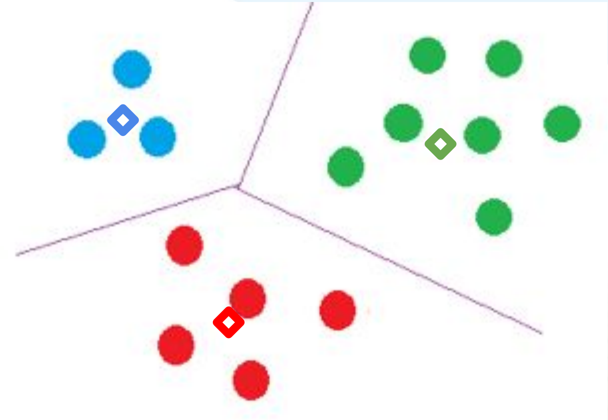
https://healthcare.ai/step-step-k-means-clustering/

K-means clustering

Assume points are close to other points in group, far from points out of group

Algorithm:

Randomly initialize cluster centers
Loop until converged:
    Calculate distance points <-> centers
    Assign each point to closest center
    Update cluster centers: avg of points



https://healthcare.ai/step-step-k-means-clustering/

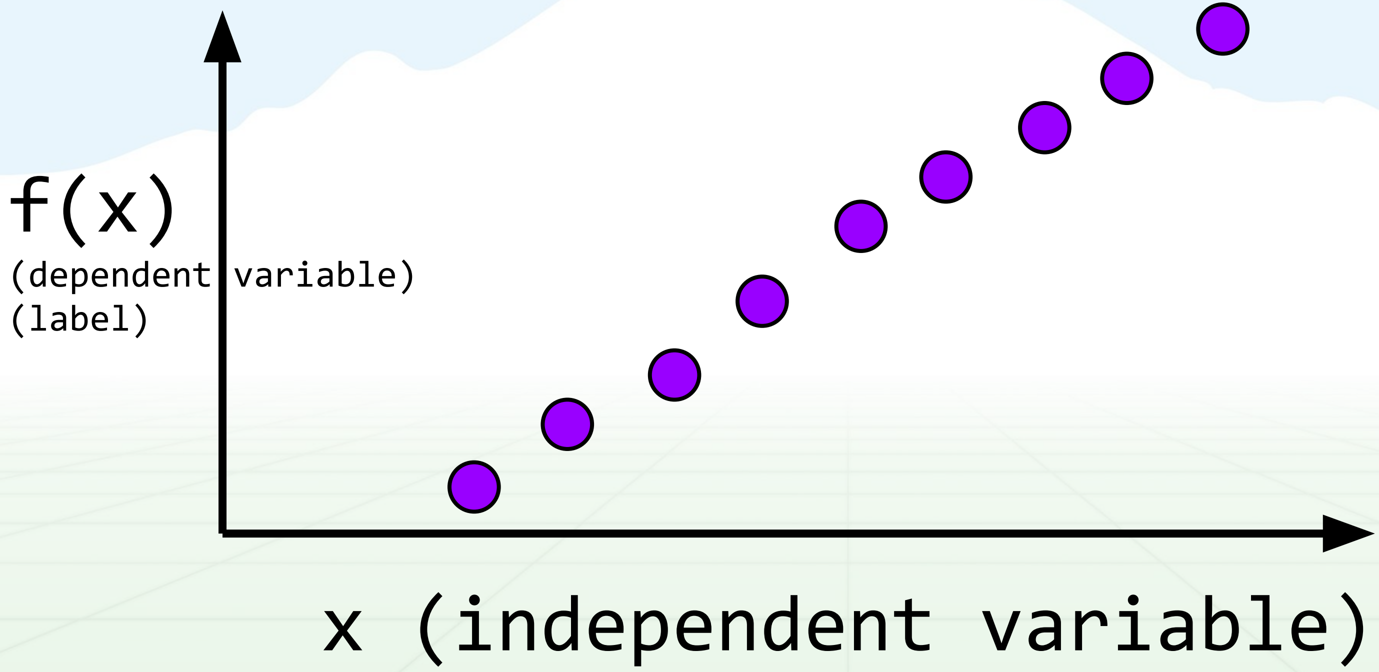
Clustering on images

Group together pixels by color, automatic segmentation: k-means, k = 2

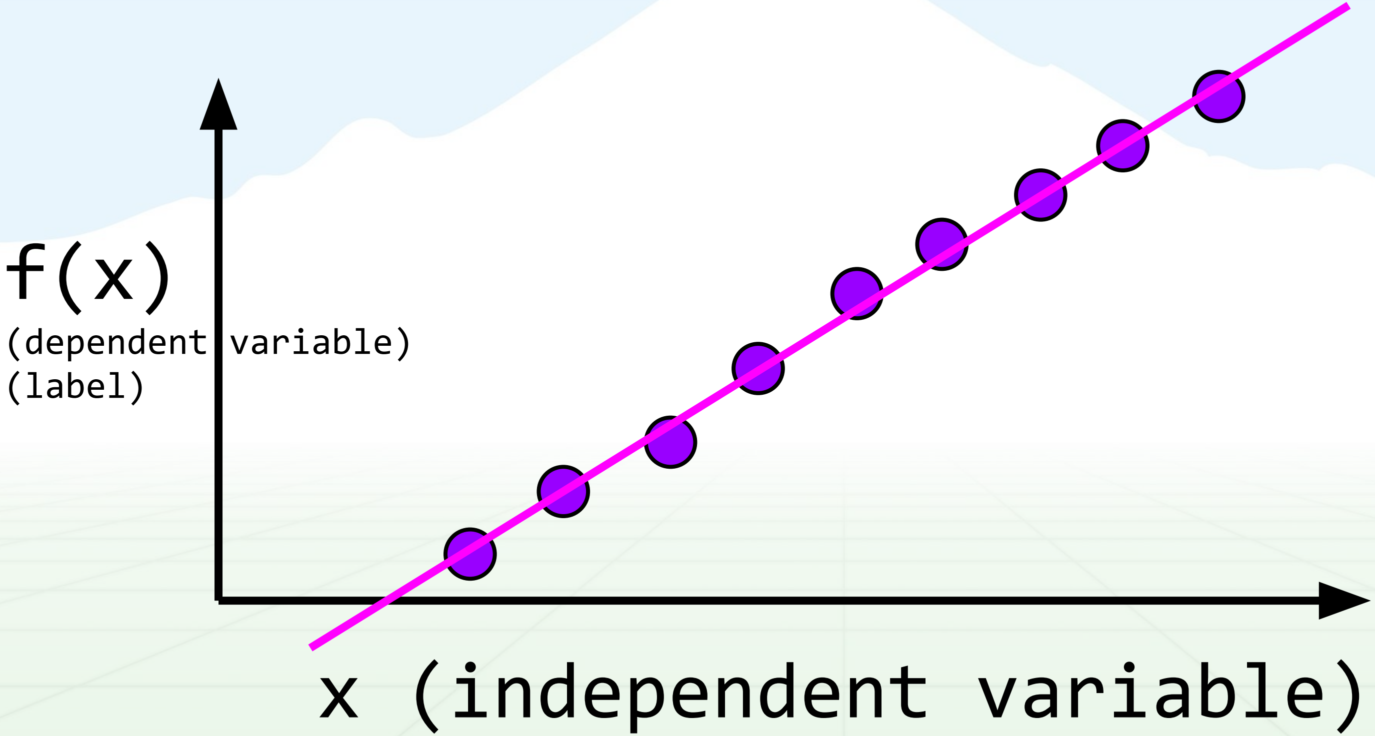
Clustering on images

Group together pixels by color, automatic segmentation: k-means, k = 4

Supervised Learning: Want to estimate f

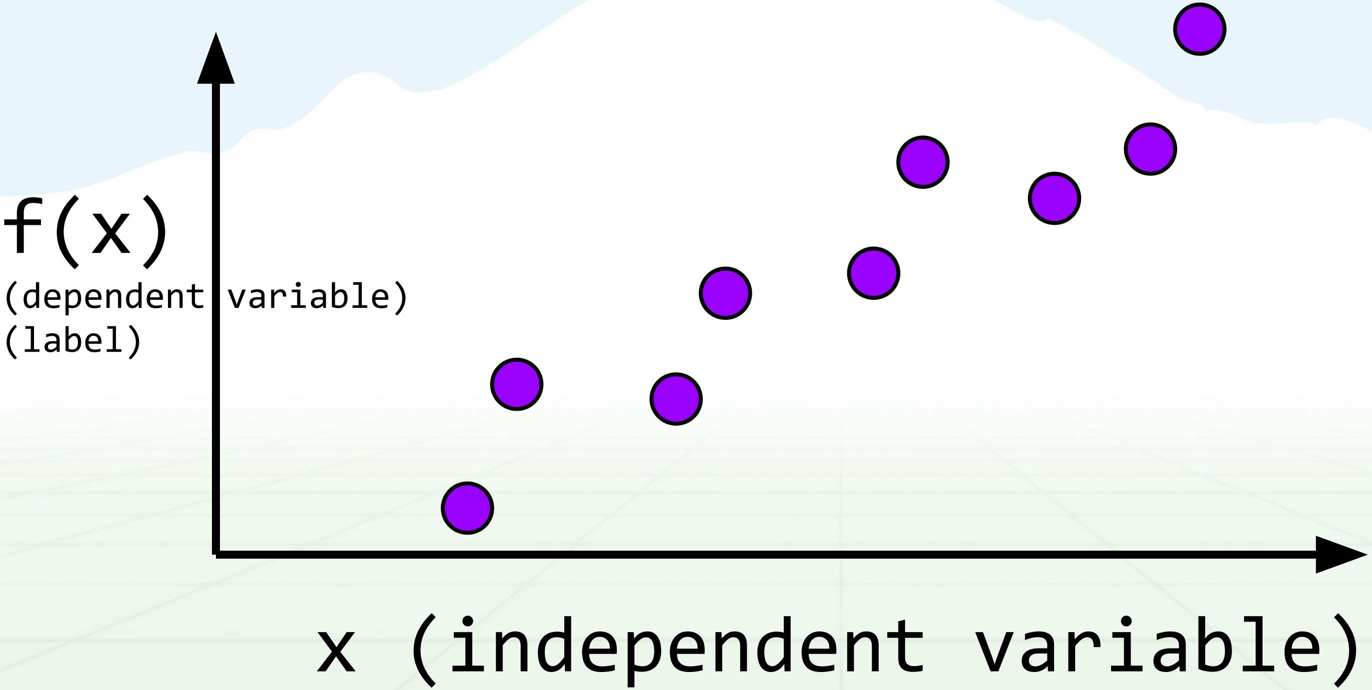


Here's one possible f*



Data often has *noise*

Why?



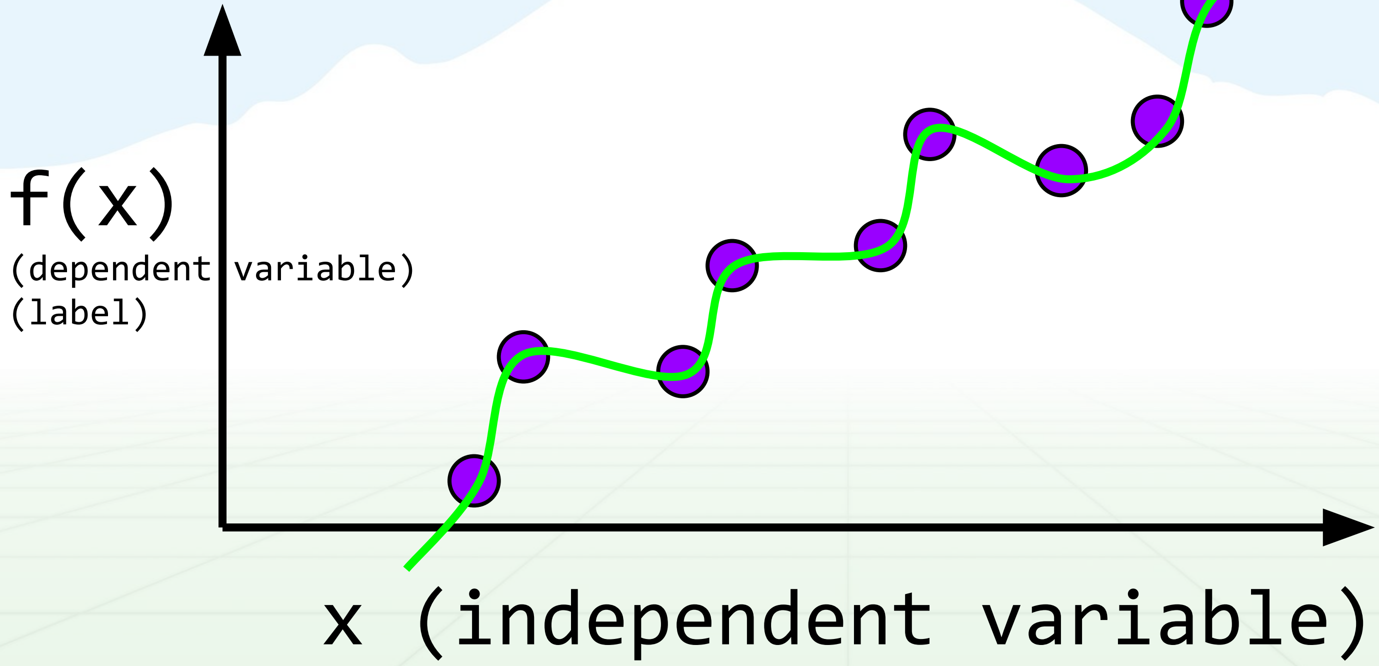
Data often has *noise*

Why?

- Randomness
  - Static in phone lines, random distribution of photons hitting sensor, sensors aren’t precise
- Mislabelled data
  - Common when humans label lots of data
- Variables outside of model
  - Variations might look like noise but are explained by a hidden, unknown variable

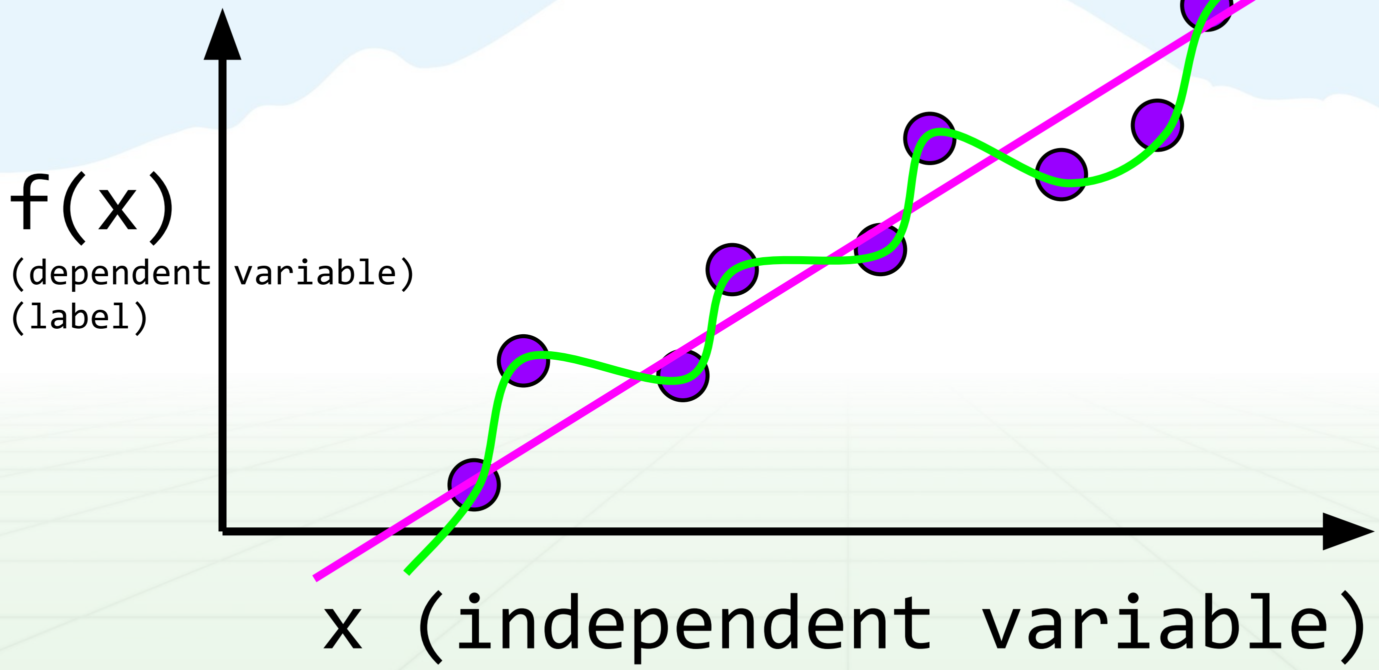
Data often has *noise*

Do we want a more complex f*?



