ECS 171: Machine Learning

Summer 2023

Edwin Solares
easolares@ucdavis.edu

Midterm Review

Dataset

X y

X_1	X_2	X_3	X_4	X_5	X_6	X_7	y
Description	**Guests**	**Seat class**	**Customer ID**	**Fare**	**Age**	**Title**	**Success**
Braund, Mr. Owen Harris; 22	1	3	1	7.25	22	Mr	0
Cumings, Mrs. John Bradley ...	1	1	2	71.3	38	Mrs	1
Heikkinen, Miss. Laina; 26	0	3	3	7.92	26	Miss	1
Futrelle, Mrs. Jacques Heath...	1	1	4	53.1	35	Mrs	1
Allen, Mr. William Henry...	0	3	5	8.05	35	Mr	0
Moran, Mr. James;	0	3	6	8.46	0	Mr	0
McCarthy, Mr. Timothy J; 54	0	1	7	51.9	54	Mr	0

Observations

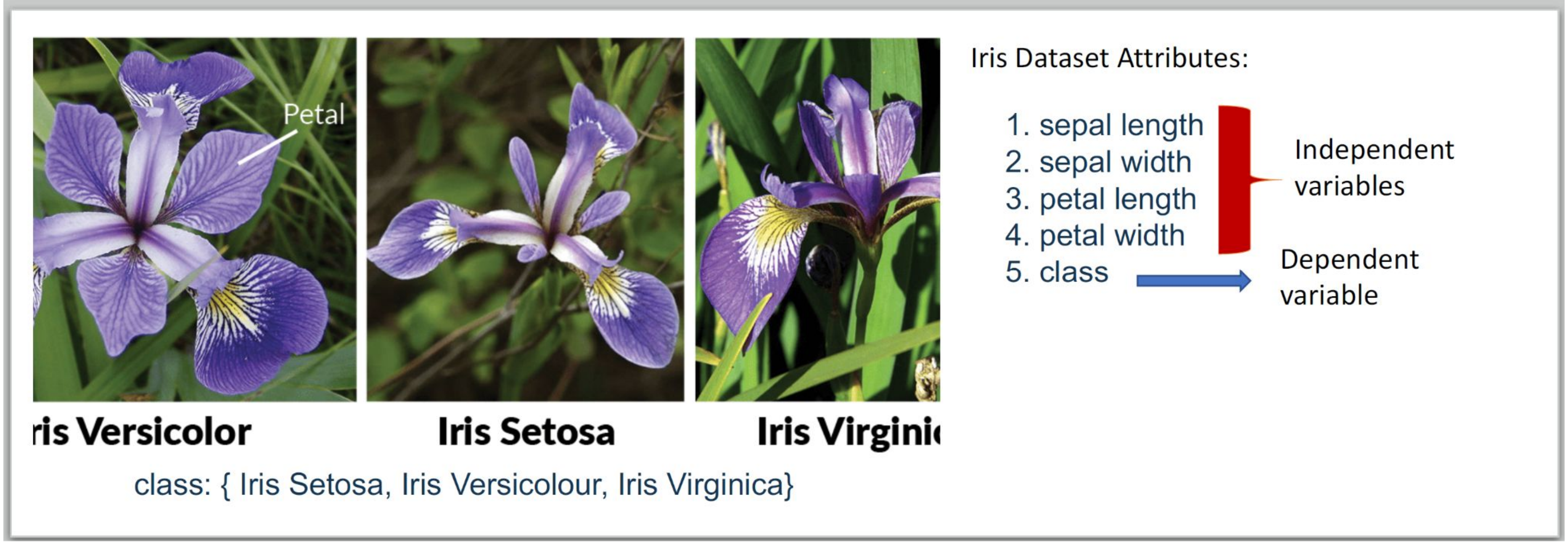
Dataset

X

X_1	X_2	X_3	X_4	X_5	X_6	X_7	y
Description	**Guests**	**Seat class**	**Customer ID**	**Fare**	**Age**	**Title**	**Success**
Braund, Mr. Owen Harris; 22	1	3	1	7.25	22	Mr	0
Cumings, Mrs. John Bradley ...	1	1	2	71.3	38	Mrs	1
Heikkinen, Miss. Laina; 26	0	3	3	7.92	26	Miss	1
Futrelle, Mrs. Jacques Heath...	1	1	4	53.1	35	Mrs	1
Allen, Mr. William Henry...	0	3	5	8.05	35	Mr	0
Moran, Mr. James;	0	3	6	8.46	0	Mr	0
McCarthy, Mr. Timothy J; 54	0	1	7	51.9	54	Mr	0

Observations

The Iris Dataset



f(x) = y

Model type: Classification, Regression, Clustering

Data Visualization

m-by-n matrix

n columns j changes

y

m rows

i changes

Description	Guests	Seat class	Customer ID	Fare	Age	Title	Success
Braund, Mr. Owen Harris; 22	1	3	1	7.25	22	Mr	0
Cumings, Mrs. John Bradley ...	1	1	2	71.3	38	Mrs	1
Heikkinen, Miss. Laina; 26	0	3	3	7.92	26	Miss	1
Futrelle, Mrs. Jacques Heath...	1	1	4	53.1	35	Mrs	1
Allen, Mr. William Henry...	0	3	5	8.05	35	Mr	0
Moran, Mr. James;	0	3	6	8.46	0	Mr	0
McCarthy, Mr. Timothy J; 54	0	1	7	51.9	54	Mr	0

Data Preprocessing

Goal: Using transforms, scale data to similar values

Scaling Data

1. Normalizing - Scaling from 0 to 1
2. Standardization - Scaling data so mean = 0 and standard deviation = 1
   a. Assumes data is already normally distributed.
3. Testing for normality - Shapiro-Wilk Test
   a. Others include Q-Qplot (quantile plots), Histogram plot, Kolmogorov-Smirnov Test

Transform data - Non Constants Transformations

1. Log Transformation
2. Square Root Transformation
3. Cube Root Transformation

Data Preprocessing

Goal: Using transforms, scale data to similar values

Encoding

1. Replace categories with integer values
2. Create new features based on k # of categories containing binary values

Imputing Data

1. Dropping null data
2. Replacing null values with mean, median, most frequent values, etc.
3. More options discussed after Midterm

Normalizing Data

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Used when data is not normally distributed

Benefits:

- Faster processing for GD methods
- Allows you to view actual importance to predicted values using weights

Implementation:

- MinMax Normalization

Standardizing Data

$$z = \frac{x - \mu}{\sigma}$$

$\mu =$ Mean
$\sigma =$ Standard Deviation

Used when data is normally distributed

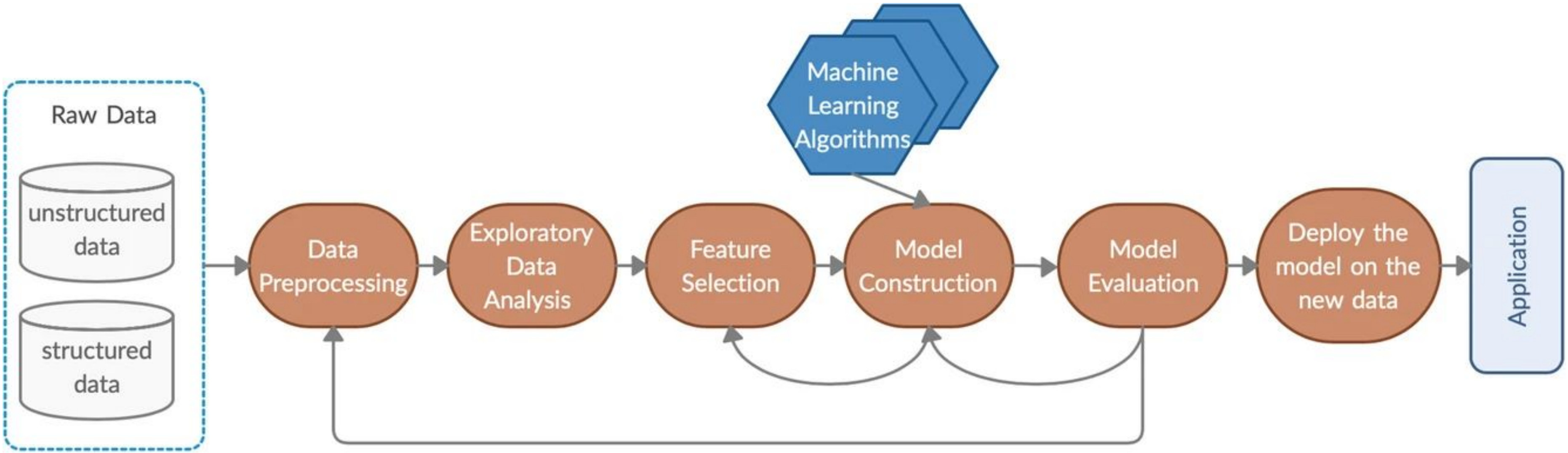
Benefits:

- Much faster processing for GD methods
- Allows you to view actual importance to predicted values using weights

Implementation:

- Z-score standardization

Machine Learning Pipeline



Machine Learning Definition

Supervised Machine Learning

- Labeled data set
- Good for prediction
- Example: Classification

Unsupervised Machine Learning

- Unlabeled data set
- Good for data exploration and association rule discovery
- Example: Clustering

Reinforcement Learning

- Interacts with its environment producing actions and discovers errors or rewards through trial and error search.
- Example algorithm: Q-Learning