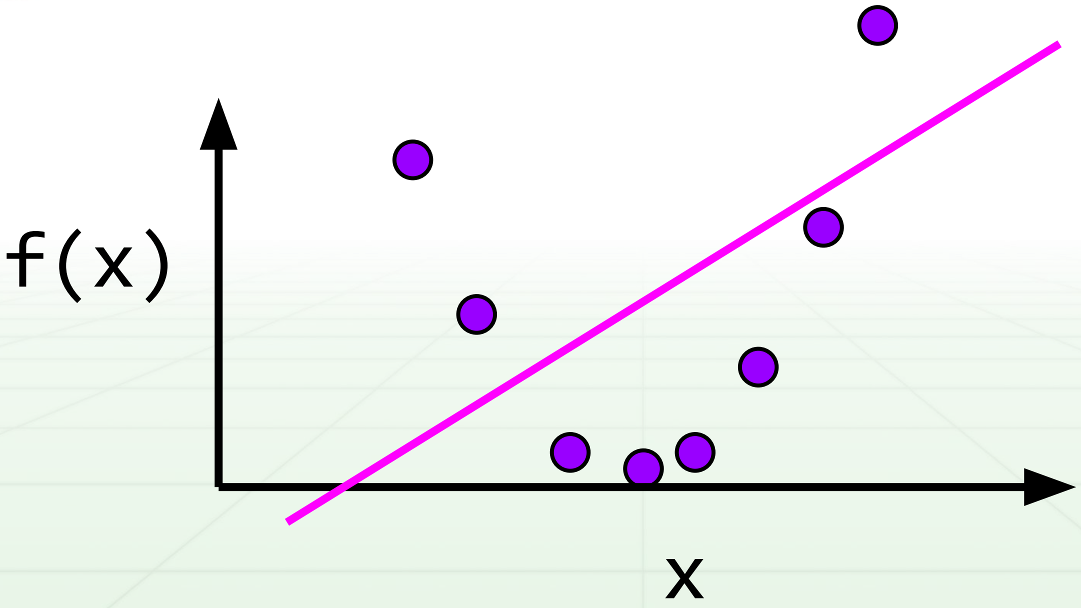
Sometimes the data is more complex

Which model is right?



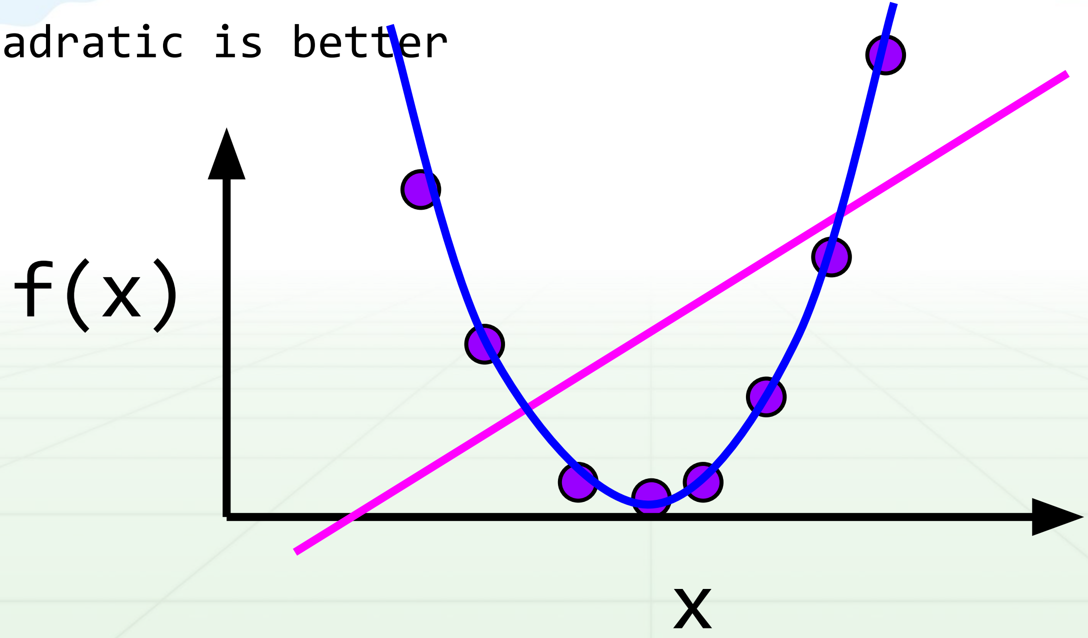
Bias / Variance tradeoff

- Bias
  - Error from assumptions model makes about data
  - Linear model assumes data is linear, bad for data that isn’t



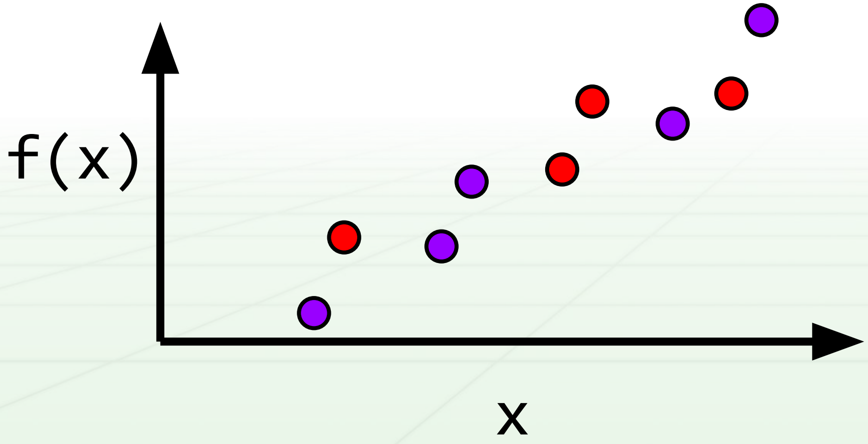
Bias / Variance tradeoff

- Bias
  - Error from assumptions model makes about data
  - Linear model assumes data is linear, bad for data that isn't
  - Quadratic is better



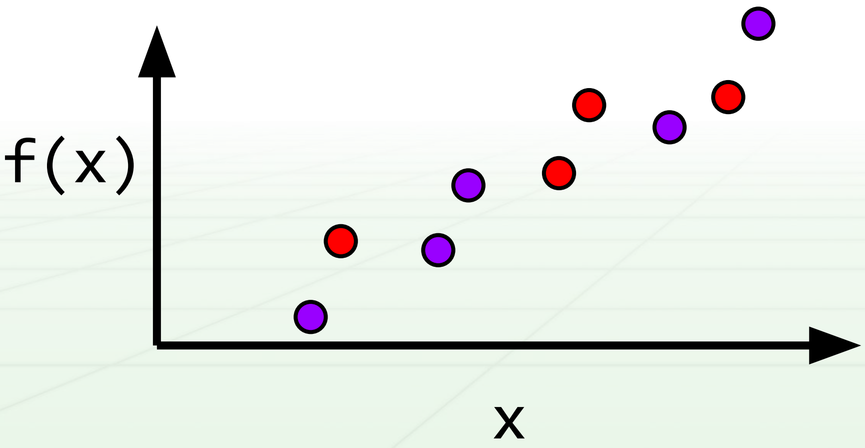
Bias / Variance tradeoff

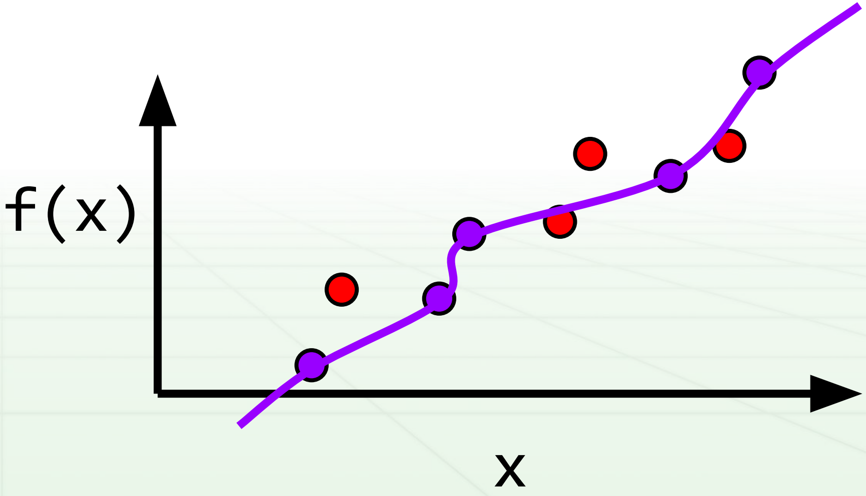
- Variance
  - Algorithm's sensitivity to noise
  - More complex algorithms are more sensitive!



Bias / Variance tradeoff

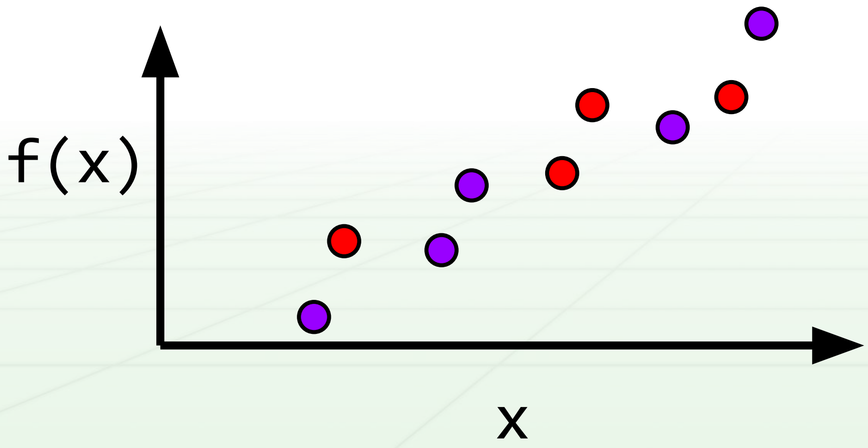
- Variance
  - Algorithm’s sensitivity to noise
  - More complex algorithms are more sensitive!

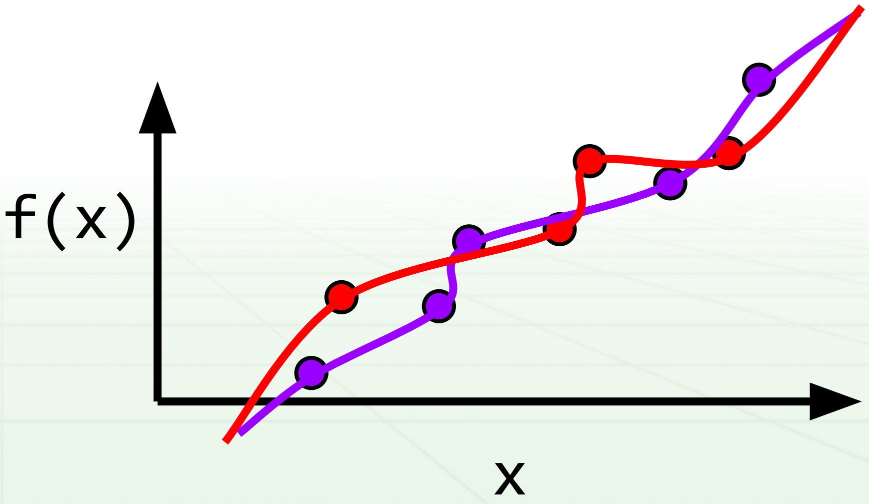




Bias / Variance tradeoff

- Variance
  - Algorithm’s sensitivity to noise
  - More complex algorithms are more sensitive!
  - High variance hurts generalization, *overfitting*



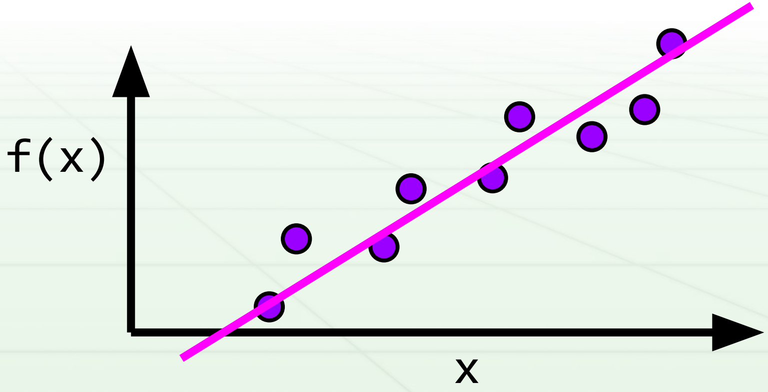


Bias / Variance tradeoff

- Noise
  - Random variations in data
- Bias
  - Error from assumptions model makes about data
  - Less complex algorithms -> more assumptions about data
- Variance
  - Algorithm’s sensitivity to noise
  - More complex algorithms are more sensitive!
  - High variance hurts generalization

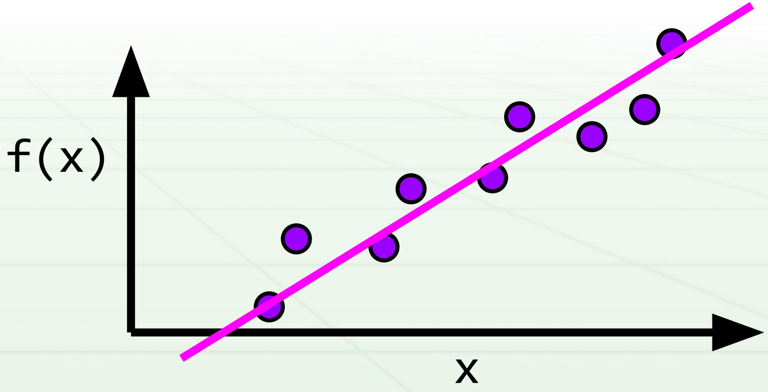
Linear regression

- $f^*(x) = ax + b$
- Learn a and b from data (how?)