Supervised Learning

Predicting known labels

Classification

Regression

Naive Bayes

Support Vector Machines

Neural Nets

Decision Trees

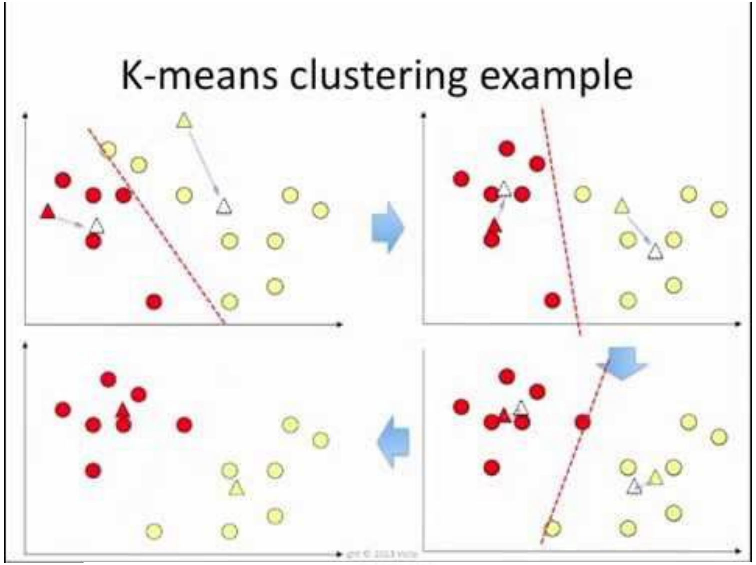
Unsupervised Learning

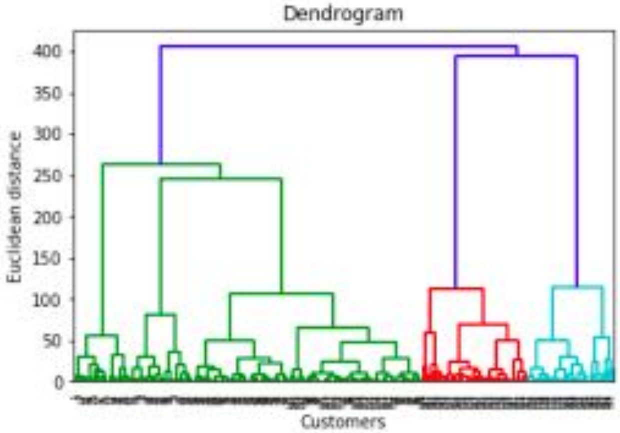
Knowledge Discovery

- Data Exploration
- Descriptive Task
- Un-labeled Dataset
- Common Algorithms

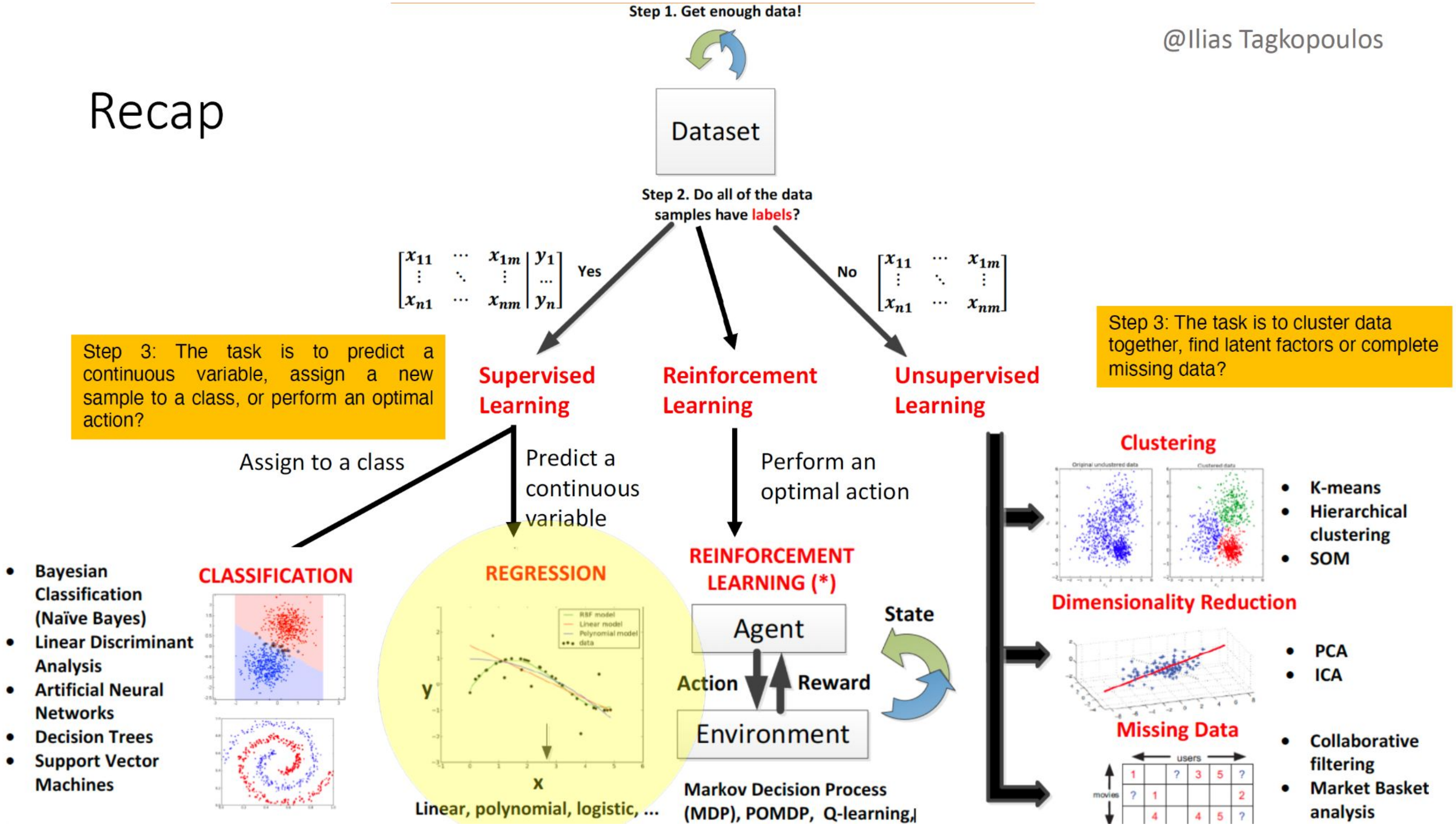
Clustering

- Example: K-means Clustering,
- Hierarchical Clustering, Self-
- Organizing Map
- Association Rule Discovery





What is Machine Learning: Recap



Supervised Learning Generalized

Find a relationship between
X (input) and **y** (output).

Goal: find ***f(X)* = *ŷ*** → ***y***

Predicting sales for a particular product

Data set Description

- Attribute(s) of the data set (**X**) includes
  - advertising budget (dollar value)
- Output **y** i.e., the class attribute
  - sales in thousands of units

Advertisement budget
(**independent variable**) ***X***

Output sales
(**dependent variable**) ***y***

Prediction of Output Sales
(**dependent variable**) ***ŷ***

Find an approximate ***y***
We will call ***ŷ***.
Model maps ***f(X)* = *ŷ*** → ***y***
For all seen ***X*** and unseen ***X***

Supervised Learning Generalized

Find a relationship between
X (input) and **y** (output).

Predicting sales for a particular product

Data set Description

- Attribute(s) of the data set (**X**) includes
  - advertising budget (dollar value)
- Output **y** i.e., the class attribute
  - sales in thousands of units

Goal: find $\boldsymbol{f(X) = \hat{y} \rightarrow y}$

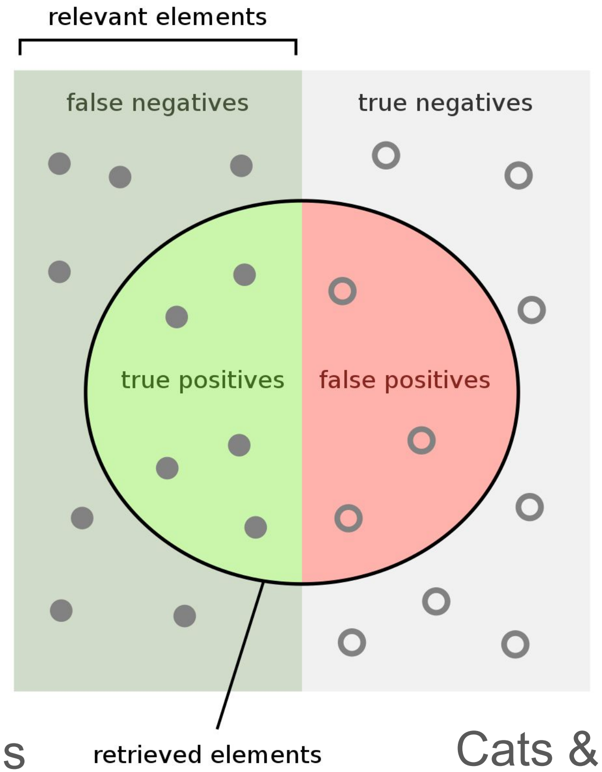
Advertisement budget
(**independent variable**) ***X***

Prediction of Output Sales
(**dependent variable**) $\boldsymbol{\hat{y}}$

Output sales
(**dependent variable**) ***y***

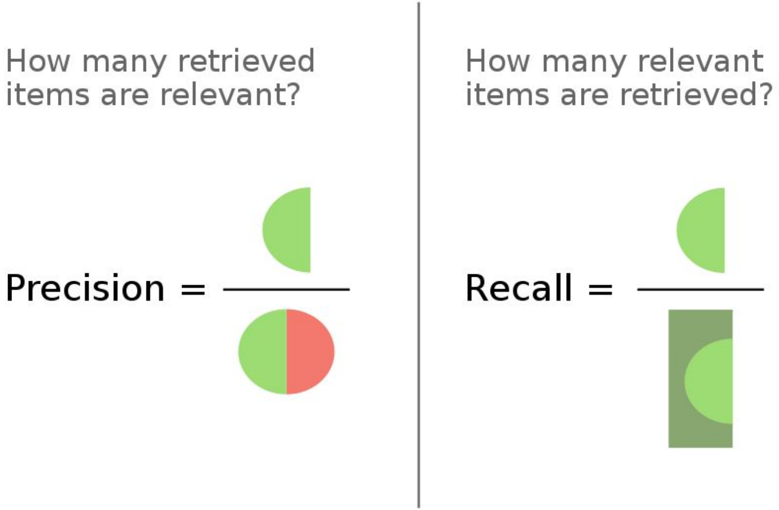
Find an approximate ***y***
We will call $\boldsymbol{\hat{y}}$.
Model maps $\boldsymbol{f(X) = \hat{y} \rightarrow y}$
For all seen ***X*** and $\boxed{\text{unseen } \boldsymbol{X}}$

Machine Learning Evaluation Metrics (wiki)



Dogs

Cats & Donkeys



$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Machine Learning Evaluation Metrics

TP,TN, FP, FN (True +, True -, False +, False -)

Precision and Recall

Receiver operating characteristic (ROC) curve and Area under curve (AUC)

Accuracy

F1 Score

- https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative
- https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall
- https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc
- https://developers.google.com/machine-learning/crash-course/classification/accuracy
- https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Cost Function

When **training** the model, the goal is to **minimize** the **error** and **update** the model **coefficients** to achieve the **best fit** line.

Error is the **difference between predicted value** (Y) generated by the model and the **class attribute value**.

Cost function L is used to **measure the error**:

$$L = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

Observed value   Predicted value

Calculating Error

Find an approximate ***y***
We will call ***ŷ***.
Model maps ***f(X)* = *ŷ* → *y***
For all seen ***X*** and $\boxed{\text{unseen } \boldsymbol{X}}$

$$L = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

Observed value   Predicted value

Expands to $w_1x_1 + w_0$

$$\sum_{i=1}^{m} (y_i - w\ x_i)^2$$

Observations - Predictions

Cross Validation

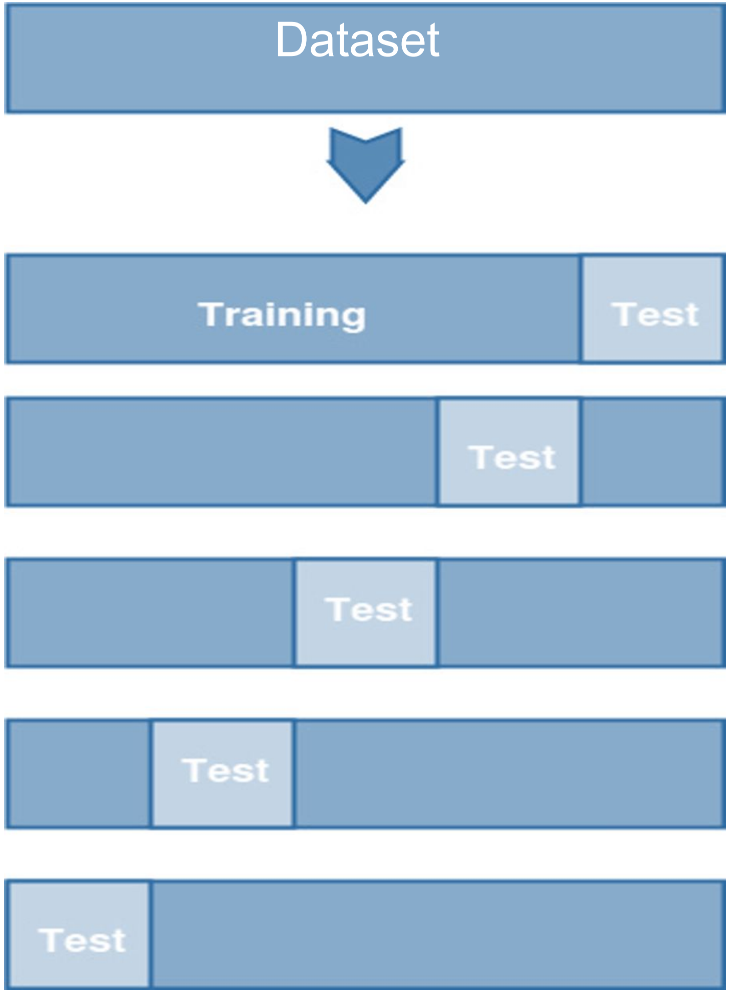

```
from sklearn import cross_validation

# value of K is 5.
data =
cross_validation.KFold(len(train_set)
, n_folds=5, indices=False)
```

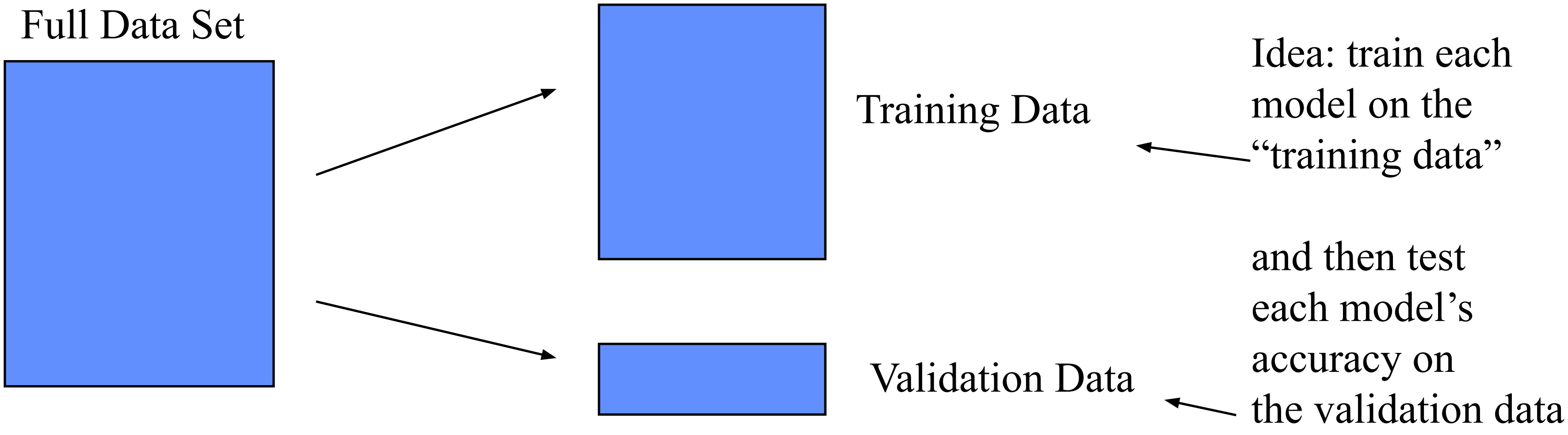
Cross Validation

```
from sklearn import cross_validation

# value of K is 5.
data =
cross_validation.KFold(len(train_set)
, n_folds=5, indices=False)
```