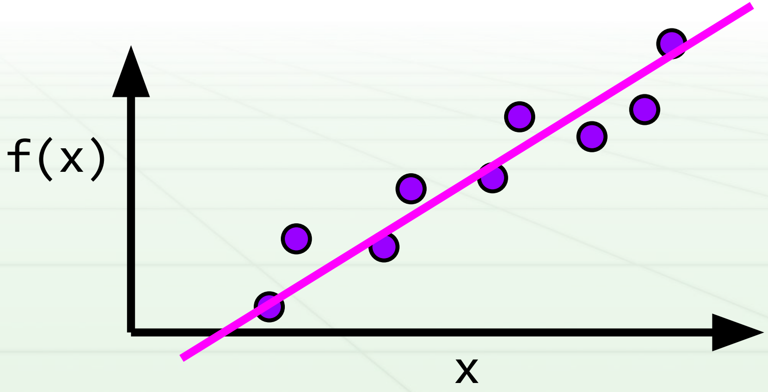
Linear regression

- $f^*(x) = ax + b$
- Learn a and b from data (how?)
  - Minimize squared error!
  - Loss function $L(f^*) = \Sigma_i ||f(x_i) - f^*(x_i)||^2$



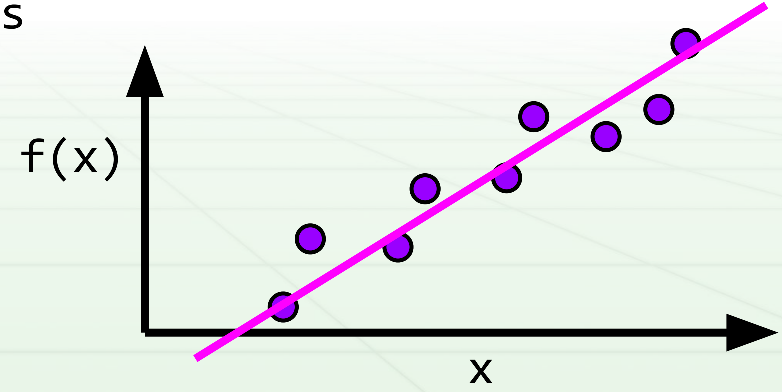
Linear regression

- $f^*(x) = ax + b$
- Learn a and b from data (how?)
  - Minimize squared error!
  - Loss function $L(f^*) = \Sigma_i \ ||f(x_i) - f^*(x_i)||^2$
  - Want $\text{argmin}_{a,b}[L(f^*)]$
  - Extrema when derivative = 0



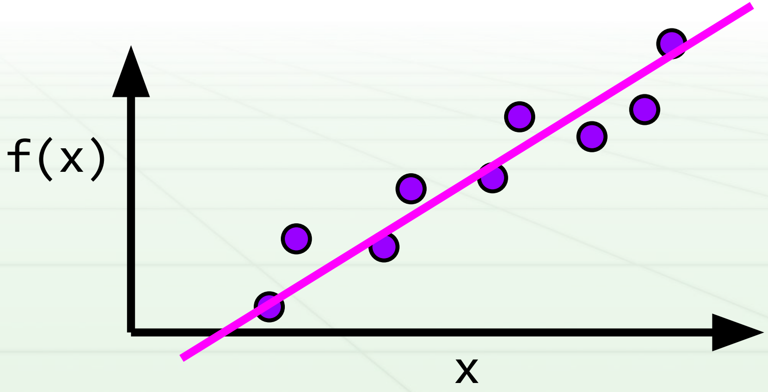
Linear regression

- $f^*(x) = ax + b*1$
- Learn a and b from data (how?)
  - Minimize squared error!
  - Loss function $L(f^*) = \Sigma_i \, ||f(x_i) - f^*(x_i)||^2$
  - Want $\text{argmin}_{a,b}[L(f^*)]$
  - Extrema when derivative = 0
  - Solve linear system of equations
    - **Ma = b**
  - Already did this!



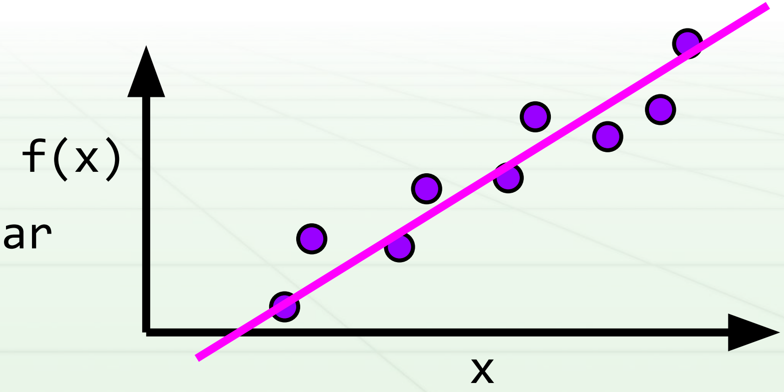
Linear regression

- $f^*(x) = ax + b$
- Learn a and b from data (how?)
- High bias: linear assumption
- Low variance
- Benefits:
  - Closed form solution
  - Fast to compute for new data



Linear regression

- $f^*(x) = ax + b$
- Learn a and b from data (how?)
- High bias: linear assumption
- Low variance
- Benefits:
    - Closed form solution
    - Fast to compute for new data
- Weaknesses:
    - Not very powerful, assumes linear
    - Underfit more interesting data

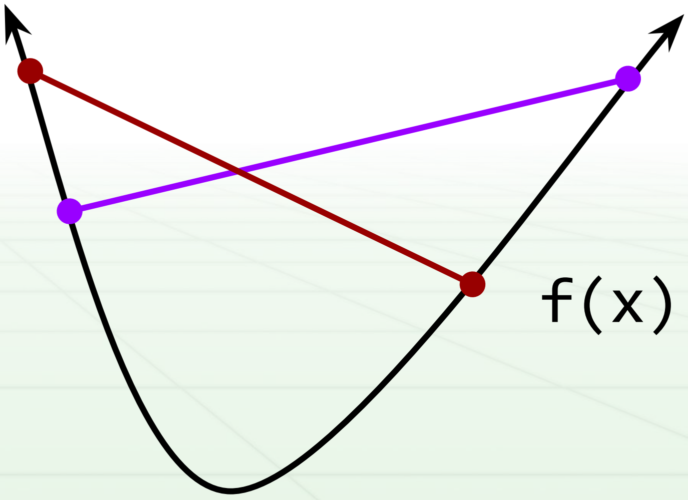


(Aside) Convex vs Non-convex

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

Why is it important? Any local extrema is global extrema!

If our loss function is convex, can set derivative = 0, solve for parameters (sometimes still no closed-form)

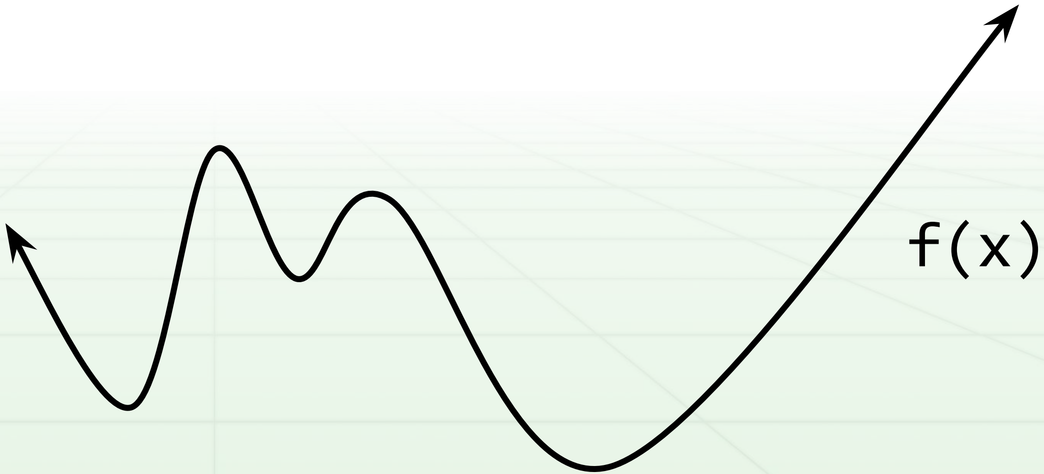


(Aside) Convex vs Non-convex

Non-convex function: no rules!

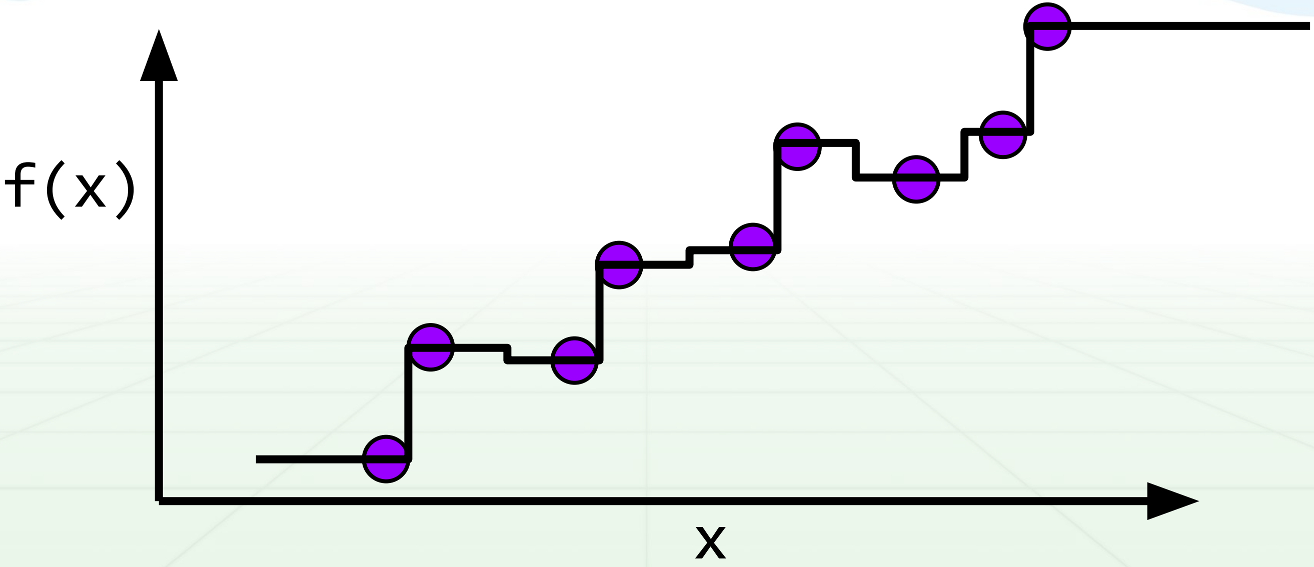
Local optima are not global optima

Usually no easy way to find global or local optima, harder to optimize



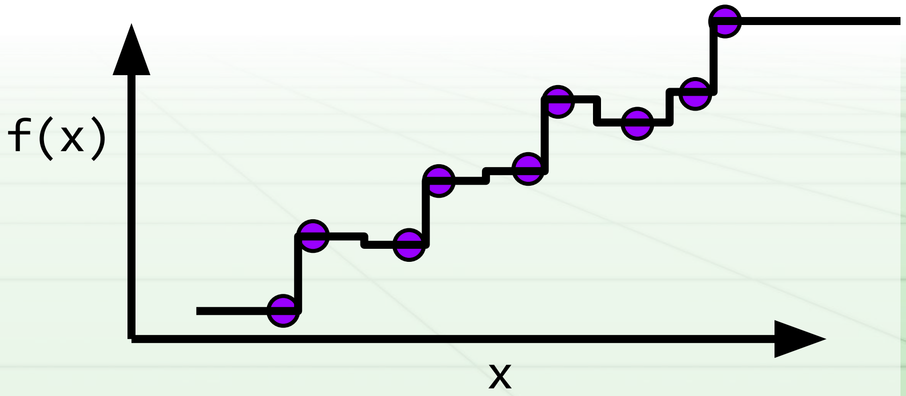
Nearest neighbor

- $f^*(x) = f(x')$ for nearest x' in training set



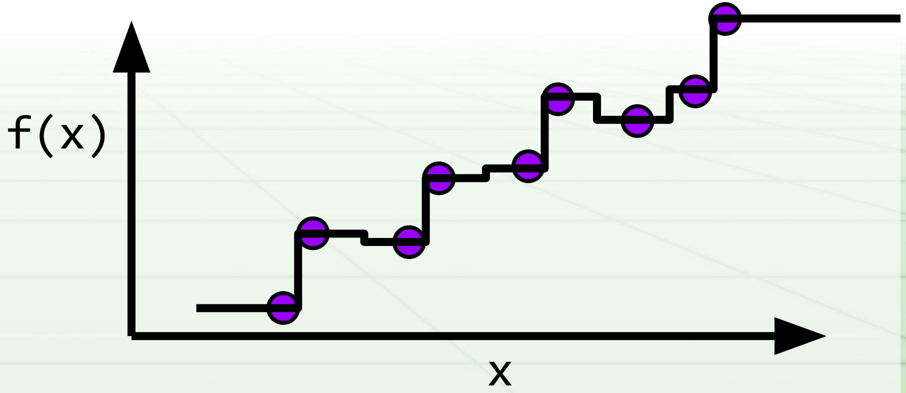
Nearest neighbor

- $f^*(x) = f(x')$ for nearest x' in training set
- Low bias: no assumptions about data
- High variance: very sensitive to training set



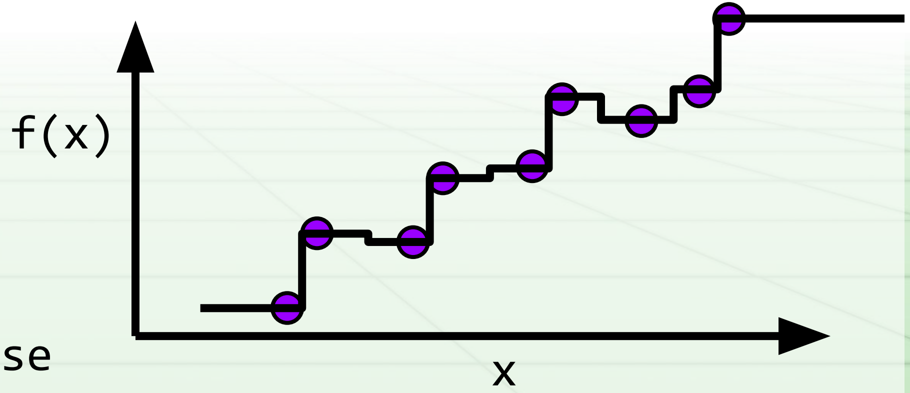
Nearest neighbor

- $f^*(x) = f(x')$ for nearest x' in training set
- Low bias: no assumptions about data
- High variance: very sensitive to training set
- Benefits:
  - Super easy to implement
  - Easy to understand
  - Arbitrarily powerful, esp with lots of data



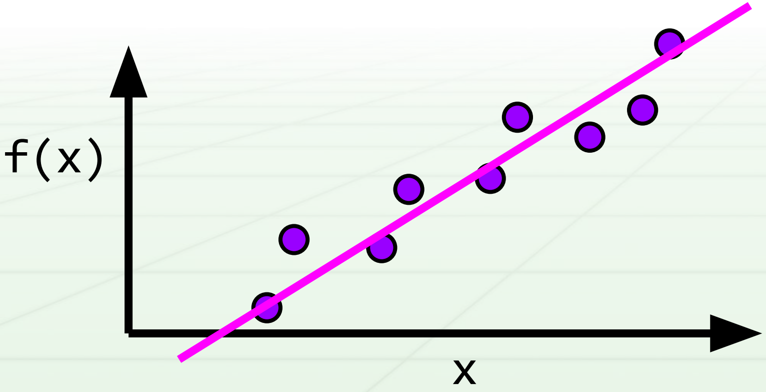
Nearest neighbor

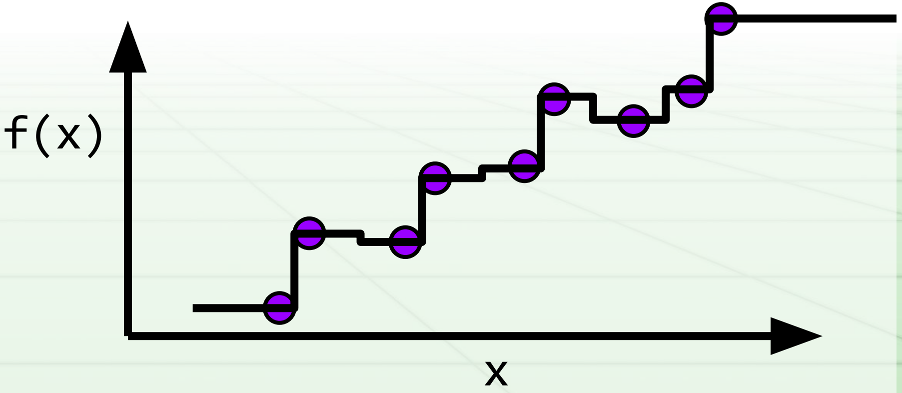
- $f^*(x) = f(x')$ for nearest x' in training set
- Low bias: no assumptions about data
- High variance: very sensitive to training set
- Benefits:
  - Super easy to implement
  - Easy to understand
  - Arbitrarily powerful, esp with lots of data
- Weaknesses:
  - Hard to scale
  - Prone to overfitting to noise



These are examples of *regression*

- Given training data
  - input variables **X**, output variables **Y**
- And new data point x’
- Predict corresponding output variables y’





A different task: *Classification*

- Training data: points associated with a class
  - Also other data about that point
- Example: Does patient have the flu?
  - Binary classification (yes or no)
  - Different types of variables (continuous, discrete)

sore throat	runny nose	nausea	temp	chills	pain	age	days	diagnosis
no	yes	yes	101.3	yes	7	15	5	flu
yes	yes	no	98.8	no	3	74	3	not flu
yes	yes	no	100.1	yes	4	46	4	flu
yes	yes	yes	99.8	yes	6	27	1	flu
yes	no	no	98.4	yes	5	35	2	not flu
yes	yes	yes	99.0	no	3	42	4	not flu

One approach: partitions

- Find best split or splits to data along one variable
- One possibility, Pr(flu) = (temp > 99.5)
  - Pretty accurate on our training data

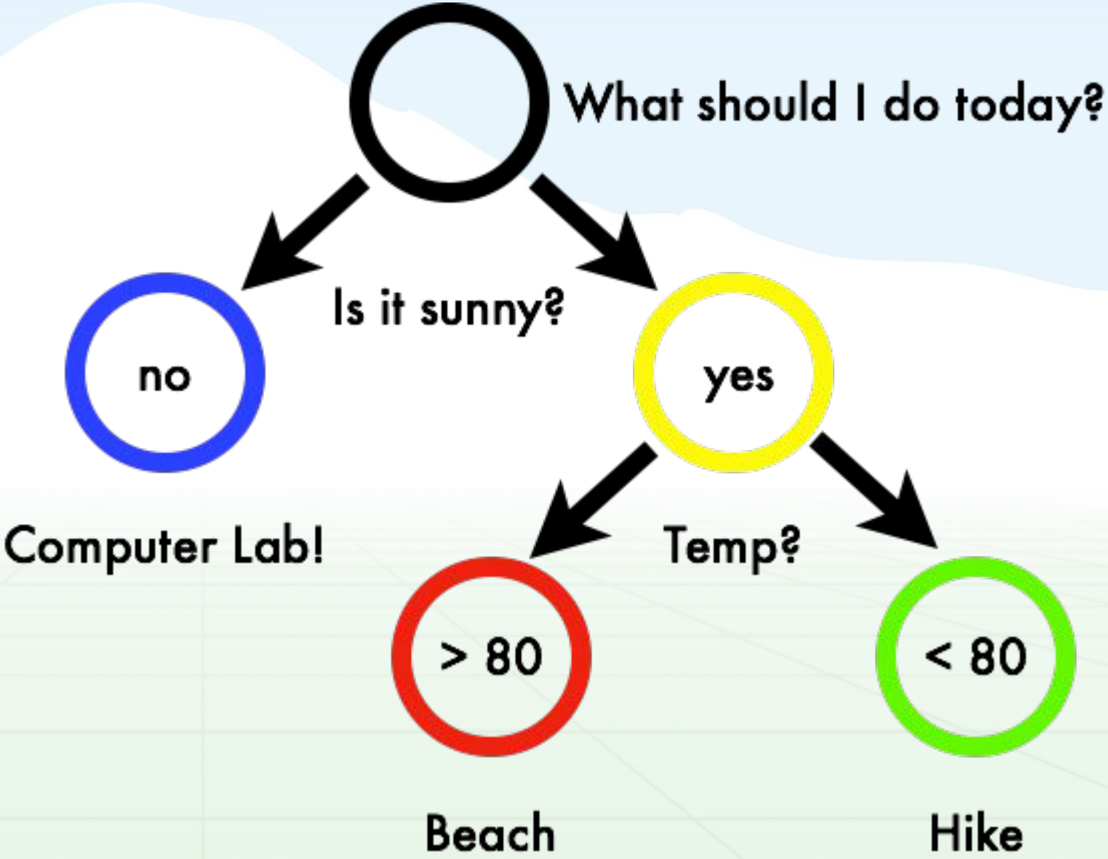
sore throat	runny nose	nausea	temp	chills	pain	age	days	diagnosis
no	yes	yes	101.3	yes	7	15	5	flu
yes	yes	no	98.8	no	3	74	3	not flu
yes	yes	no	100.1	yes	4	46	4	flu
yes	yes	yes	99.8	yes	6	27	1	flu
yes	no	no	98.4	yes	5	35	2	not flu
yes	yes	yes	99.9	no	3	42	4	not flu

Trees: layers of partitions

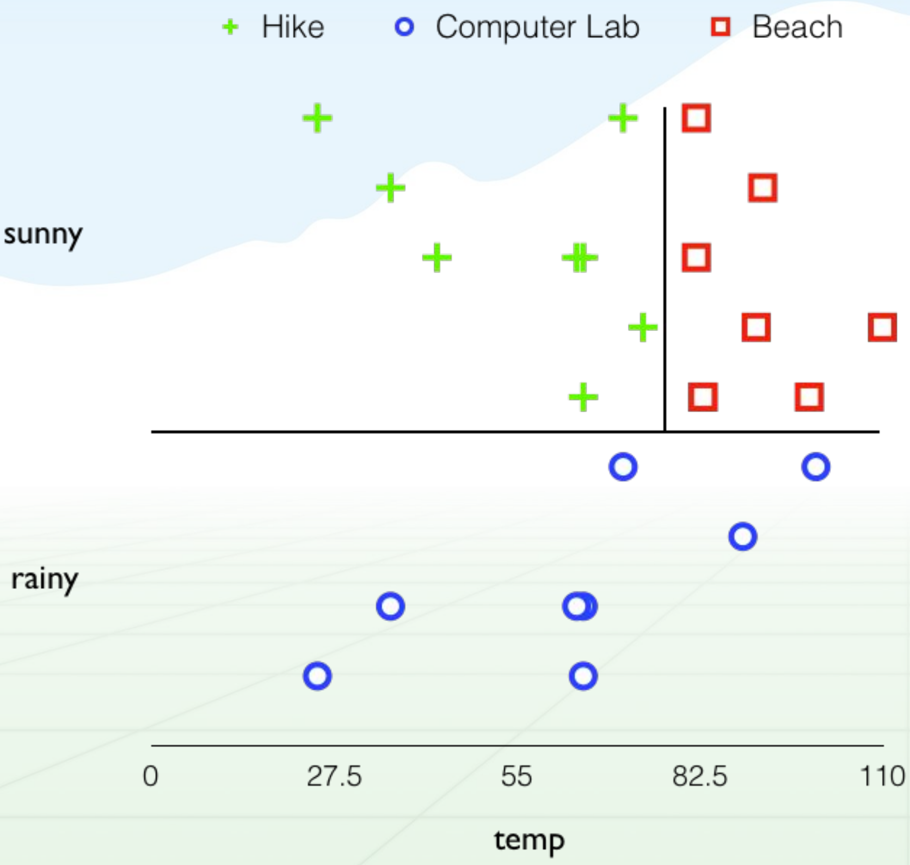
Very simple models

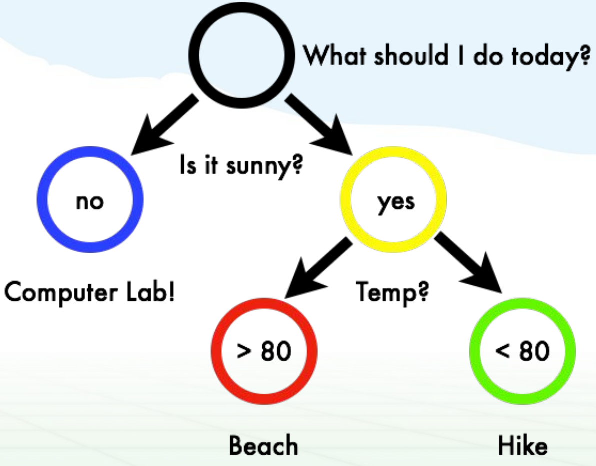
Benefits:

- Interpretable
- Easy to use
- Good for applications
    - E.g. medicine

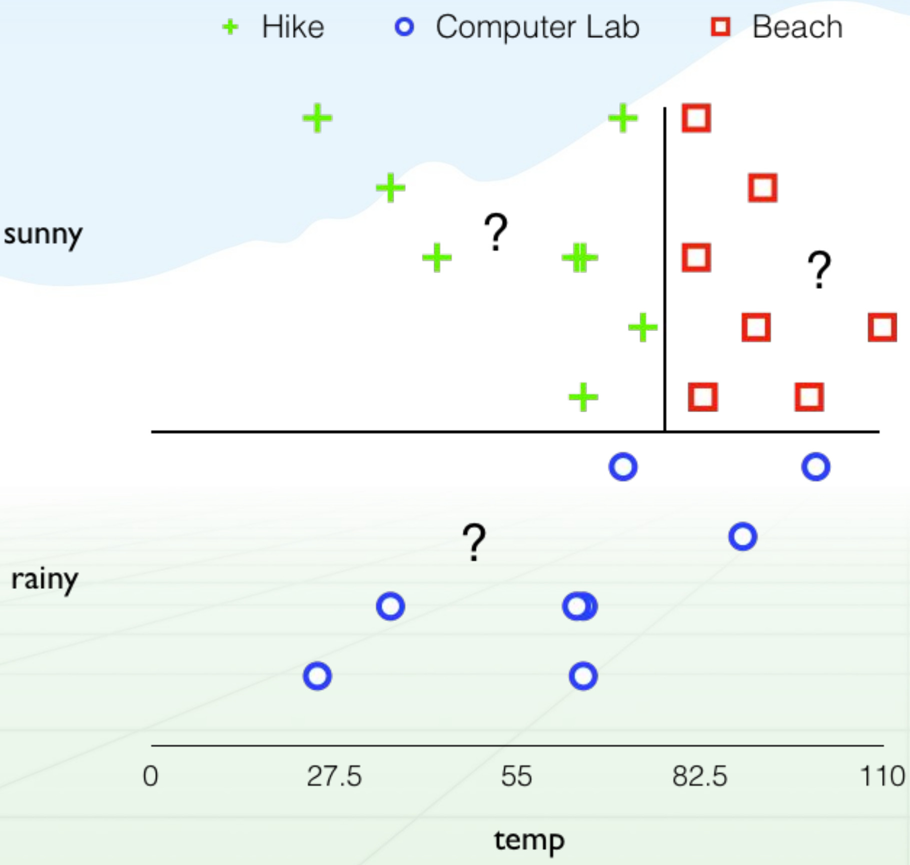


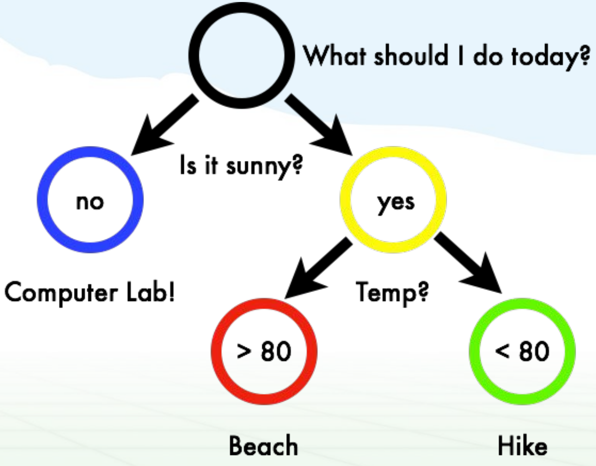
Trees are partitions of data



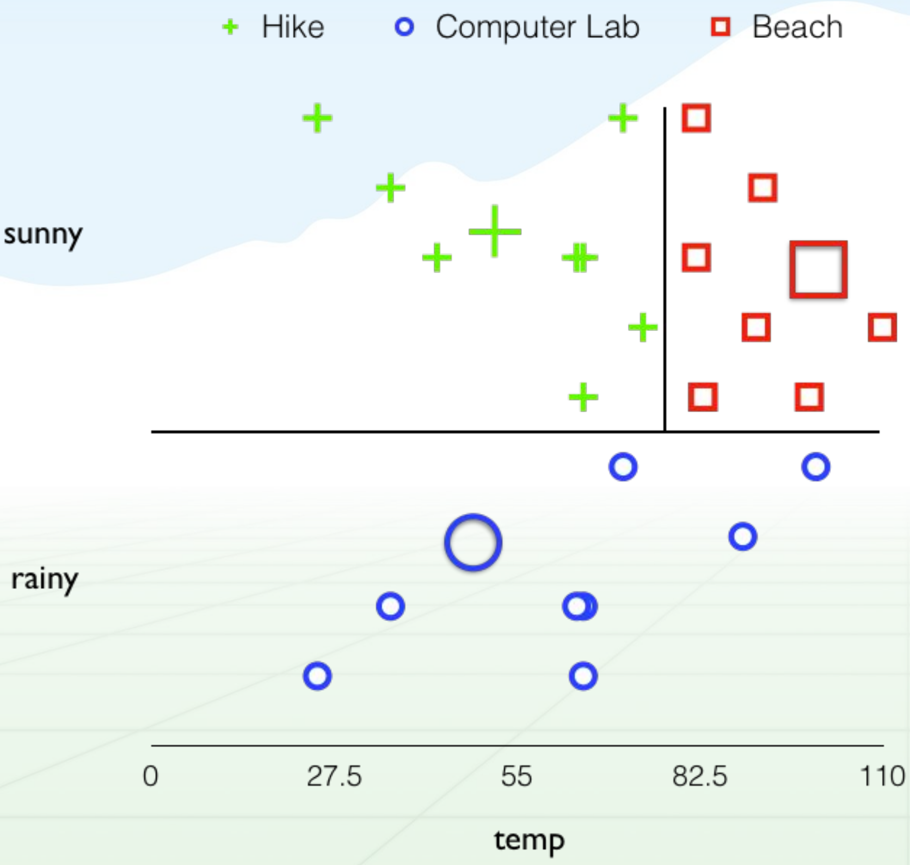


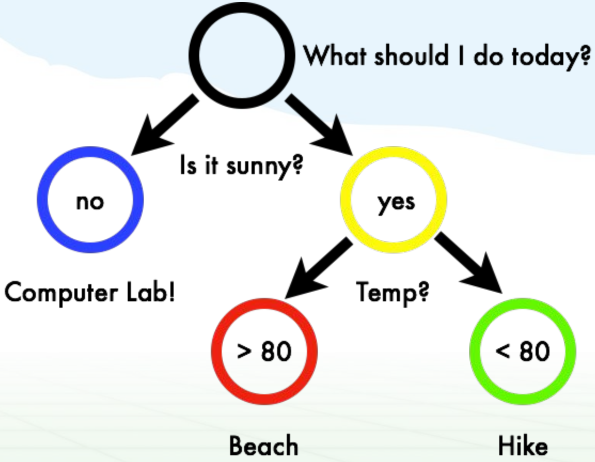
Predict new data based on what region it falls into