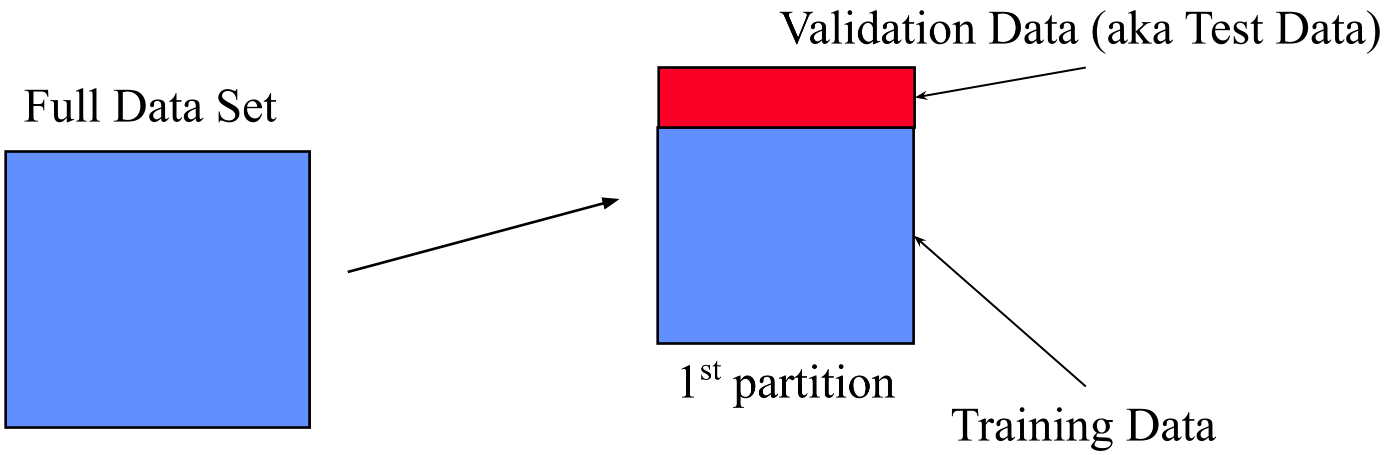
Cross Validation



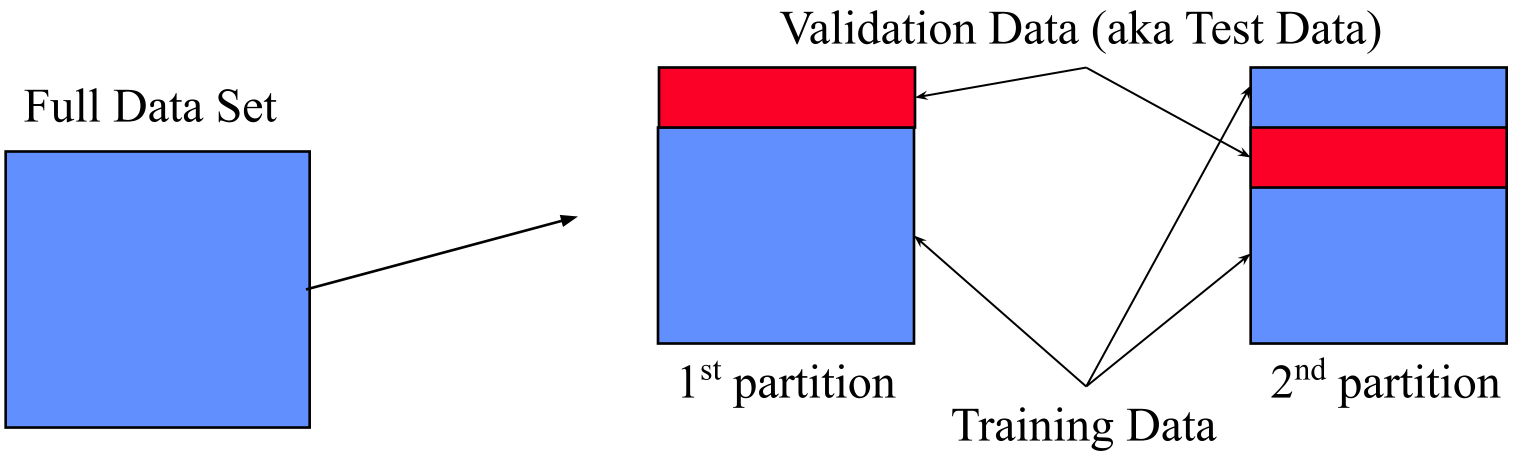
Training data performance is typically optimistic

- e.g., error rate on training data
- build a model on the training data
- assess performance on the test data

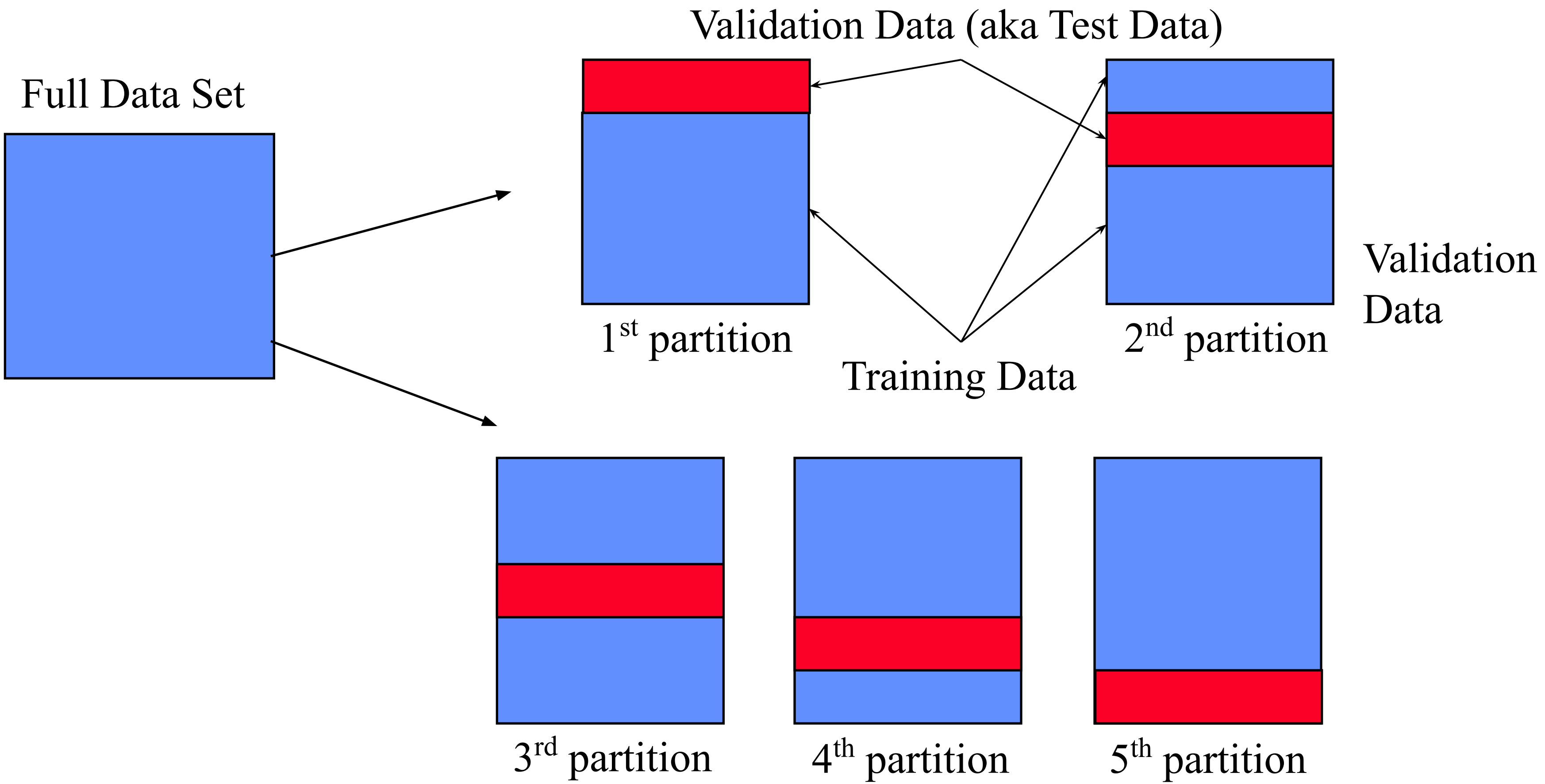
Disjoint Validation Data Sets for k = 5



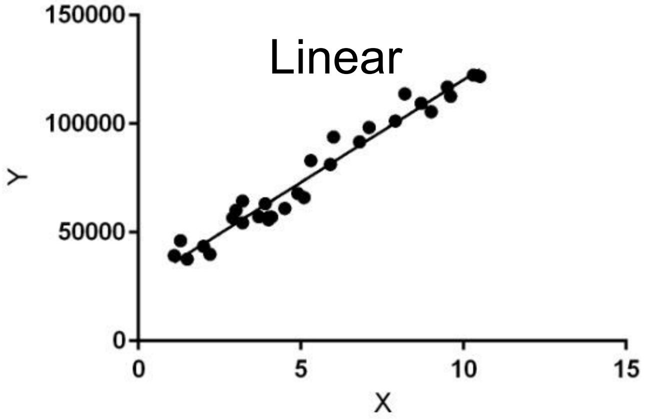
Disjoint Validation Data Sets for k = 5



Disjoint Validation Data Sets for k = 5



Identifying a Regression Problem

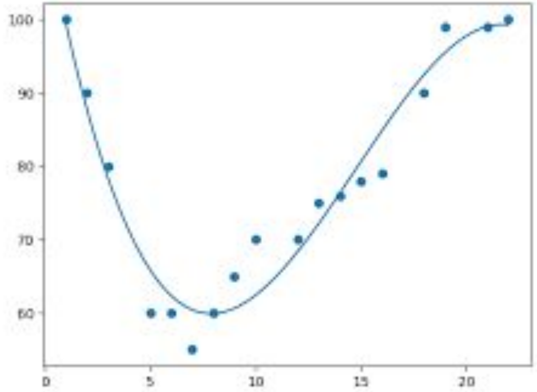


Do we want to **predict values**/targets?

Target data **continuous**?

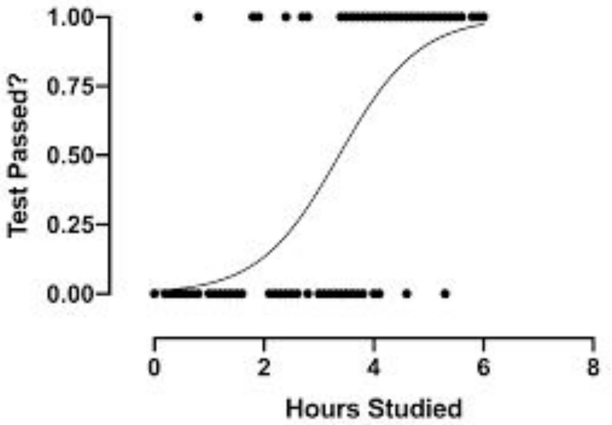
Does it **plot well** in a **scatter plot** i.e. ***y = mx + b*** where x can be any order

- Linear
- Polynomial
- Logistic
- Logarithmic
- Exponential



Polynomial

Logistic



Visualizing the Math

m x n * n x 1 matrix multiplication creates an m x 1 vector

$$w_0 + \quad X \quad w \quad = \quad \hat{y}$$

$$w_0 + \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ x_{2,1} & \cdots & x_{2,n} \\ \cdots & \cdots & \cdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_n \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \cdots \\ \hat{y}_m \end{bmatrix}$$

Where W is a column vector

Visualizing the Math

$$
X \qquad w \qquad = \hat{y}
$$

$$
\begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,n} \\ 1 & x_{2,1} & \cdots & x_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \cdots \\ w_n \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \cdots \\ \hat{y}_m \end{bmatrix}
$$

$$
\begin{bmatrix} 1 & x_{1,1} & x_{1,2} & x_{1,3} \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = w_0 + w_1 x_{1,1} + w_2 x_{1,2} + w_3 x_{1,3} = \hat{y}_i
$$

Where w is a column vector

Simple Linear Regression Function

$$\begin{array}{ccc} X & w & = & \hat{y} \\ \begin{bmatrix} 1 & x_{1,1} \\ 1 & x_{2,1} \\ \cdots & \cdots \\ 1 & x_{m,1} \end{bmatrix} & \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} & = & \begin{bmatrix} \hat{y_1} \\ \hat{y_2} \\ \cdots \\ \hat{y_m} \end{bmatrix} \end{array}$$