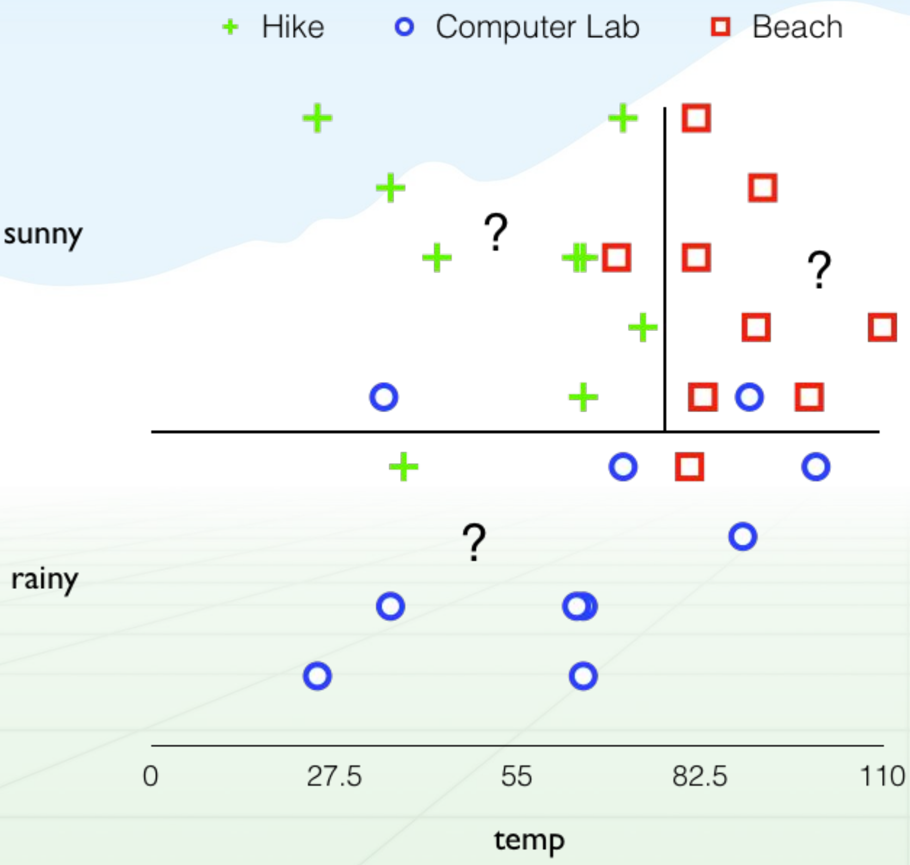


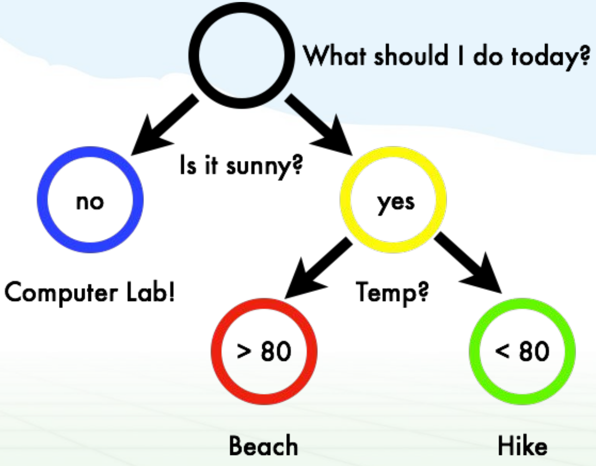
Predict new data based on what region it falls into



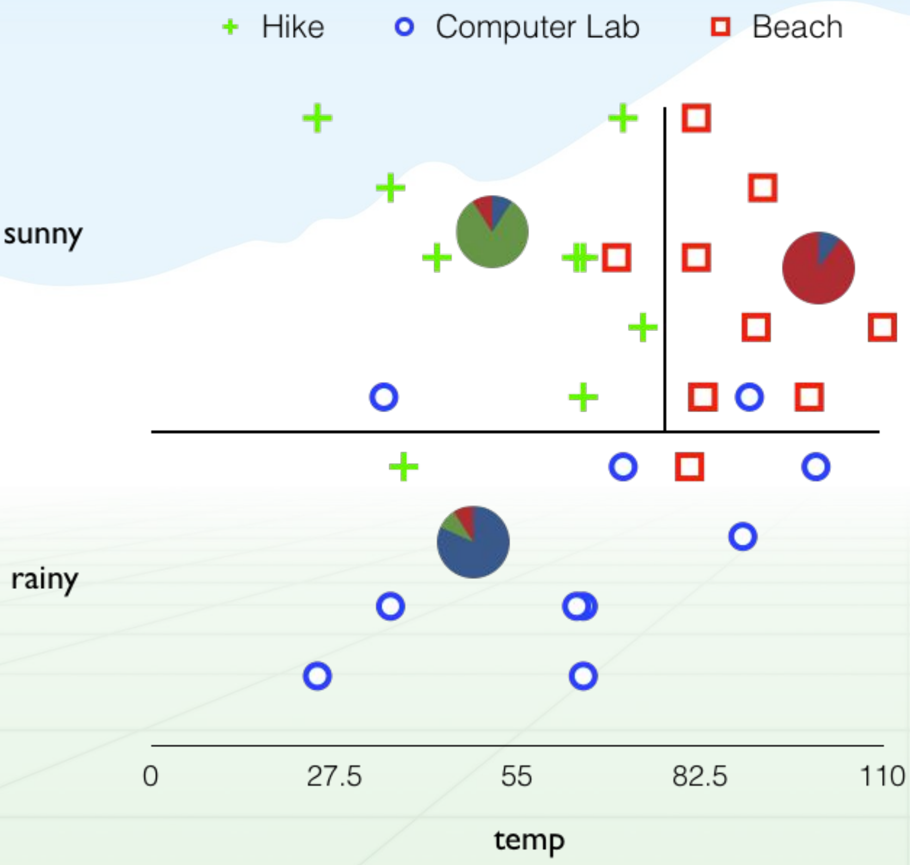


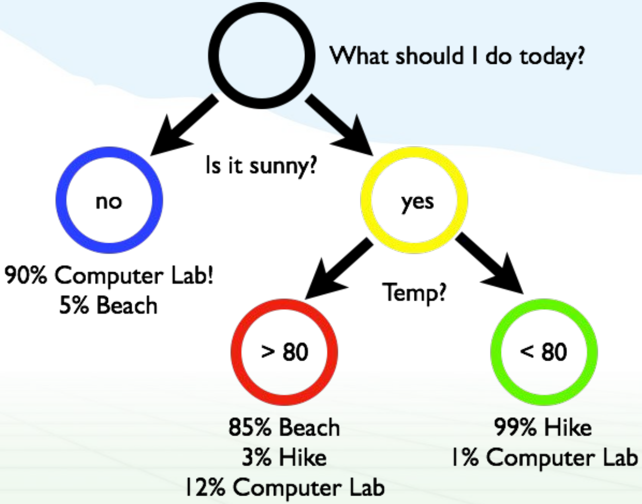
Data might be noisy, use soft assignments





Data might be noisy, use soft assignments





Case study: Viola-Jones Face detection

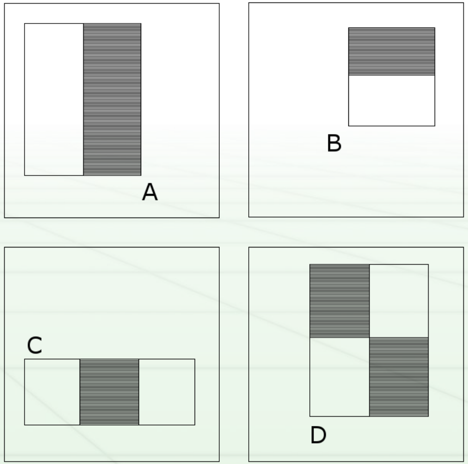
Want it to be very fast and accurate

Run on a camera or cell phone, low cost

Use simple features and simple classifiers

Haar features:

Response = Σ pix in black region - Σ pix in white region



https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

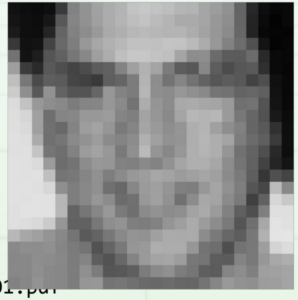
Case study: Viola-Jones Face detection

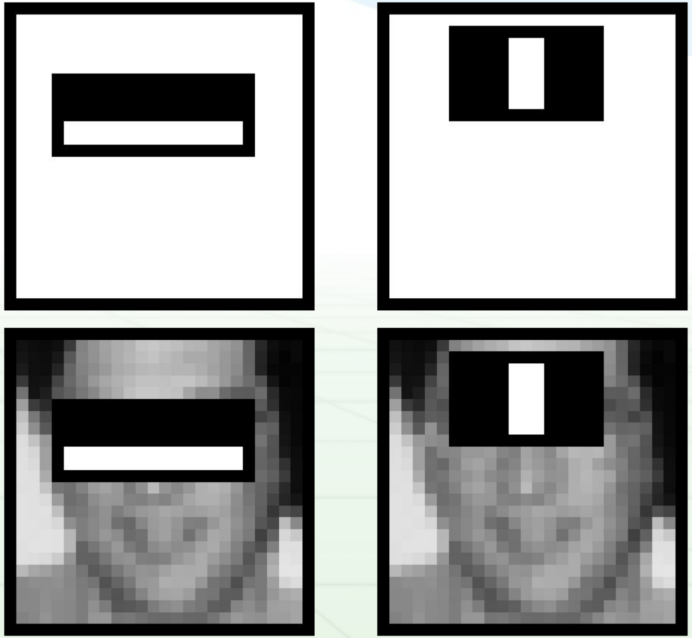
Why do Haar features work?

- Eyes are generally darker than cheeks
- Bridge of nose lighter than eyes
- Etc.

Also, fast to compute!

- *Integral images* - fast sums over regions.

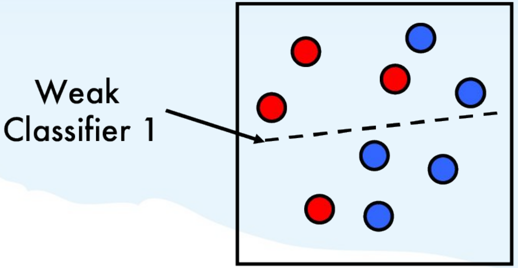




https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

Case study: Viola-Jones Face detection

Classifier: *boosted* partitions



Boosting

Way to make weak classifiers better

Train a weak classifier

https://www.cs.ubc.ca/~lowe/425/slides/13-ViolaJones.pdf

Case study: Viola-Jones Face detection

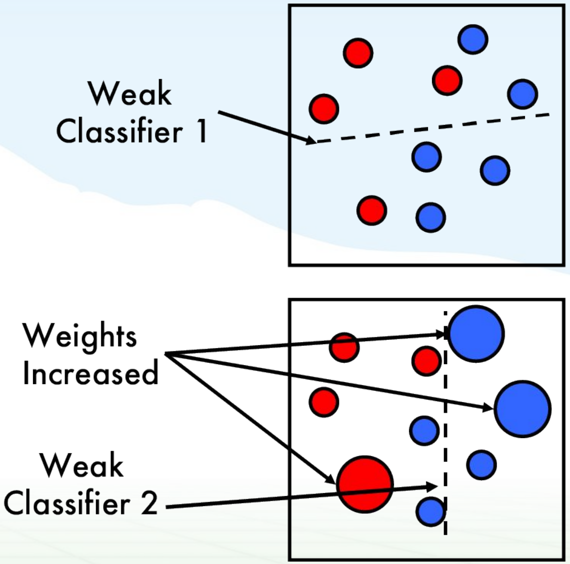
Classifier: *boosted* partitions

Boosting

Way to make weak classifiers better

Train a weak classifier

Reweight data we got wrong, train again



https://www.cs.ubc.ca/~lowe/425/slides/13-ViolaJones.pdf

Case study: Viola-Jones Face detection

Classifier: *boosted* partitions

Boosting

Way to make weak classifiers better

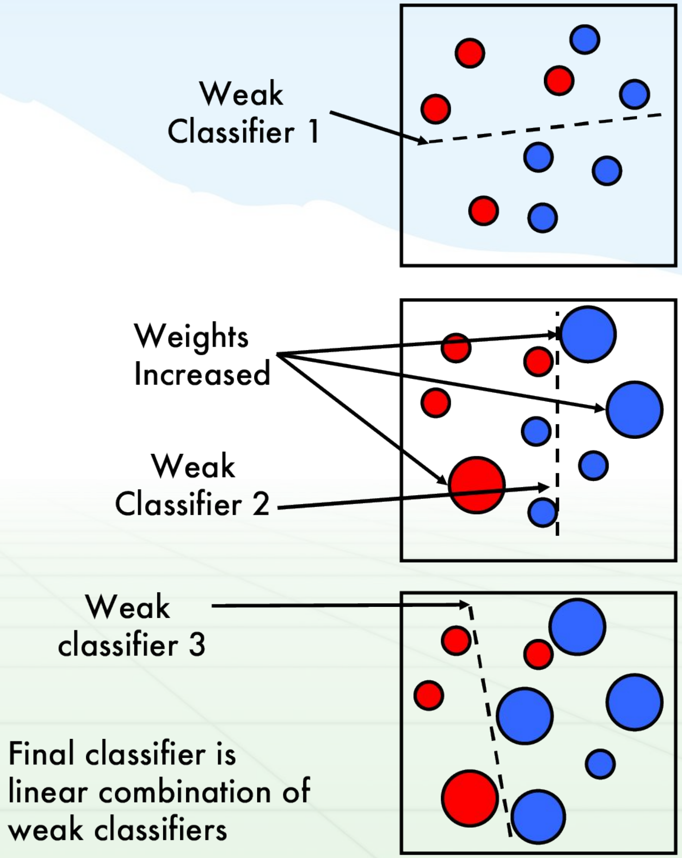
Train a weak classifier

Reweight data we got wrong, train again

...and again

Until you feel like stopping

Final classifier is combination of these



https://www.cs.ubc.ca/~lowe/425/slides/13-ViolaJones.pdf

Case study: Viola-Jones Face detection

Finally, use a cascade of classifiers

1st classifier

Very fast, throws out easy negatives

2nd classifier

Fast, throws out harder negatives

3rd classifier

Slower, throws out hard negatives

Only run slow, good classifiers on hard examples

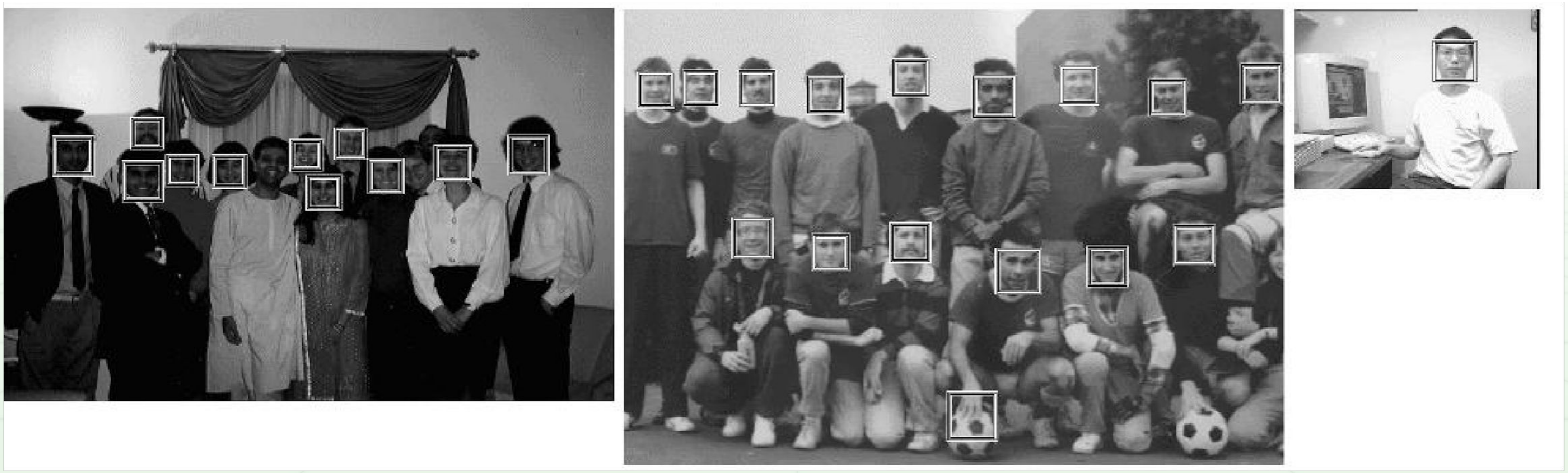
Fast classifier that is still very accurate

Case study: Viola-Jones Face detection

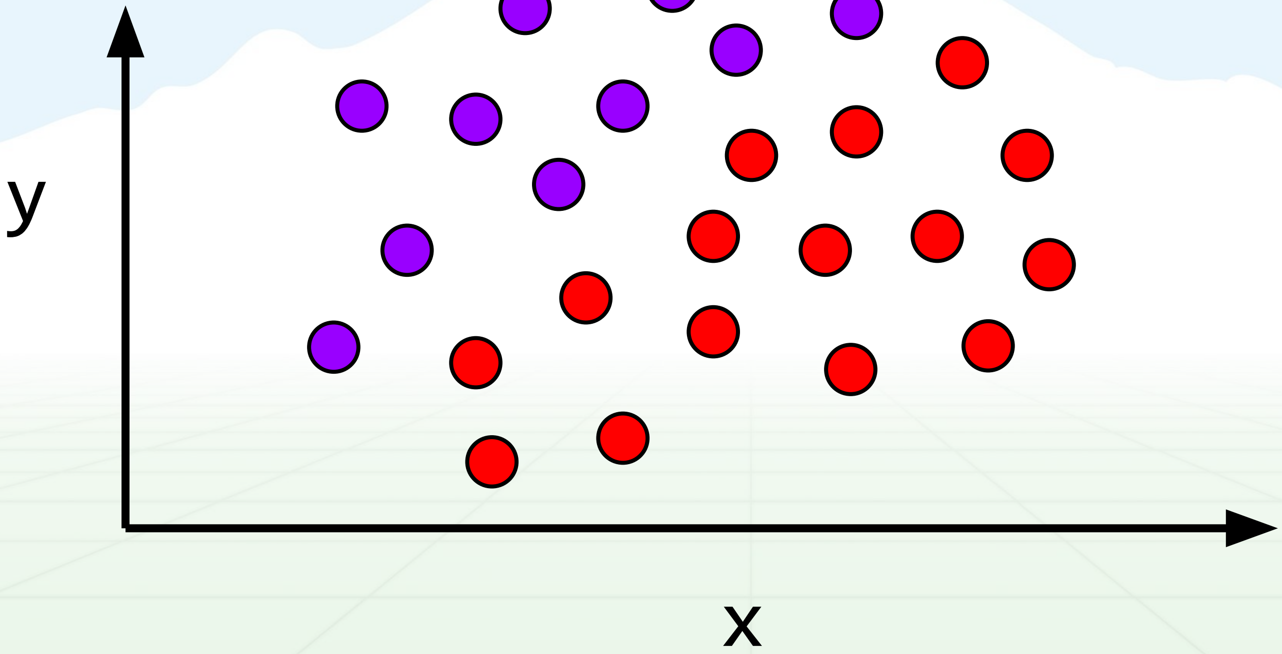
Haar features

Cascade

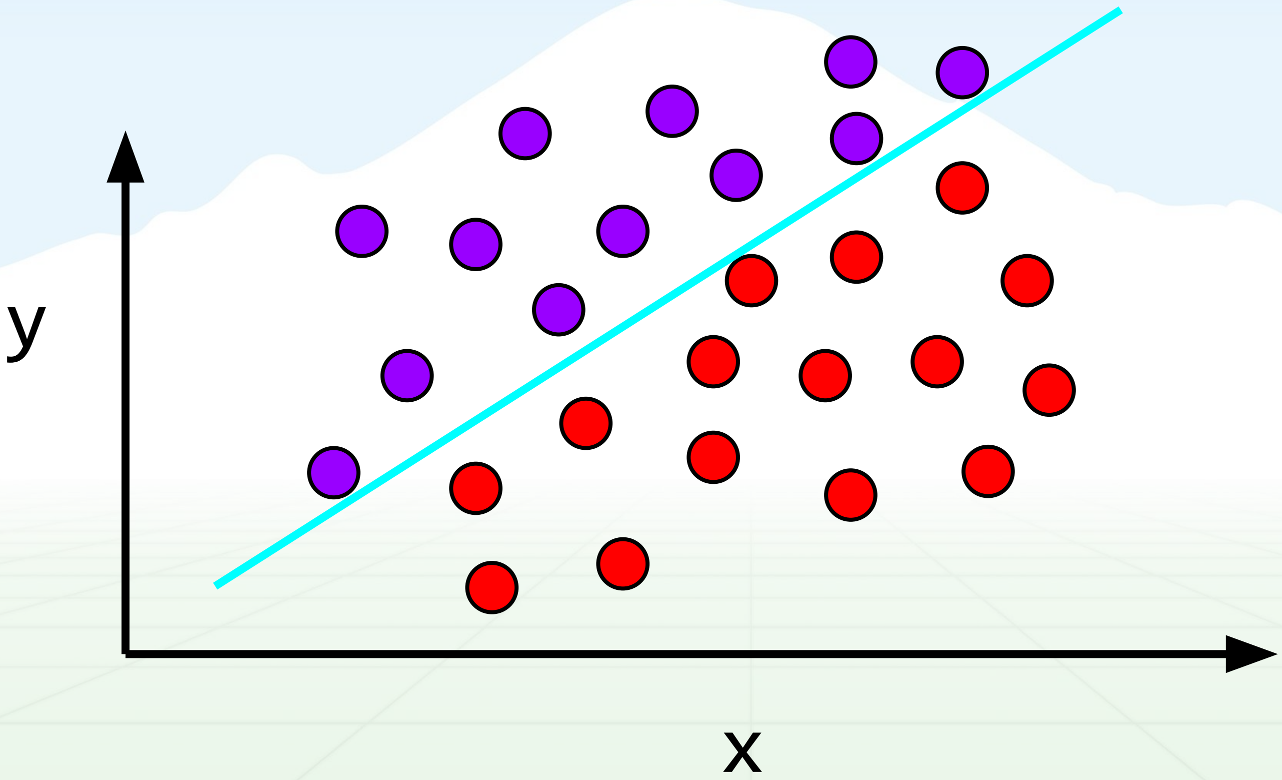
- Of boosted classifiers



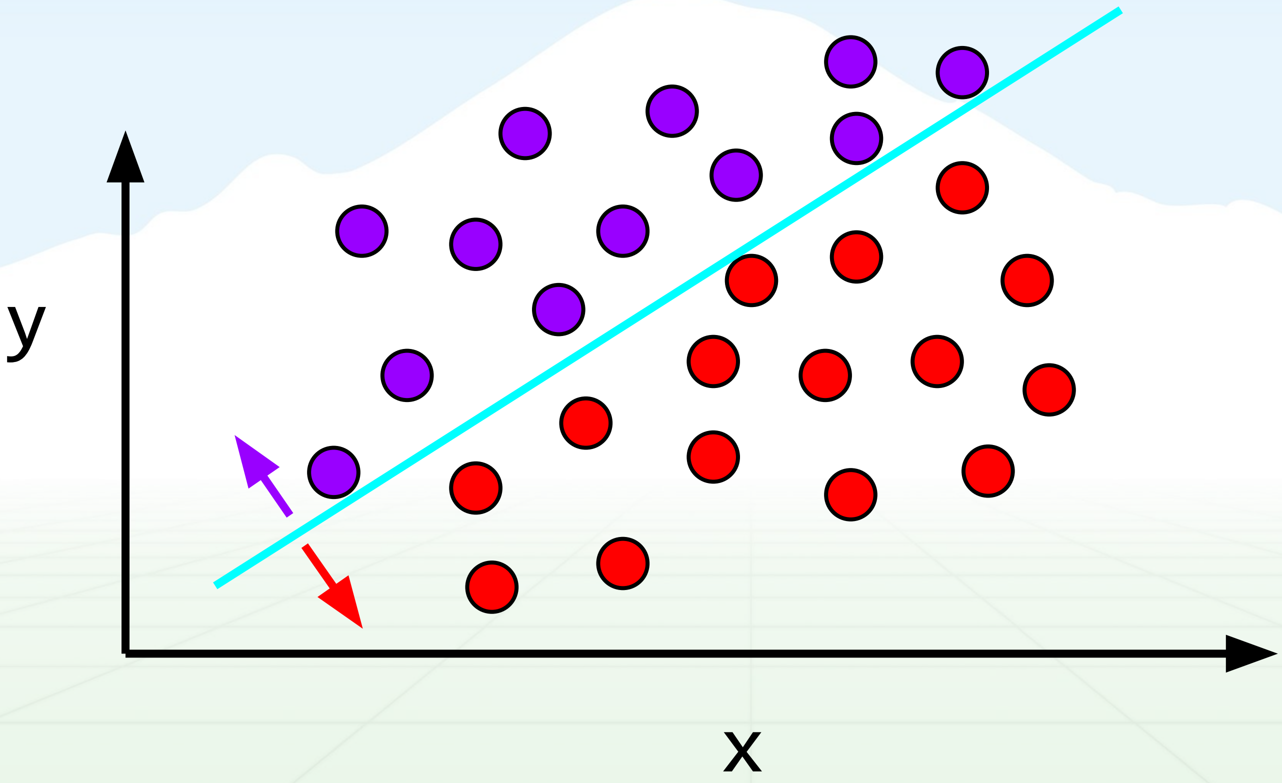
Classification in two dimensions



Classification in two dimensions

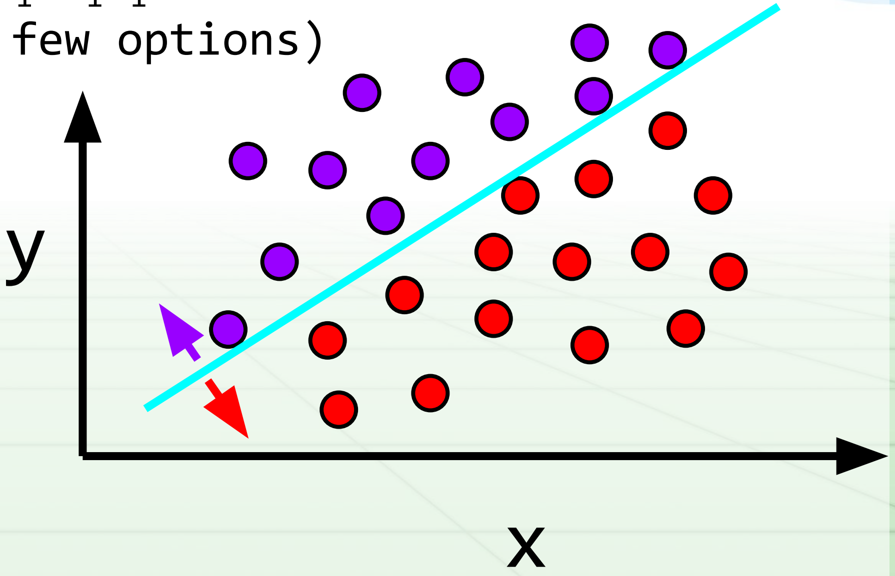


Classification in two dimensions



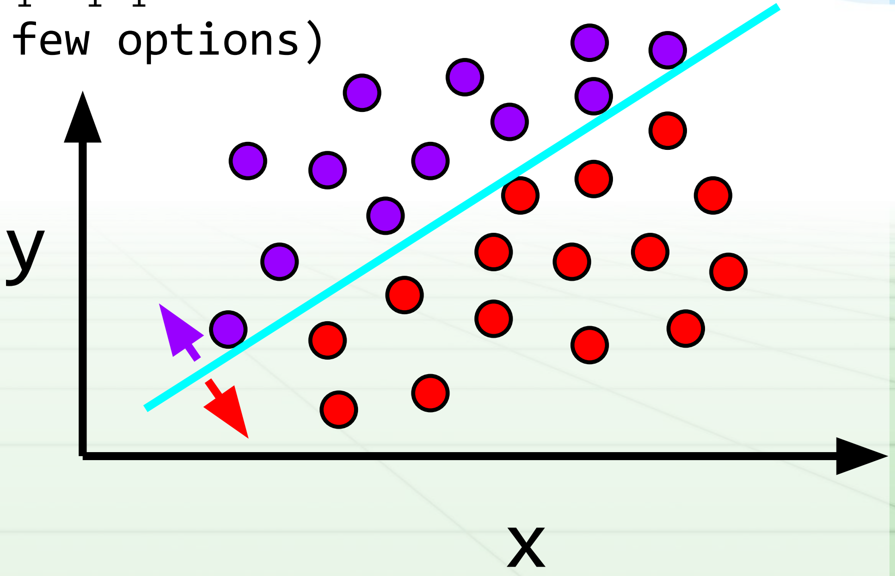
Classification in two dimensions

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



Classification in two dimensions

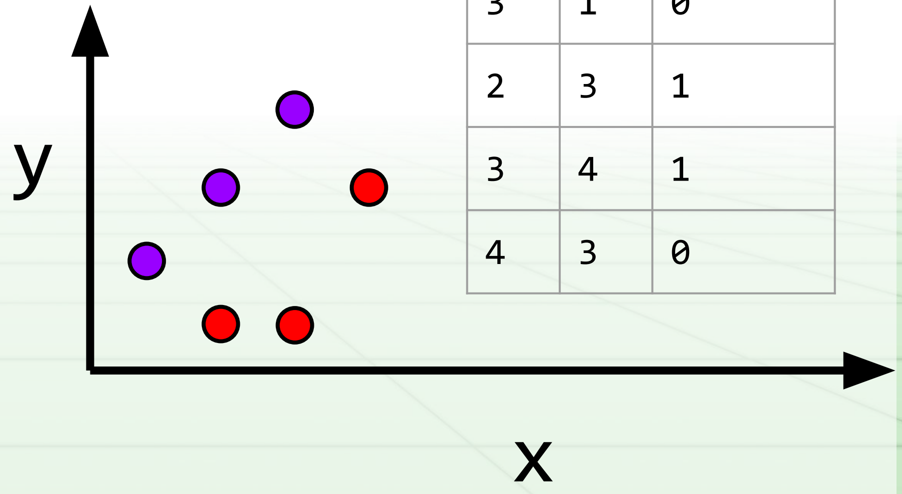
- Linear classifier
  - Given dataset, learn weights **w**
  - Output of model is weighted sum of inputs
  - $P(\text{purple} \mid \mathbf{x}) = f(\mathbf{w} \cdot \mathbf{x}) = f(\Sigma_i(w_i x_i))$
  - Where f is some function (a few options)
  - Typically a bias term:
    - $f(\Sigma_i(w_i x_i) +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



Classification in two dimensions

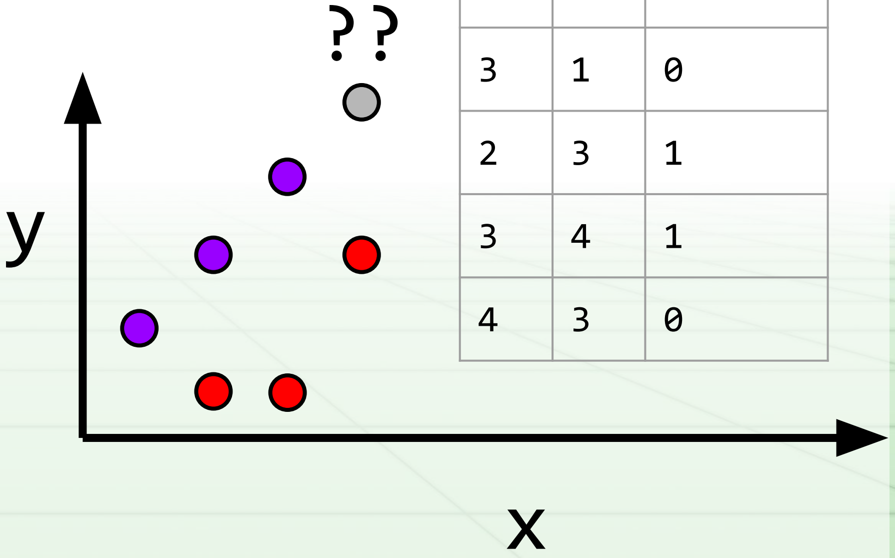
- Simple example:
  - Learned weights: [-1, 1]
  - f is threshold at 0
  -