1st Order Simple Polynomial Regression

Where W is a column vector

2nd Order Polynomial Regression

$$\begin{array}{ccc} & n = 1 & n = 2 \\ \end{array}$$

$$\begin{bmatrix} 1 & x_{1,1} & (x_{1,1})^2 \\ 1 & x_{2,1} & (x_{2,1})^2 \\ \dots & \dots & \dots \\ 1 & x_{m,1} & (x_{m,1})^2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_m \end{bmatrix}$$

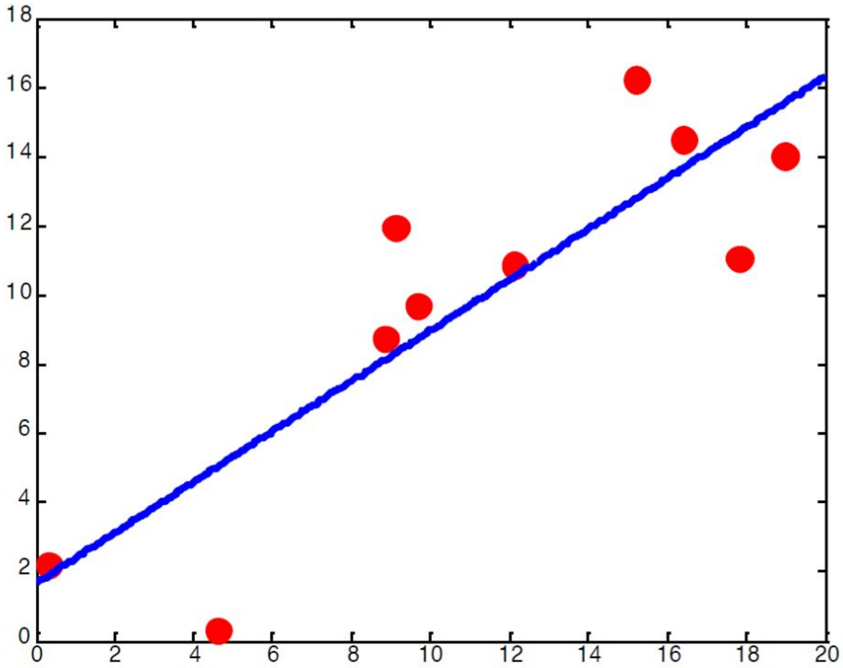
n^{th} Order Polynomial Regression

$$\begin{array}{ccc} X & w & = & \hat{y} \\ \begin{bmatrix} 1 & x_{1,1}^1 & x_{1,2}^2 & \cdots & x_{1,n}^n \\ 1 & x_{2,1}^1 & x_{2,2}^2 & \cdots & x_{2,n}^n \\ \cdots & \cdots & \cdots & \cdots\cdot & \cdots \\ 1 & x_{m,1}^1 & x_{m,2}^2 & \cdots & x_{m,n}^n \end{bmatrix} & \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \cdots \\ w_n \end{bmatrix} & = & \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \cdots \\ \hat{y}_m \end{bmatrix} \end{array}$$

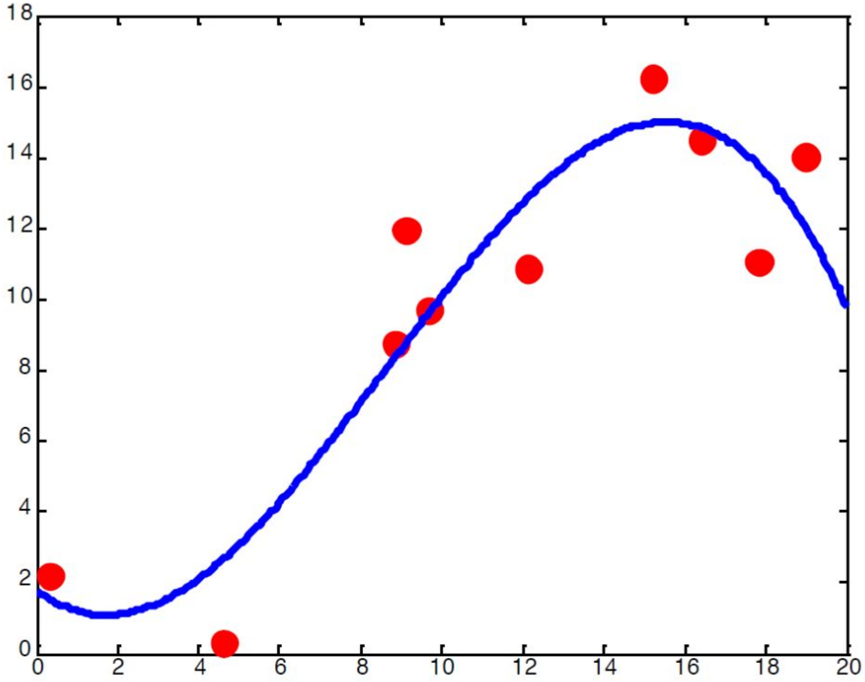
Where W is a column vector

Polynomial Fit

1st Order

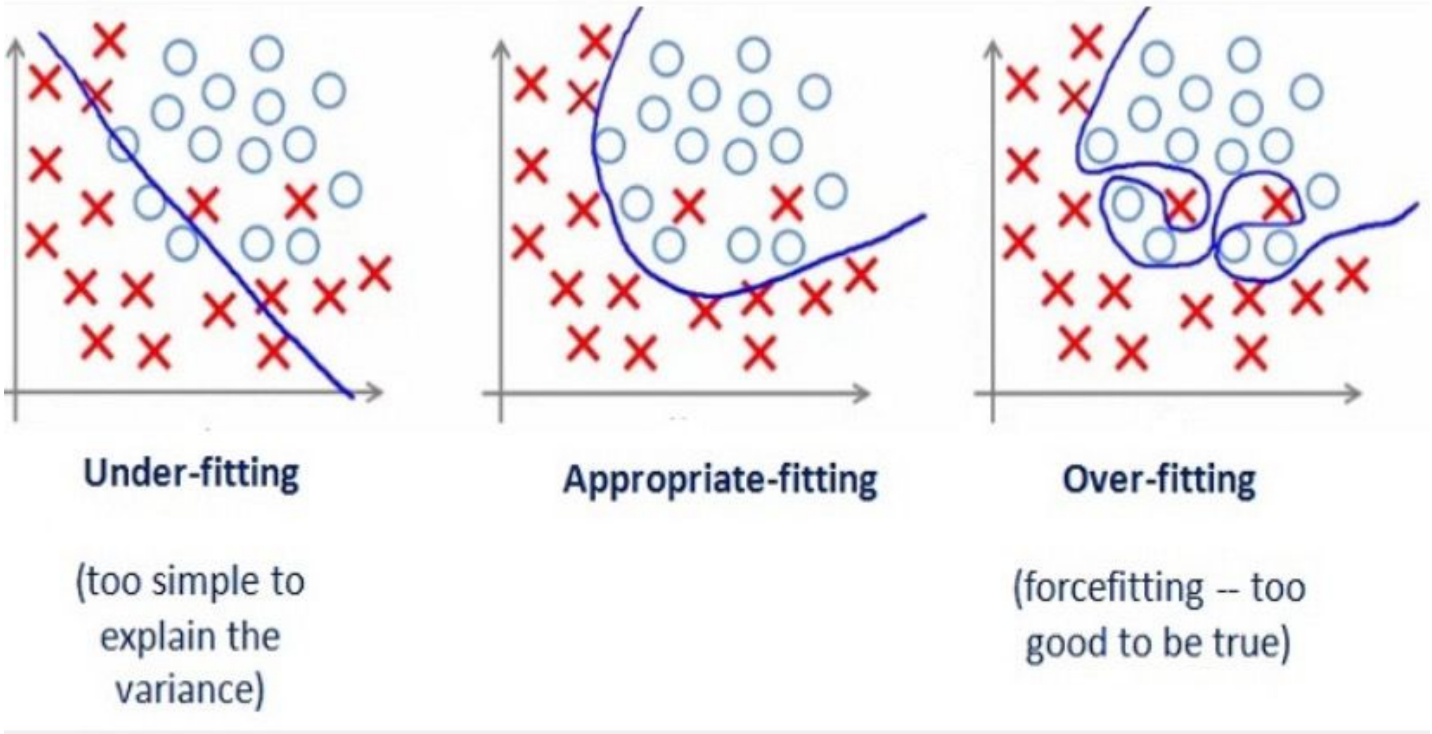


3rd Order

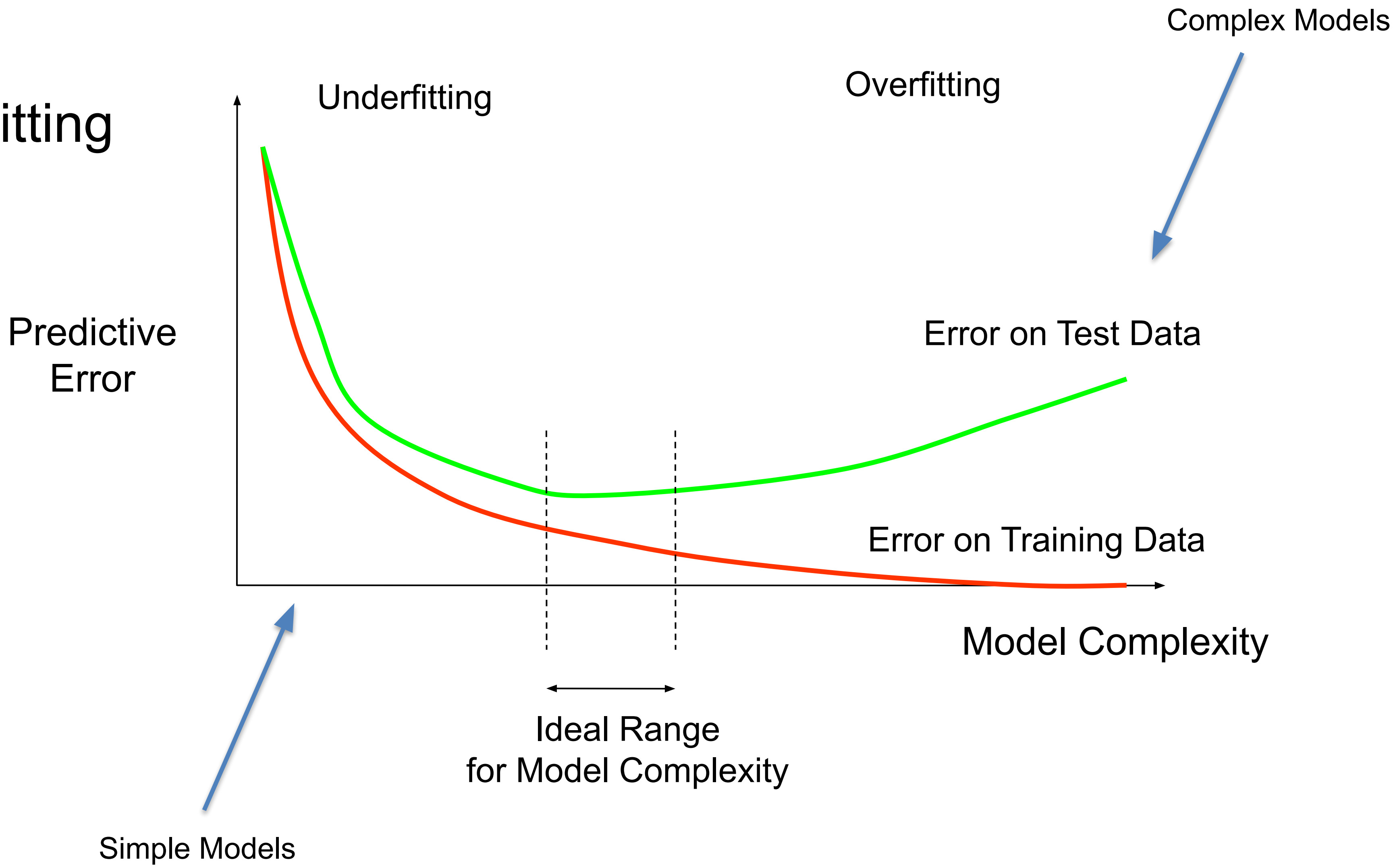


(credit) Dr. Alexander Ihler

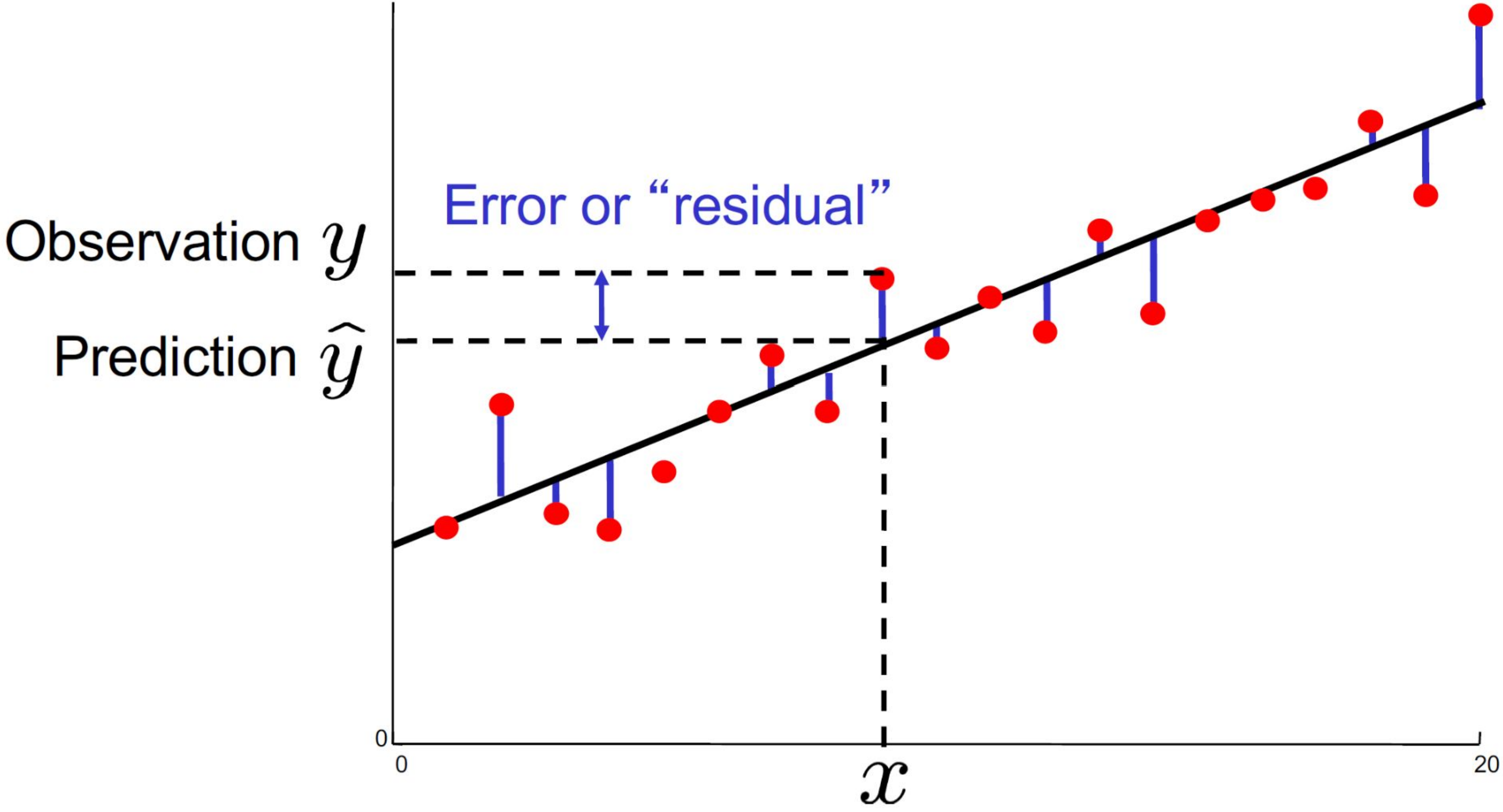
Fitting


Under-fitting
(too simple to explain the variance)
Appropriate-fitting
Over-fitting
(forcefitting -- too good to be true)

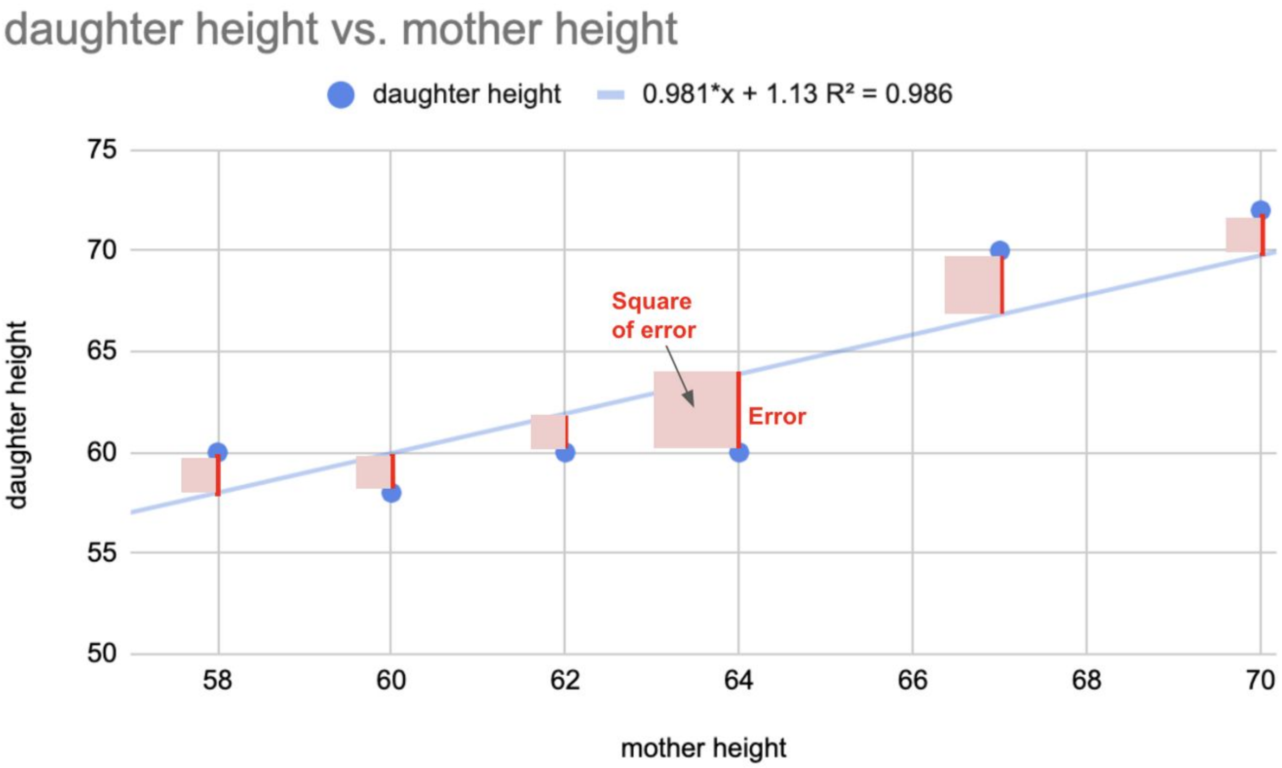
Fitting


Complex Models
Underfitting
Overfitting
Predictive Error
Error on Test Data
Error on Training Data
Model Complexity
Ideal Range
for Model Complexity
Simple Models

Calculating error



Square of the Error Visual



Ordinary Least Squares (OLS)

$$RSS_t := \min(RSS_{t-1}),\ where\ w\ is\ changed\ at\ each\ time\ step\ t$$

$$RSS = \sum_{i=1}^{m} (y_i - w \cdot x_i)^2$$

Observations - Predictions

We want to find some change in ***w*** where (Observations - Predictions)^2 = 0
Rate of change = 0

$$find\ \frac{\delta}{\delta w} RSS = 0$$

Where W is a column vector

Expands to $w_1x_1 + w_0$

Finding the Derivative in our Loss Function

$$RSS = \sum_{i=1}^{m} (y_i - w \cdot x_i)^2$$

Observations - Predictions

Take the derivative

$$\frac{\delta}{\delta w} \sum_{i=1}^{m} (y_i - x_i \cdot w)^2 = 0$$

$$2 \sum_{i=1}^{m} x_i (y_i - x_i \cdot \hat{w}) - (y_i - x_i \cdot \hat{w}) = 0$$

OLS Method

$$\hat{w} = \frac{\sum_{i=1}^{m} (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{m} (x_i - \bar{x})^2}$$

$$\sigma_{xy} = \sum_{i=1}^{m} (y_i - \bar{y})(x_i - \bar{x})$$

$$\sigma_x = \sum_{i=1}^{m} (x_i - \bar{x})$$

$$\hat{w} = \frac{\sigma_{xy}}{\sigma_x^2}$$

Covariance(x,y)/Variance(x)

approximation

Gradient Descent

$$\frac{\delta}{\delta w} \; MSE$$

$$\frac{\delta}{\delta w} \boxed{\frac{1}{2m}} \sum_{i=1}^{m} (y_i - \hat{y})^2 = 0$$

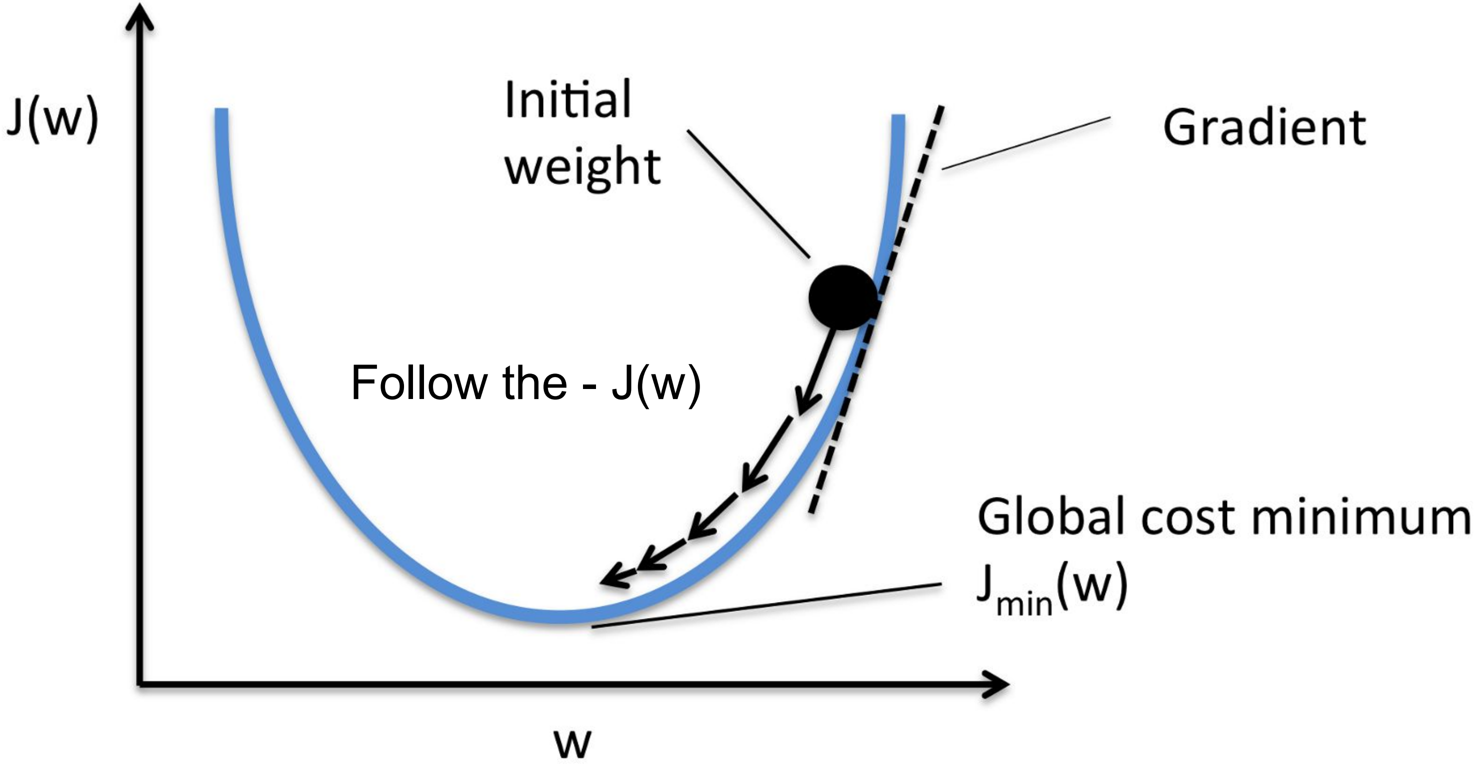
Expands to $w_1 x_1 + w_0$

$$2 \sum_{i=1}^{m} x_i(y_i - x_i \hat{w}) - (y_i - x_i \hat{w}) = 0$$

Gradient Descent

J(w) = + when cost is going up
J(w) = - when cost is going down
So we want to min J(w)

Courtesy of Nvidia Website



Expands to $w_1x_1 + w_0$

Gradient Descent Methods

$$J(w_j)_t := J(w_j)_{t-1} - \alpha(\frac{1}{m}\sum_{i=1}^{m}(y_i - x_i w) + \frac{1}{m}\sum_{i=1}^{m}x_i(y_i - x_i w))$$