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

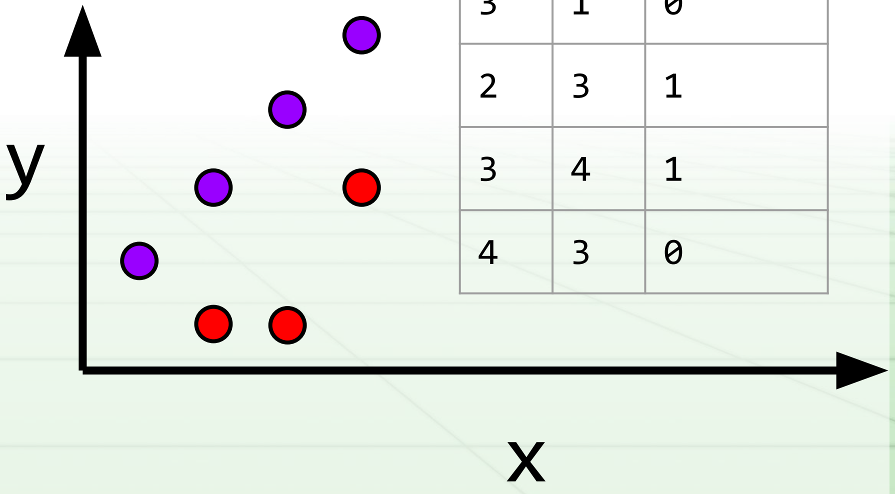
- Simple example:
  - Learned weights: [-1, 1]
  - f is threshold at 0
- New data point (4, 5)
  - (**w**·**x**) = (4,5)·(-1, 1)
    = 4*-1 + 5*1 = 1



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

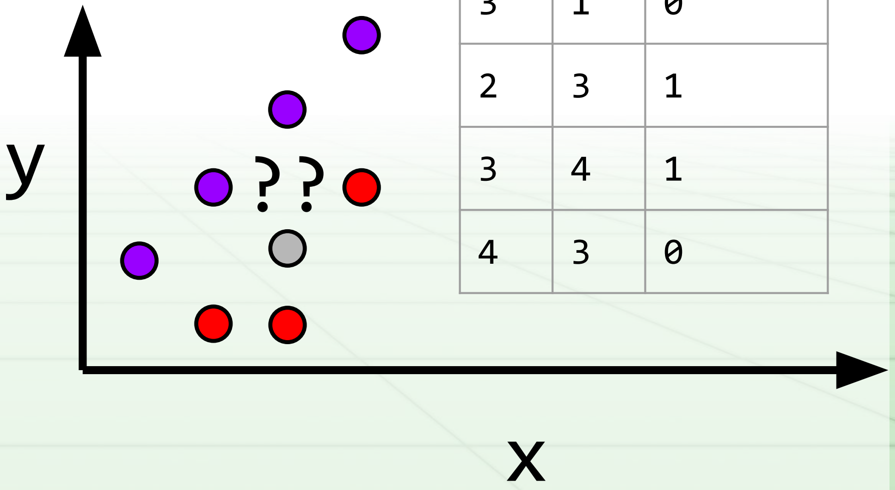
- Simple example:
  - Learned weights: [-1, 1]
  - f is threshold at 0
- New data point (4, 5)
  - $(\mathbf{w} \cdot \mathbf{x}) = (4,5) \cdot (-1, 1)$
    $= 4*-1 + 5*1 = 1$
  - $f(\mathbf{w} \cdot \mathbf{x}) = f(1) = 1$



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)
  - $f(\mathbf{w} \cdot \mathbf{x}) = 1$
- New data point (3, 2)
  - $(\mathbf{w} \cdot \mathbf{x}) = (3,2) \cdot (-1, 1)$
    $= 3*-1 + 2*1 = -1$
  - $f(\mathbf{w} \cdot \mathbf{x}) = f(-1) = 0$
  -

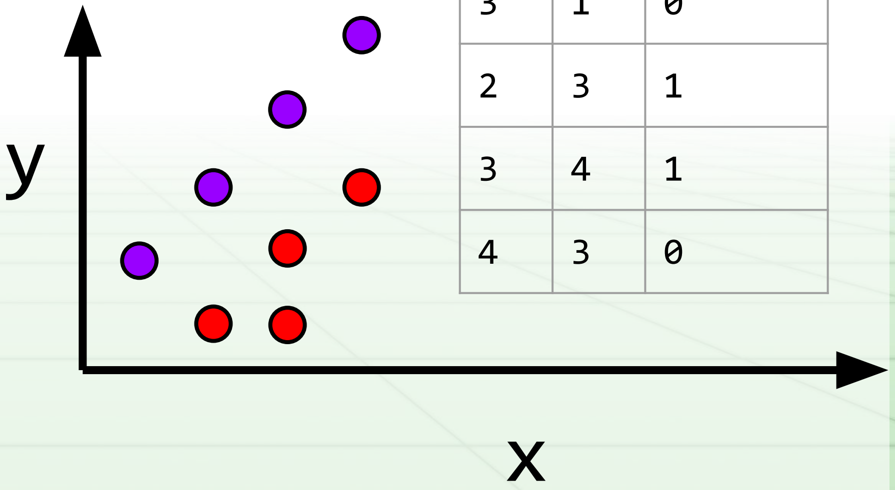


x	y	purple
1	2	1
2	1	0
3	1	0
2	3	1
3	4	1
4	3	0

Classification in two dimensions

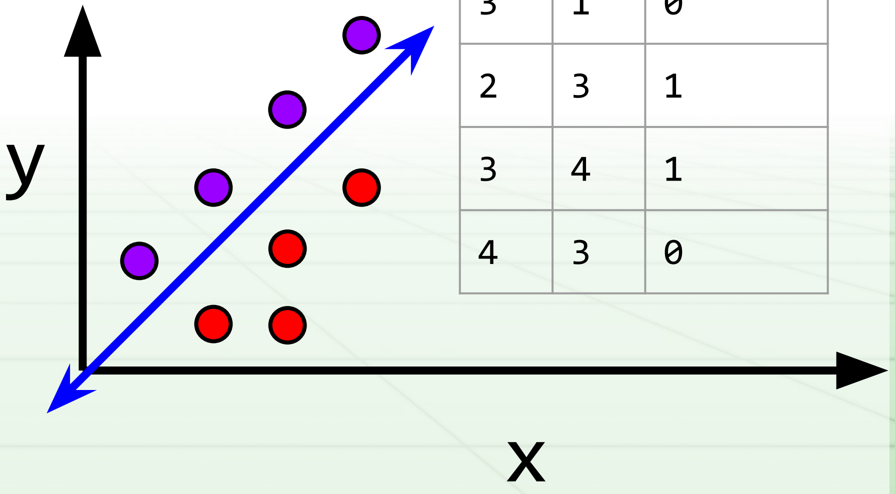
- Simple example:
  - Learned weights: [-1, 1]
  - f is threshold at 0
- New data point (4, 5)
  - $f(\mathbf{w} \cdot \mathbf{x}) = 1$
- New data point (3, 2)
  - $f(\mathbf{w} \cdot \mathbf{x}) = 0$

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

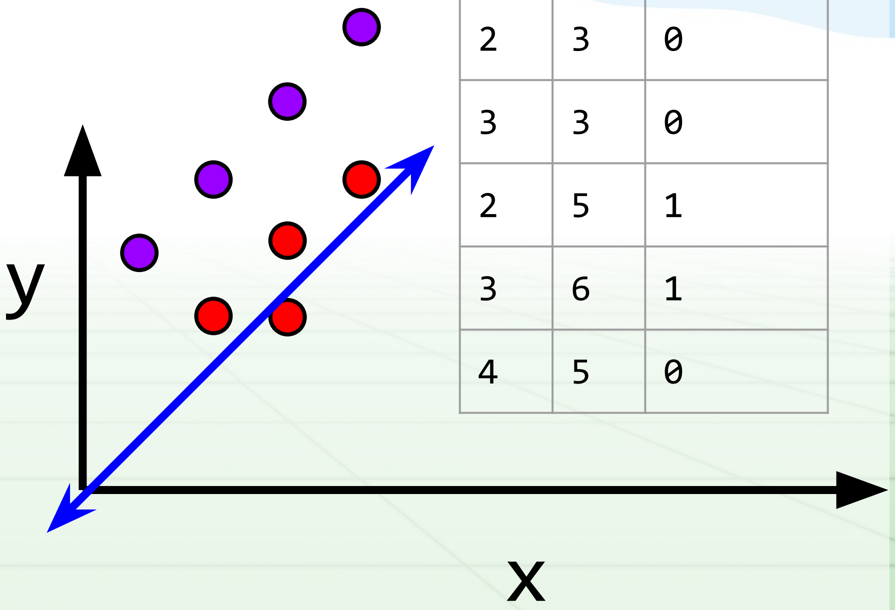
- Simple example:
  - Learned weights: [-1, 1]
  - f is threshold at 0
- New data point (4, 5)
  - f(w·x) = 1
- New data point (3, 2)
  - f(w·x) = 0
- Decision boundary: x=y



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

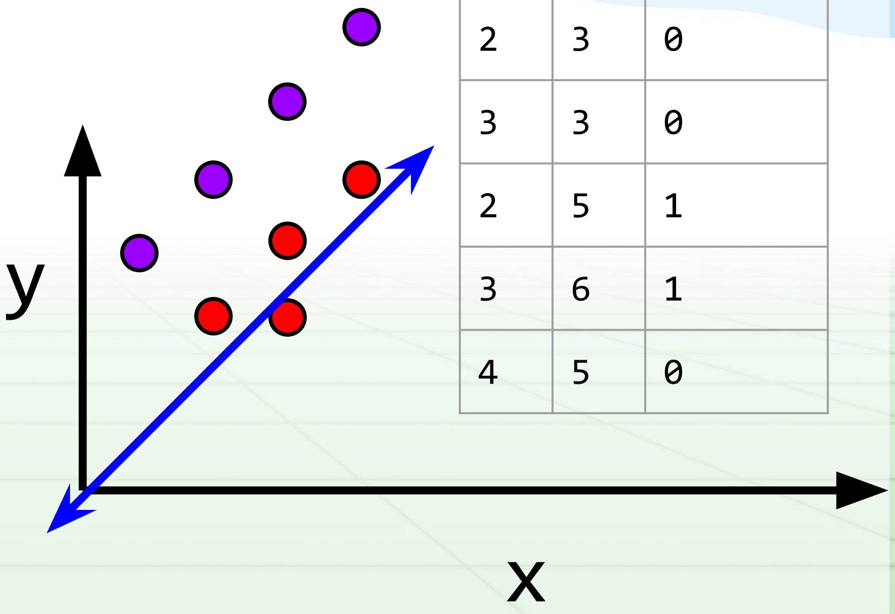


x	y	purple
1	4	1
2	3	0
3	3	0
2	5	1
3	6	1
4	5	0

What if data is shifted up by two?

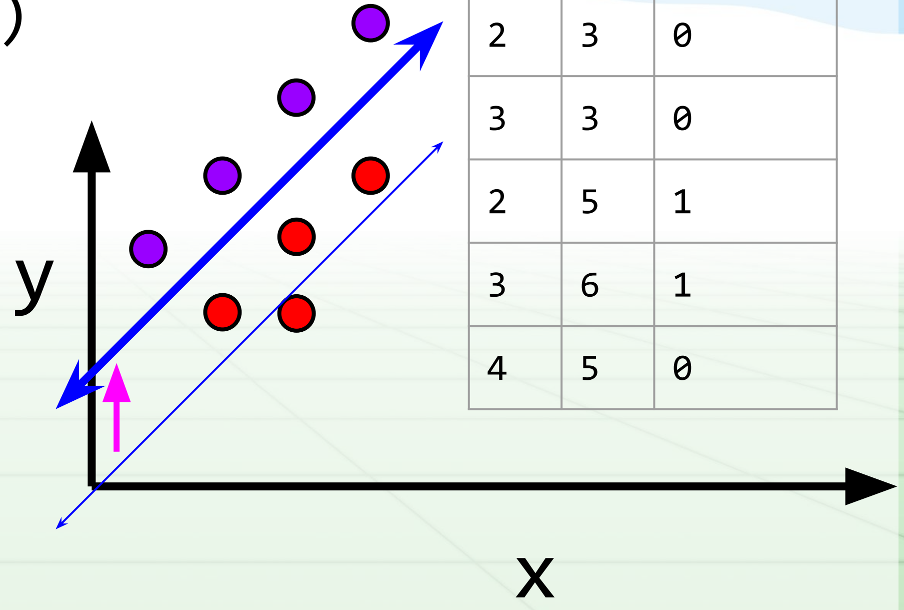
- Need a bias!:
  - Learned weights: [-1, 1, -2]
  - f is threshold at 0

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?

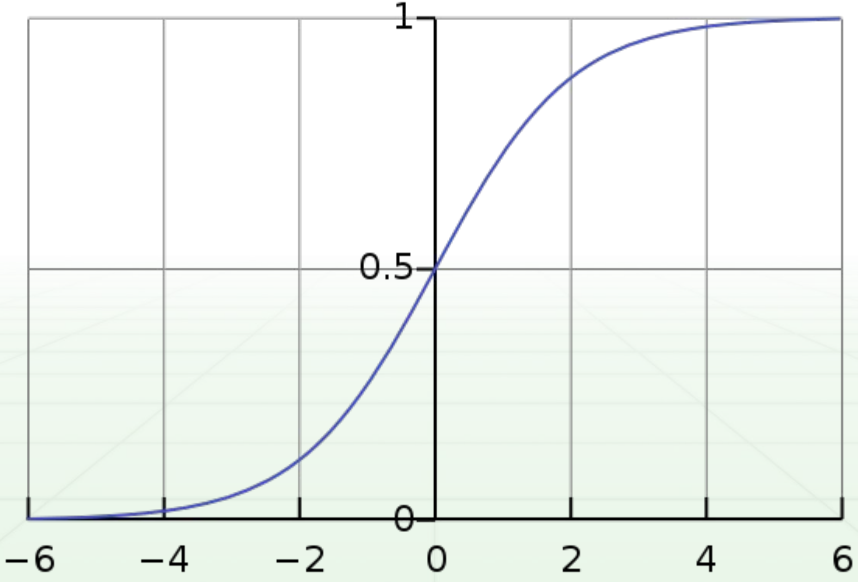
- Need a bias!:
  - Learned weights: [-1, 1, -2]
  - f is threshold at 0
- New data point (4, 7, 1)
  - (**w**·**x**) = (4,7,1)·(-1,1,-2)
    = 4*-1 + 7*1 + 1*-2 = 1
  - f(**w**·**x**) = f(1) = 1



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

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
  - $L(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = \Pr(\mathbf{Y} \mid \mathbf{X},\mathbf{w}) = \prod_i \Pr(\mathbf{Y}_i \mid \mathbf{X}_i,\mathbf{w})$

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 $Y_i = 1$, $\sigma(\mathbf{w}\cdot\mathbf{X}_i)$

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 $Y_i = 1$, $\sigma(\mathbf{w}\cdot\mathbf{X}_i)$
    - If $Y_i = 0$, $1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i)$
  - $L(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = \mathbf{\Pi}_i[(\sigma(\mathbf{w}\cdot\mathbf{X}_i))^{Yi} * (1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))^{(1-Yi)}]$

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 $Y_i = 1$, $\sigma(\mathbf{w}\cdot\mathbf{X}_i)$
    - If $Y_i = 0$, $1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i)$
  - $L(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = \mathbf{\Pi}_i[(\sigma(\mathbf{w}\cdot\mathbf{X}_i))^{Yi} * (1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))^{(1-Yi)}]$
  - 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!
  - $\log L(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = \log \mathbf{\Pi}_i[(\sigma(\mathbf{w}\cdot\mathbf{X}_i))^{Yi} * (1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))^{(1-Yi)}]$
  - $= \Sigma_i \log[(\sigma(\mathbf{w}\cdot\mathbf{X}_i))^{Yi} * (1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))^{(1-Yi)}]$
  - $= \Sigma_i [\mathbf{Y}_i\log(\sigma(\mathbf{w}\cdot\mathbf{X}_i)) + (1-\mathbf{Y}_i)\log(1 - \sigma(\mathbf{w}\cdot\mathbf{X}_i))]$

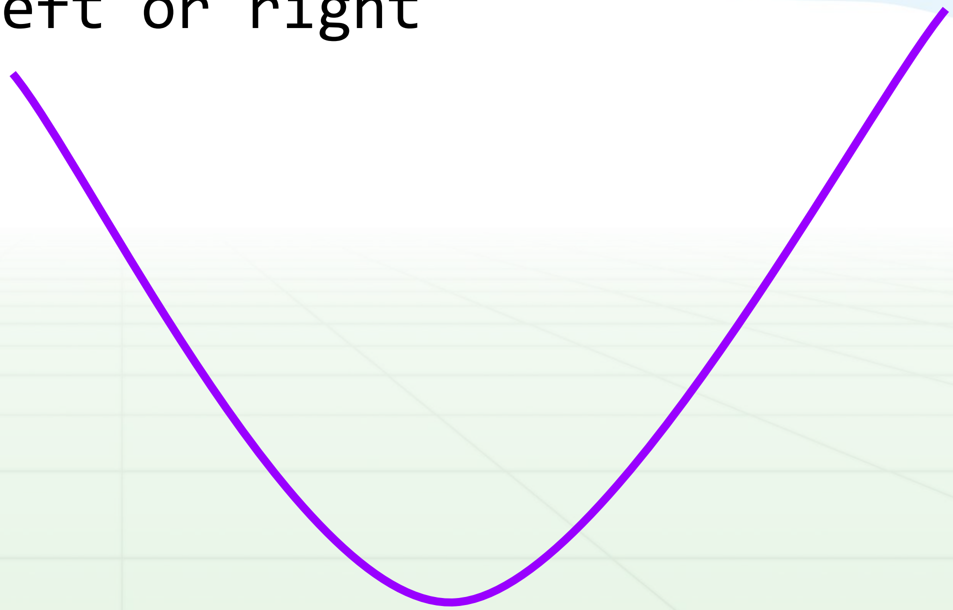
Logistic regression

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

Gradient descent

For some loss function $L(\mathbf{w})$, gradient $\nabla L(\mathbf{w})$ points towards in direction of steepest ascent.

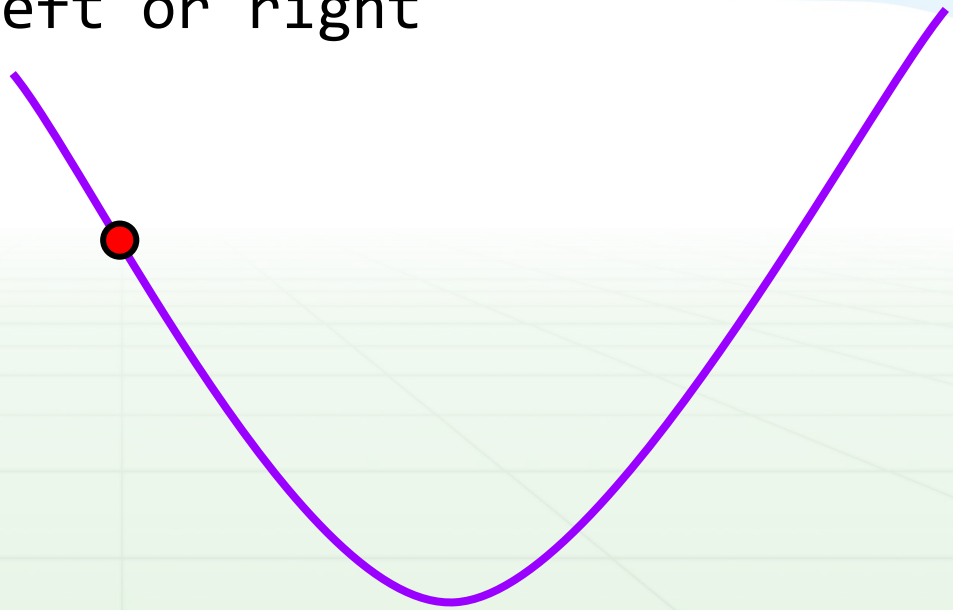
In 1d, either points left or right



Gradient descent

For some loss function $L(\mathbf{w})$, gradient $\nabla L(\mathbf{w})$ points towards in direction of steepest ascent.

In 1d, either points left or right



Gradient descent

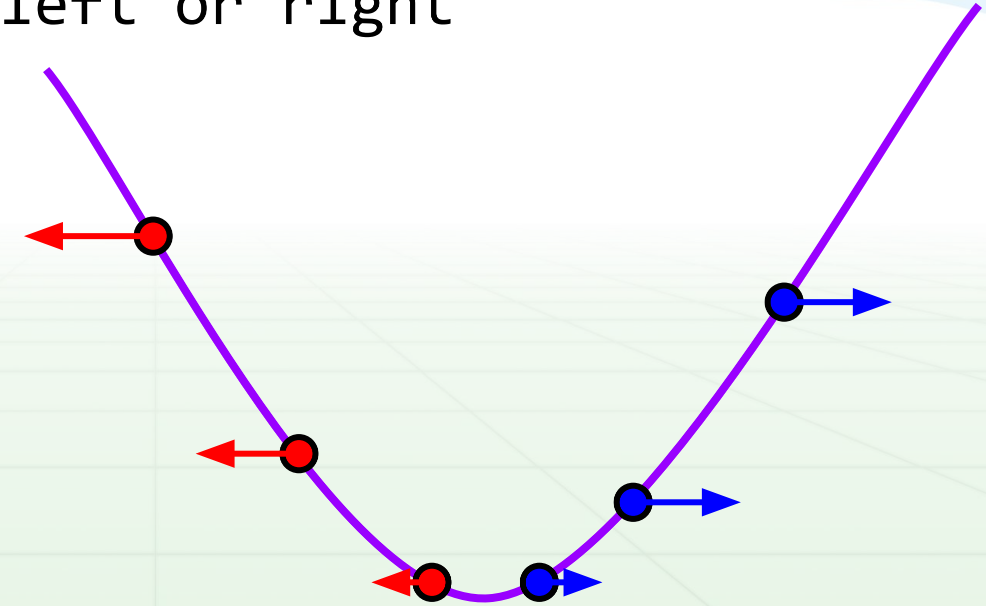
For some loss function $L(\mathbf{w})$, gradient $\nabla L(\mathbf{w})$ points towards in direction of steepest ascent.

In 1d, either points left or right

Gradient descent

For some loss function $L(\mathbf{w})$, gradient $\nabla L(\mathbf{w})$ points towards in direction of steepest ascent.

In 1d, either points left or right



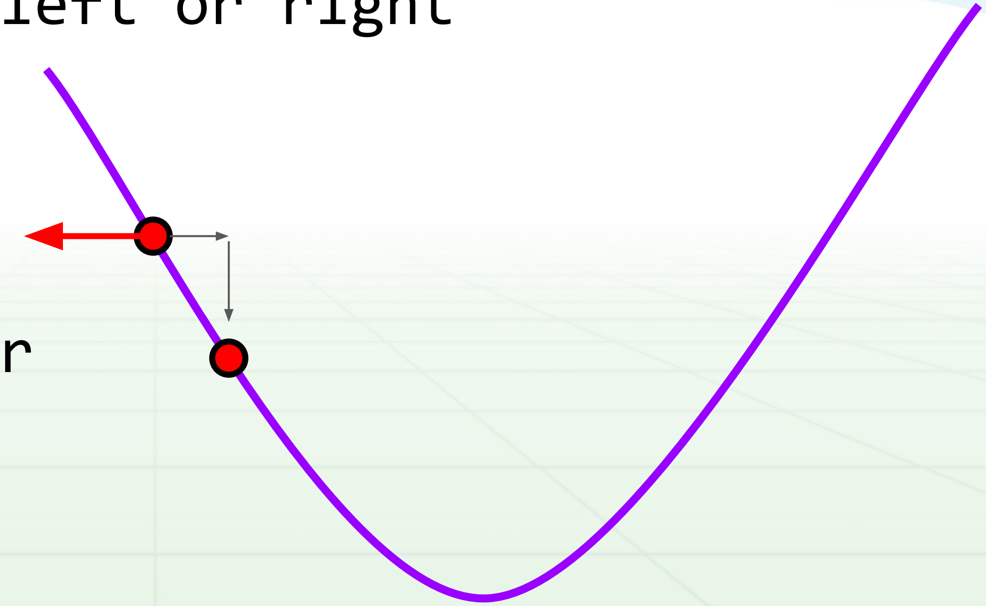
Gradient descent

For some loss function $L(\mathbf{w})$, gradient $\nabla L(\mathbf{w})$ points towards in direction of steepest ascent.

In 1d, either points left or right

Algorithm:

Take derivative
Move slightly in other direction
Repeat



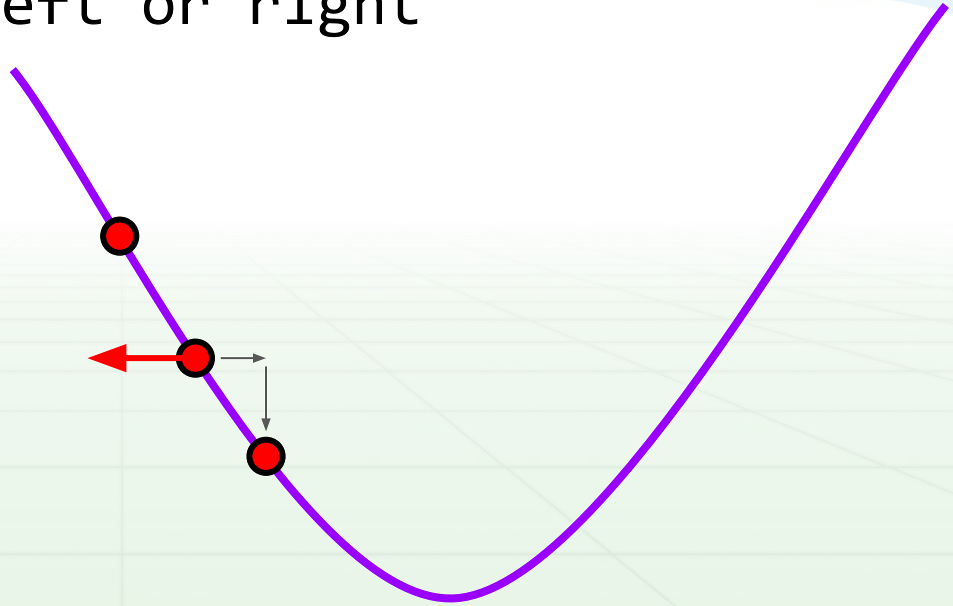
Gradient descent

For some loss function $L(\mathbf{w})$, gradient $\nabla L(\mathbf{w})$ points towards in direction of steepest ascent.

In 1d, either points left or right

Algorithm:

Take derivative
Move slightly in other direction
Repeat

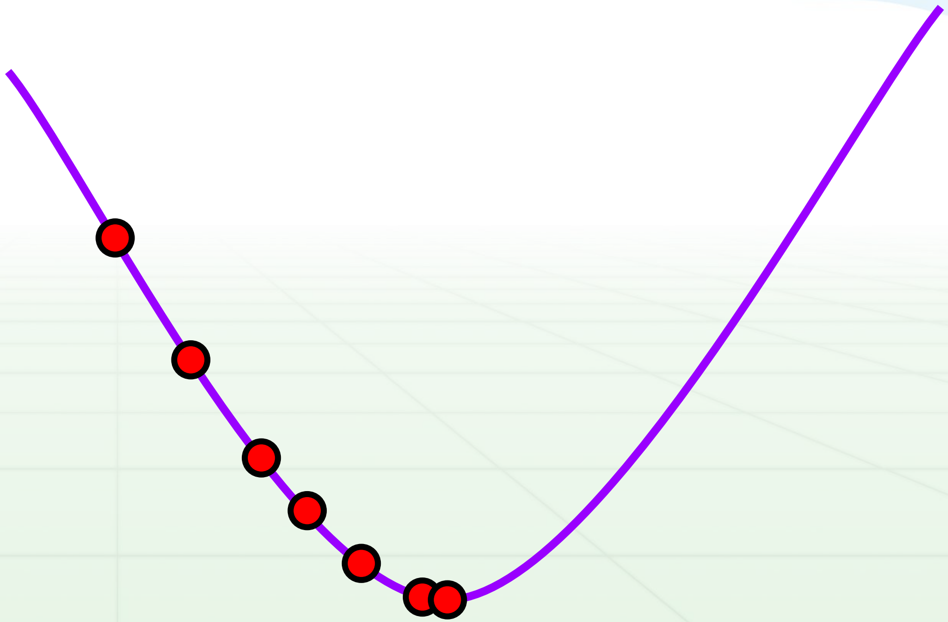


Gradient descent

Algorithm:

Take derivative
Move slightly in other direction
Repeat

End up at local optima

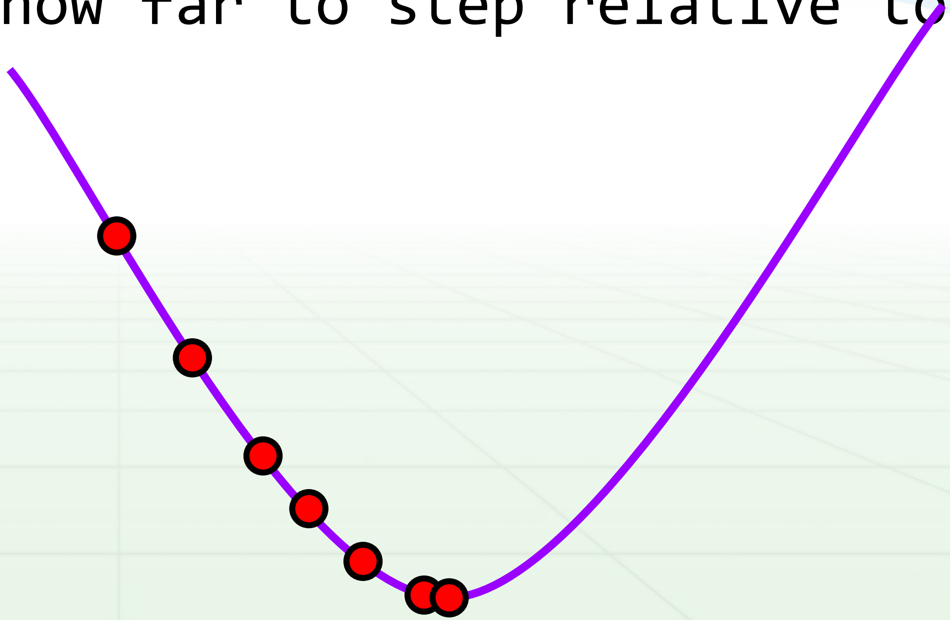


Gradient descent

Formally:

$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla L(\mathbf{w})$

Where η is *step size*, how far to step relative to the gradient



Gradient descent

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

(Aside) Loss vs Likelihood

Sometimes we have a loss function (cost, penalty, etc.) we want to minimize.

Sometimes we have a likelihood (reward, etc.) we want to maximize.

Does it matter?

No!

Just a sign flip, either climbing up the gradient or descending down.

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

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

But wait again:

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

So:

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

Need to take derivatives:

$$\nabla \log L_i(\mathbf{w}) = \nabla Y_i(\mathbf{w}\cdot \mathbf{X}_i) - \nabla \log(1+e^{(\mathbf{w}\cdot X_i)})$$
$$= Y_i \mathbf{X}_i - 1/(1+e^{(\mathbf{w}\cdot X_i)}) * e^{(\mathbf{w}\cdot X_i)} * \mathbf{X}_i = \mathbf{X}_i[Y_i - e^{(\mathbf{w}\cdot X_i)}/(1+e^{(\mathbf{w}\cdot X_i)})]$$
$$= \mathbf{X}_i[Y_i - \sigma(\mathbf{w}\cdot \mathbf{X}_i)]$$

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

SGD with logistic regression

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

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