Batch GD → Performing GD on all observations

SGD → Calculating GD & performing step on a randomly selected observation

mini-Batch GD → SGD but with several data points (a subset of observations)

Neural Networks can use either, but computation is expensive so SGD is often used

Update theta from previous theta - alpha*derivatives that include w_1 and w_0

Standard Logistic Growth Function

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

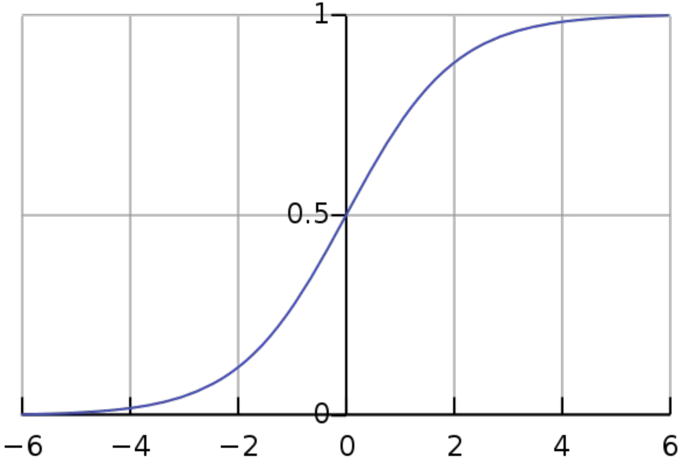
$f(x)$ = output of the function

L = the curve's maximum value

k = logistic growth rate or steepness of the curve

x_0 = the x value of the sigmoid midpoint

x = real number

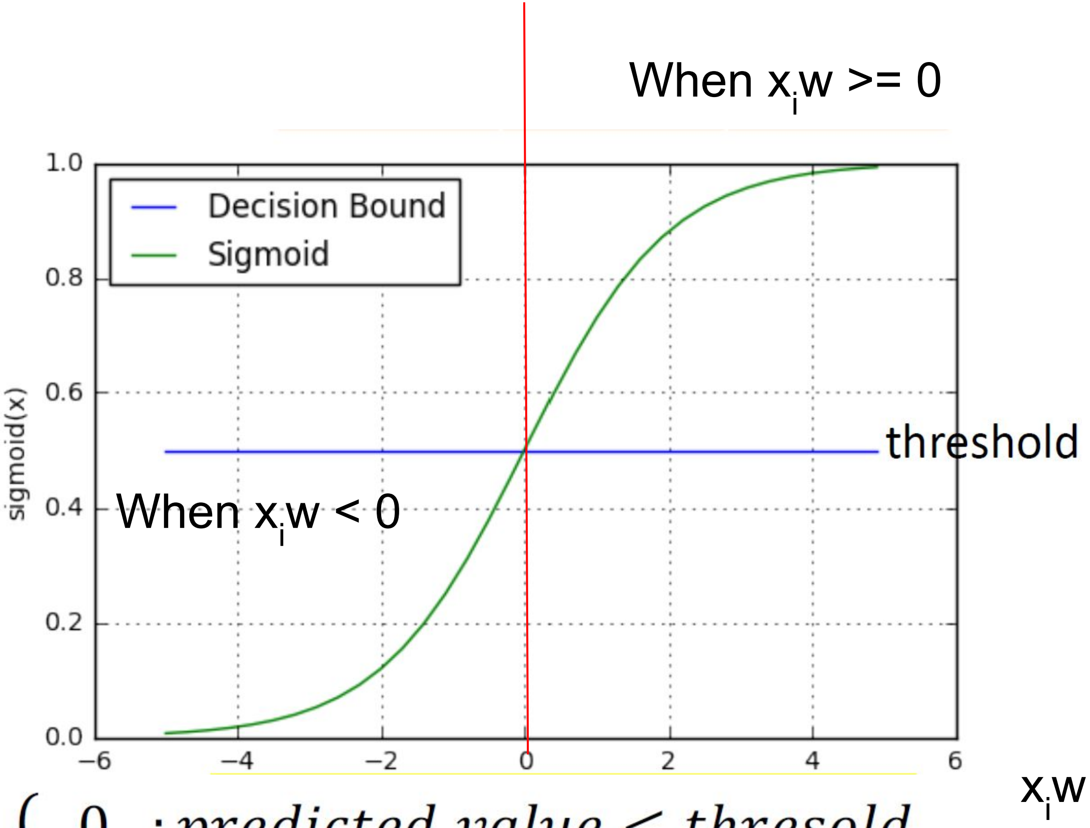


$$L = 1, k = 1, x_0 = 0,$$

Logistic Regression

$\hat{y}_i$ Is our prediction given formula on the right, but is given a value of 0 or 1 based on the threshold of 0.5

Prior to threshold we can say our raw value is our probability. As our raw value is continuous from 0 to 1.



$$\hat{y}_i = \begin{cases} 0 & ; predicted\ value < thresold \\ 1 & ; predicted\ value \geq threshold \end{cases}$$

Logistic Regression

$$\hat{y} = \frac{1}{1 + e^{(\mu - x)/s}}$$

$\hat{y}_i$ Is our prediction given formula on the right, but is given a value of 0 or 1 based on the threshold of 0.5

Prior to threshold we can say our raw value is our probability. As our raw value is continuous from 0 to 1.

$$\hat{y} = \frac{1}{1 + e^{-(w_0 + w_1 x)}}$$

where $w_0 = -\mu/s$ and $w_1 = 1/s$ $\therefore$ we can solve for μ and s

$$\mu = w_0/w_1 \text{ and } s = 1/w_1$$

Logistic Regression

Where we have Bernoulli observations
And p_k is the probability of y_k=1 and
$1-p_k$ is the probability $y_k = 0$

The log loss for the *k*-th point is:

We can say p_k is our raw value, which is our probability of of y_k = 1 given $x_i w$

$$\text{Cost} \begin{cases} -\ln p_k & \text{if } y_k = 1, \\ -\ln(1-p_k) & \text{if } y_k = 0. \end{cases}$$

$$-y_k \ln p_k - (1 - y_k)\ln(1 - p_k)$$

log-likelihood

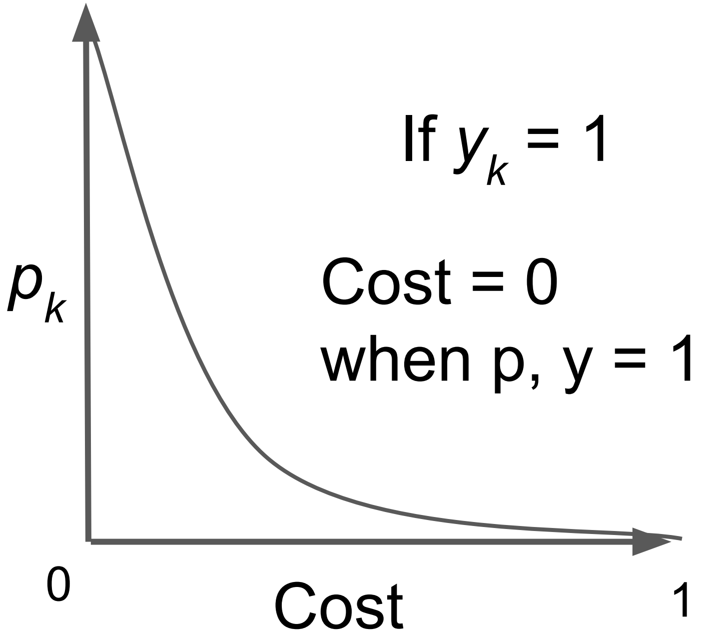
$$\ell = \sum_{k:y_k=1} \ln(p_k) + \sum_{k:y_k=0} \ln(1-p_k) = \boxed{\sum_{k=1}^{K} (y_k \ln(p_k) + (1-y_k)\ln(1-p_k))}$$

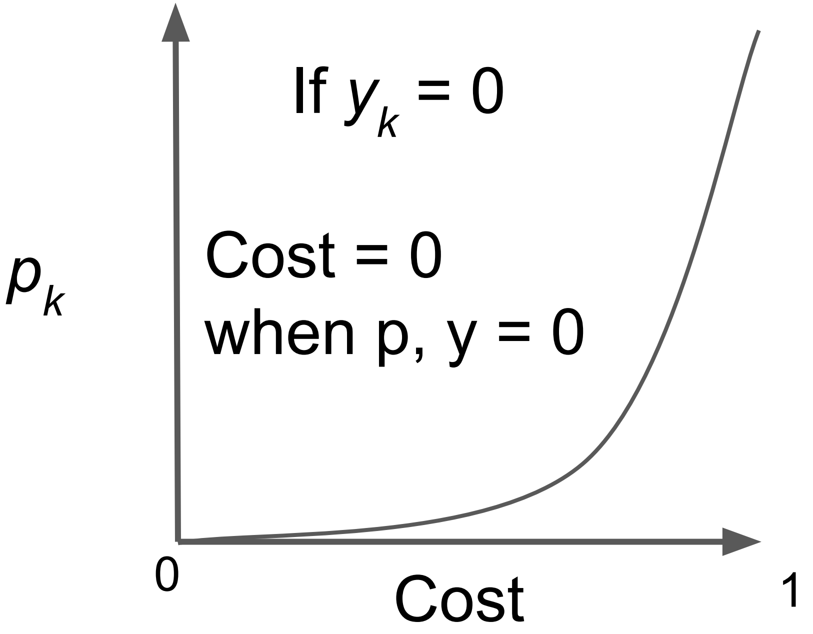
$J(w)$ = d/dw ℓ* 1/k

Logistic Regression

The log loss for the k-th point is:

$$\mathrm{Cost}(p_k, y_k) \begin{cases} -\ln p_k & \text{if } y_k = 1, \\ -\ln(1 - p_k) & \text{if } y_k = 0. \end{cases}$$





Logistic Regression

log-likelihood

$$\ell = \sum_{k:y_k=1} \ln(p_k) + \sum_{k:y_k=0} \ln(1-p_k) = \sum_{k=1}^{K} \left(y_k \ln(p_k) + (1-y_k) \ln(1-p_k) \right)$$

$J(w)$ = 1/k d/dw ℓ

d/dw ℓ = $\sum_{k=1}^{K} (y_k - p_k) + (y_k - p_k) x_k$

Logistic Regression: Parameter Estimation

$$0 = \frac{\partial \ell}{\partial \beta_0} = \sum_{k=1}^{K} (y_k - p_k)$$

$$0 = \frac{\partial \ell}{\partial \beta_1} = \sum_{k=1}^{K} (y_k - p_k) x_k$$

Lasso & Ridge Regression

$$Cost(w) = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{y})^2$$

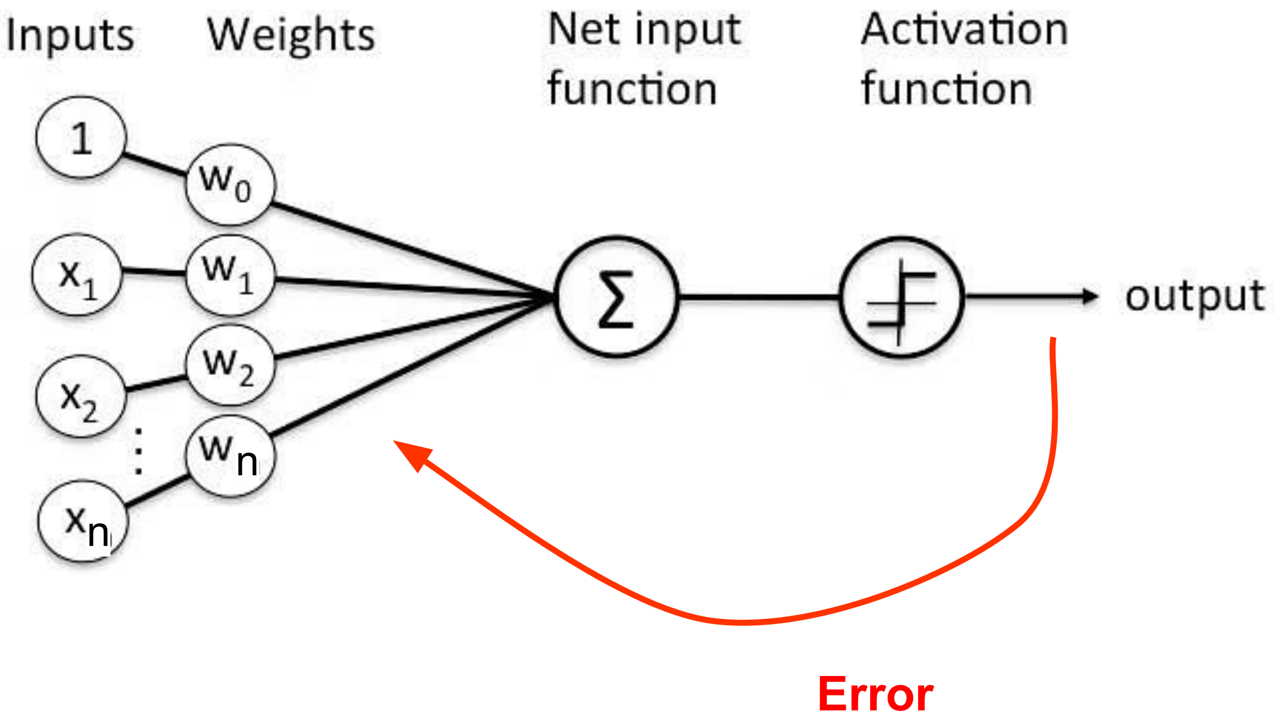
Lasso

$$Cost(w) = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{y})^2 + \lambda\sum_{j=1}^{D}|w_j|$$

Ridge

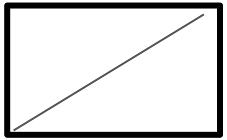
$$Cost(w) = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{y})^2 + \lambda\sum_{j=1}^{D}w_j^2$$

Perceptron

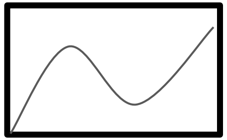

Inputs
Weights
Net input function
Activation function
1
w_0
x_1
w_1
x_2
w_2
x_n
w_n
Σ
output
Error

Activation Functions

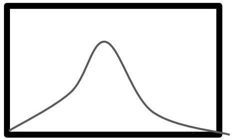
Linear



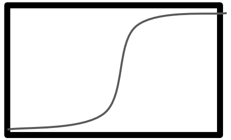
Polynomial



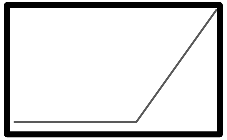
Gaussian



Sigmoid/Logistic

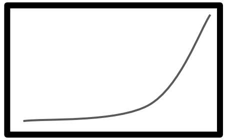


ReLU (Rectified Linear Unit)



max(0,x)

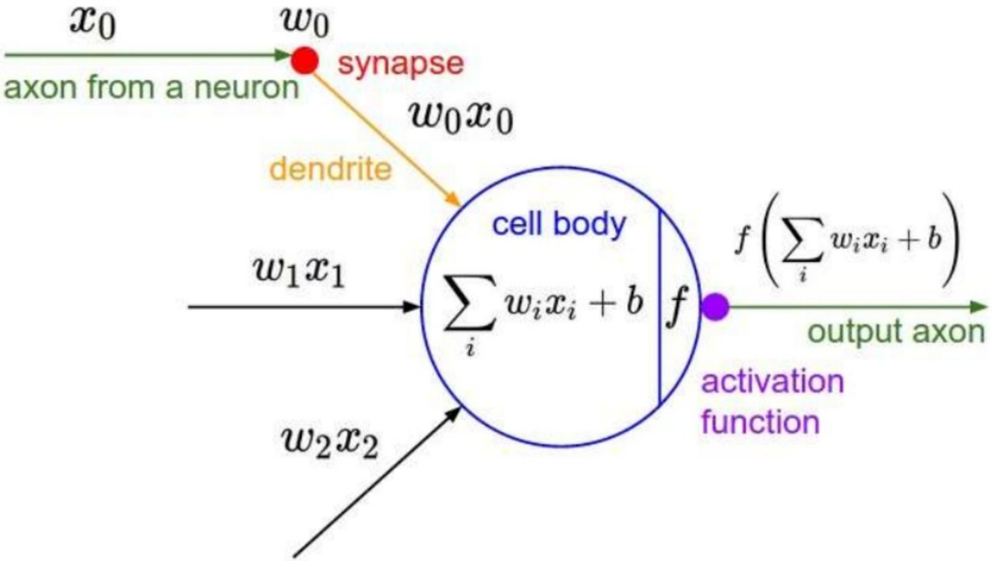
SoftMax



https://cs231n.github.io/neural-networks-1/
https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html#elu

Artificial Neuron History

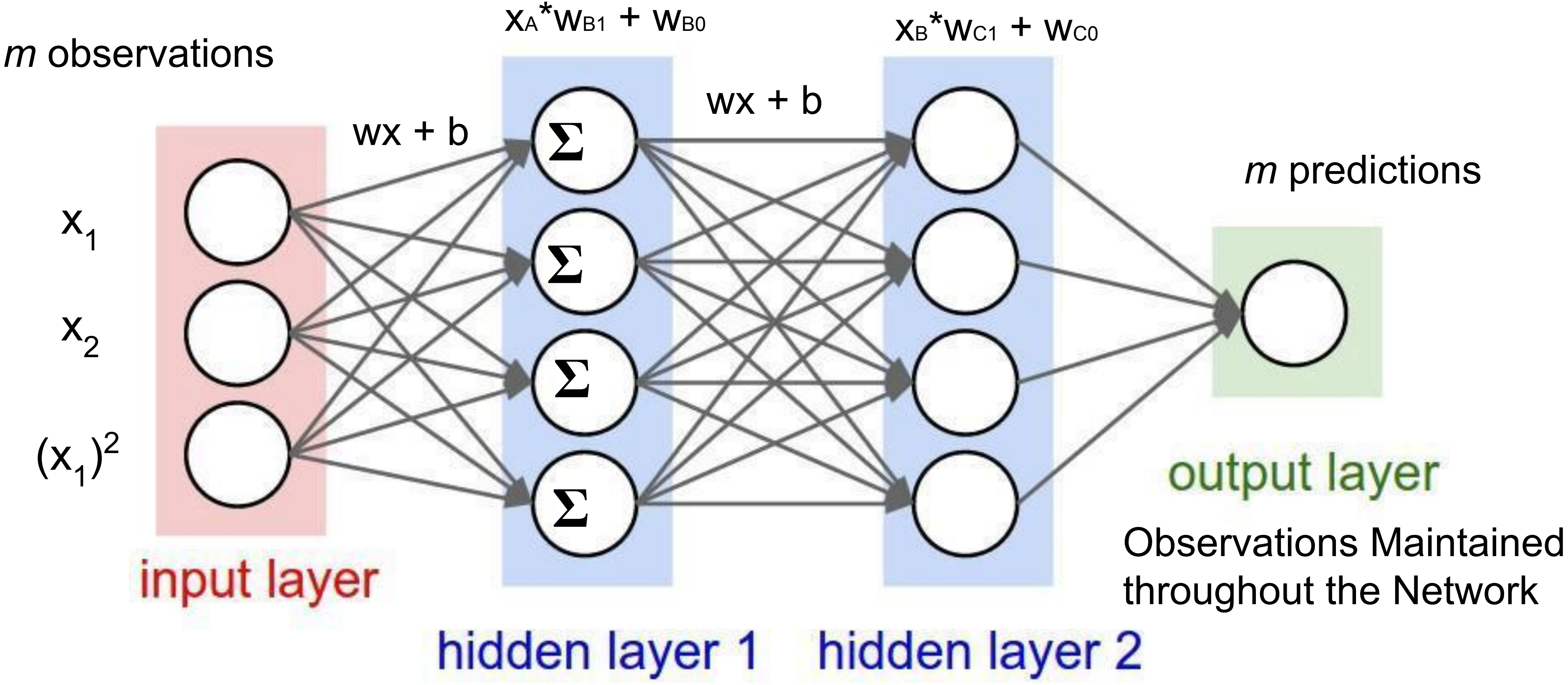
2nd Generation Neuron



https://www.slideshare.net/hitechpro/introduction-to-spiking-neural-networksfrom-a-computational-neuroscience-perspective/30

Courtesy of Nvidia Website

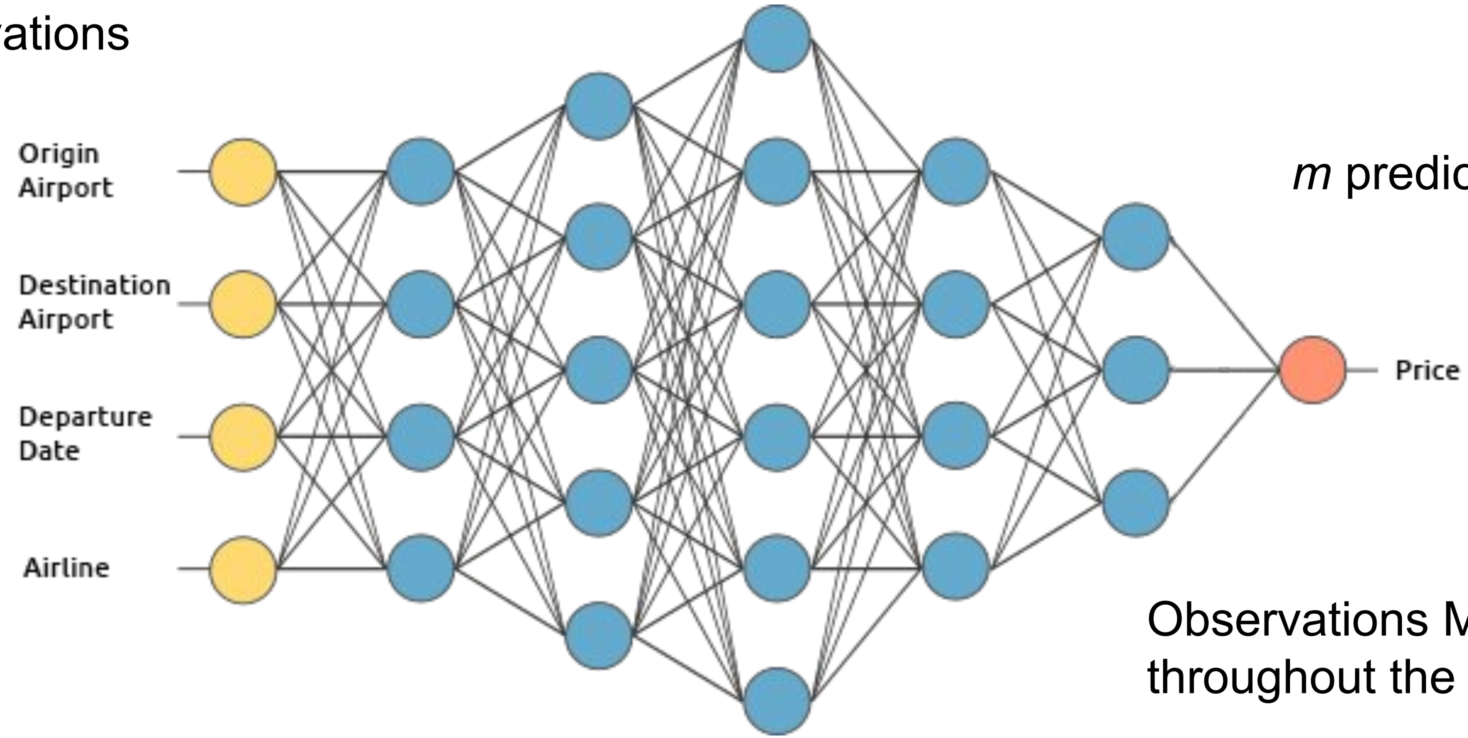
Simple Neural Net: 2 Hidden Layers



Courtesy of Nvidia Website

Deep Neural Net: Several Hidden Layers

m observations



m predictions

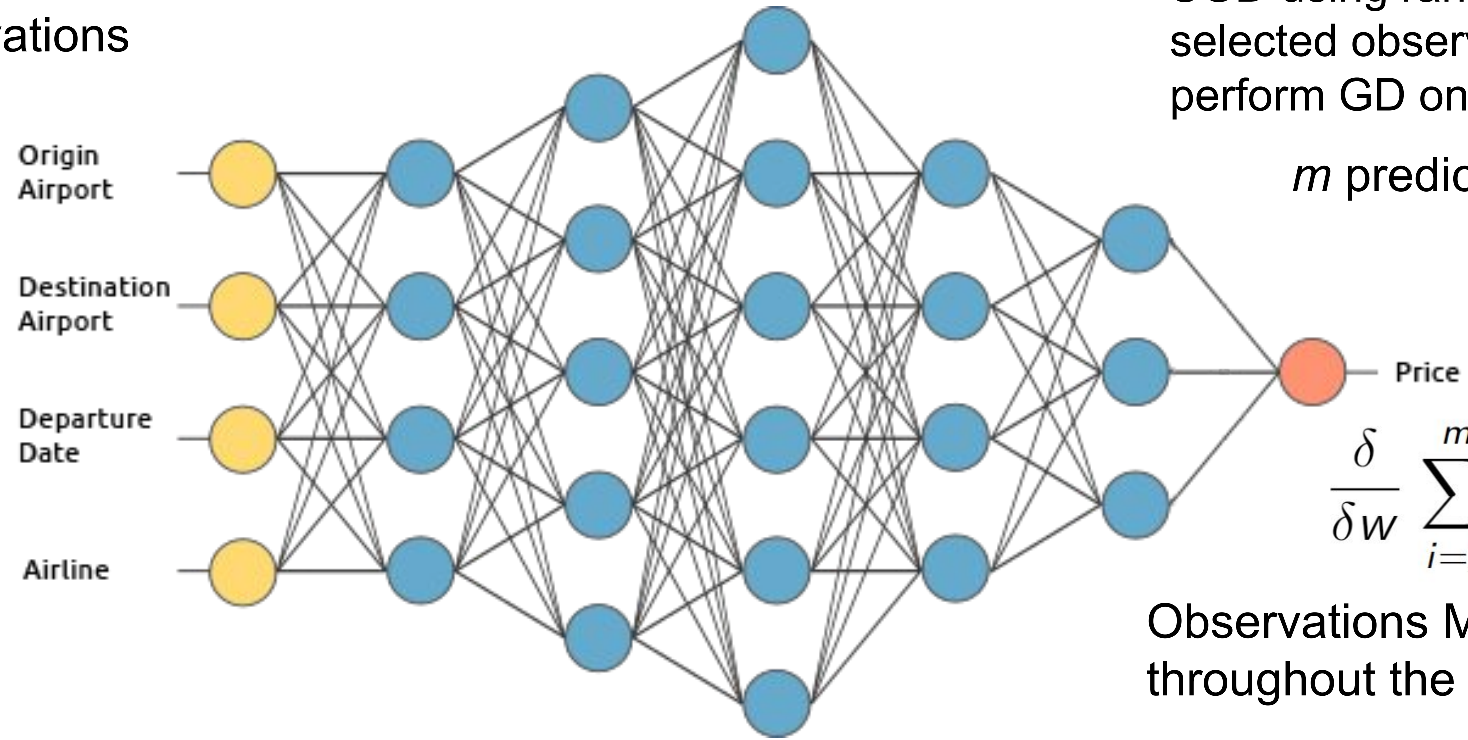
Observations Maintained throughout the Network

Deep Neural Net: Several Hidden Layers

m observations

SGD using randomly selected observation and perform GD on it

m predictions



SGD

$$\frac{\delta}{\delta w}\sum_{i=1}^{m}(y_i - \hat{y})^2$$

Observations Maintained throughout the Network

Courtesy of Nvidia Website

Can Have Hidden Layers but Must Have Convolution Layers

CNN’s

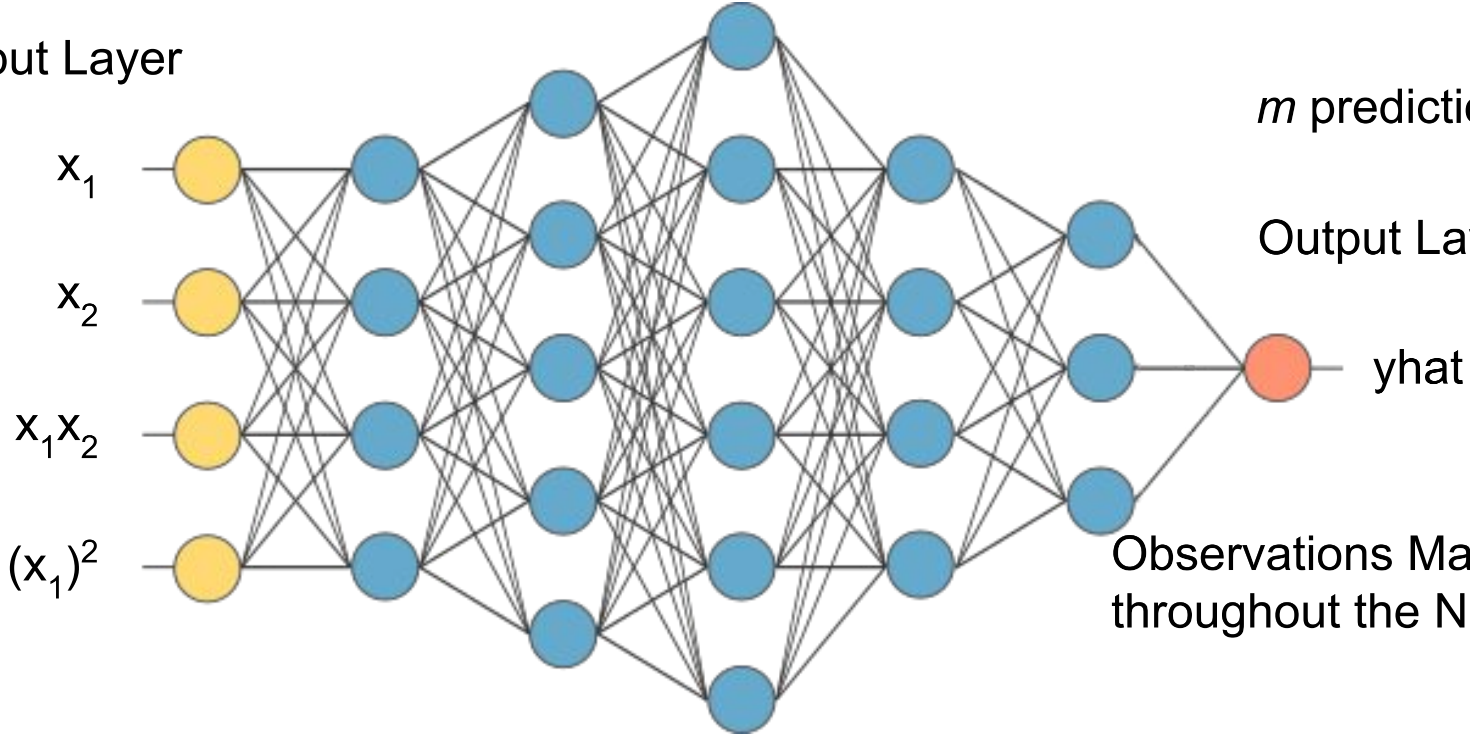
m observations

Input Layer

Convolutional Layer

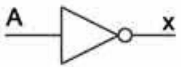
m predictions

Output Layer

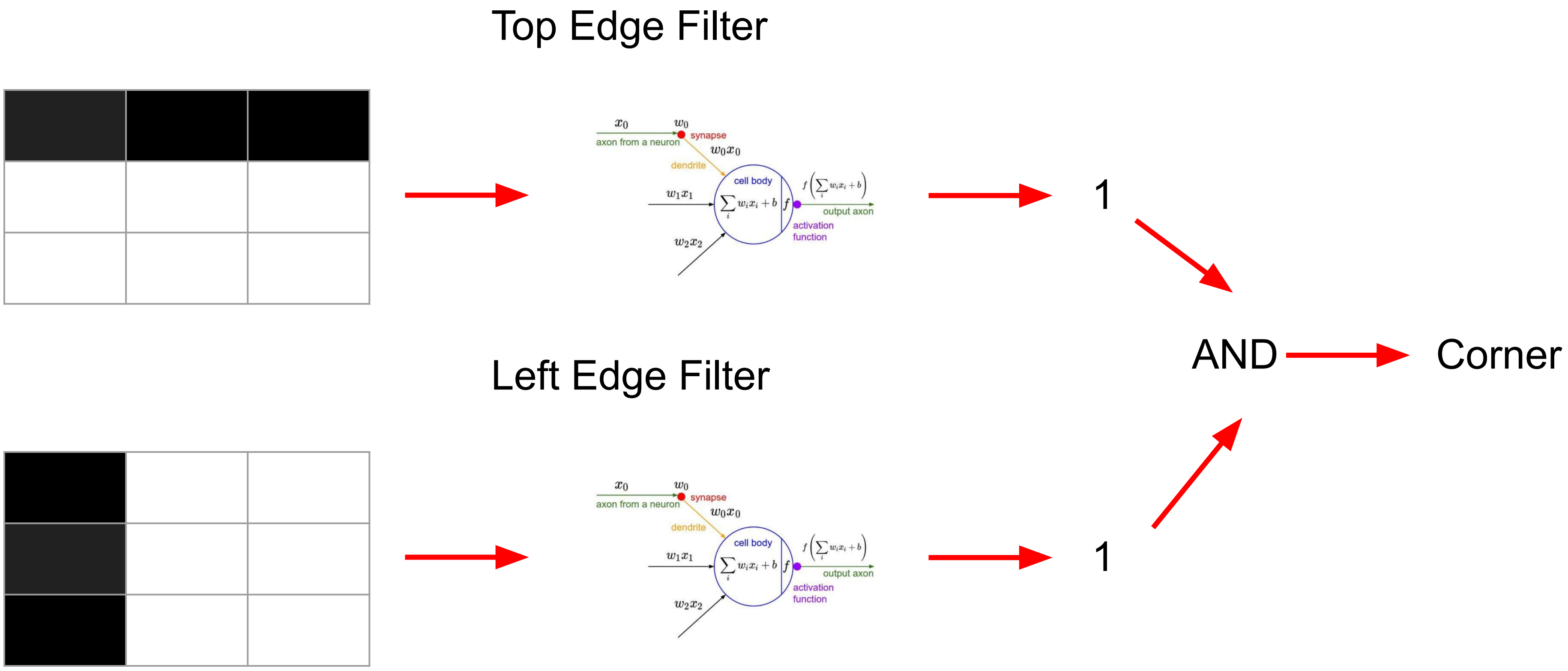


Observations Maintained throughout the Network

Logic Gates Instead of Aggregation Functions

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
Symbol							
Truth Table	A X 0 **1** 1 **0**	B A X 0 0 **0** 0 1 **0** 1 0 **0** 1 1 **1**	B A X 0 0 **1** 0 1 **1** 1 0 **1** 1 1 **0**	B A X 0 0 **0** 0 1 **1** 1 0 **1** 1 1 **1**	B A X 0 0 **1** 0 1 **0** 1 0 **0** 1 1 **0**	B A X 0 0 **0** 0 1 **1** 1 0 **1** 1 1 **0**	B A X 0 0 **1** 0 1 **0** 1 0 **0** 1 1 **1**

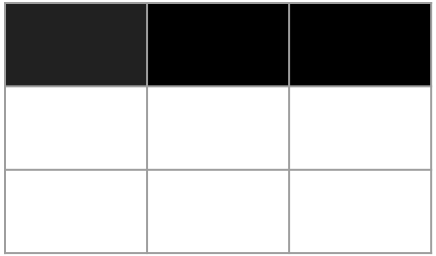
Filters



Top Edge Filter
x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$w_1 x_1$
$\sum_i w_i x_i + b$
f
$f\left(\sum_i w_i x_i + b\right)$
output axon
activation function
$w_2 x_2$
1
AND
Corner
Left Edge Filter
x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$w_1 x_1$
$\sum_i w_i x_i + b$
f
$f\left(\sum_i w_i x_i + b\right)$
output axon
activation function
$w_2 x_2$
1

Filters: Matrix Convolution

Top Edge Filter



1	1	1
0	0	0
-1	-1	-1

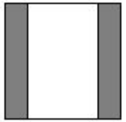
Left Edge Filter

1	0	-1
1	0	-1
1	0	-1

Filters: Matrix Convolution

0 value when pixel is uniform
Large value when pixel is not uniform
Max value when pixel fits pattern

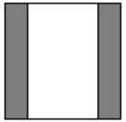
$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \rightarrow \begin{bmatrix} 1*10 & 0*10 & -1*10 \\ 1*10 & 0*10 & -1*10 \\ 1*10 & 0*10 & -1*10 \end{bmatrix} \rightarrow \begin{bmatrix} \boxed{0} & 30 & 30 & 0 \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$$



Filters: Matrix Convolution

0 value when pixel is uniform
Large value when pixel is not uniform
Max value when pixel fits pattern

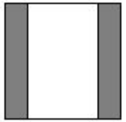
$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} \quad \boxed{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}} \rightarrow \begin{bmatrix} 1*10 & 0*10 & -1*0 \\ 1*10 & 0*10 & -1*0 \\ 1*10 & 0*10 & -1*0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & \boxed{30} & 30 & 0 \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$$



Filters: Matrix Convolution

0 value when pixel is uniform
Large value when pixel is not uniform
Max value when pixel fits pattern

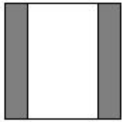
$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \rightarrow \begin{bmatrix} 1*10 & 0*0 & -1*0 \\ 1*10 & 0*0 & -1*0 \\ 1*10 & 0*0 & -1*0 \end{bmatrix} \quad \begin{bmatrix} 0 & 30 & \boxed{30} & 0 \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$$



Filters: Matrix Convolution

0 value when pixel is uniform
Large value when pixel is not uniform
Max value when pixel fits pattern

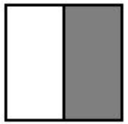
$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \rightarrow \begin{bmatrix} 1*0 & 0*0 & -1*0 \\ 1*0 & 0*0 & -1*0 \\ 1*0 & 0*0 & -1*0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 30 & 30 & 0 \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$$

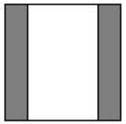


Filters: Matrix Convolution

0 value when pixel is uniform
Large value when pixel is not uniform
Max value when pixel fits pattern

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

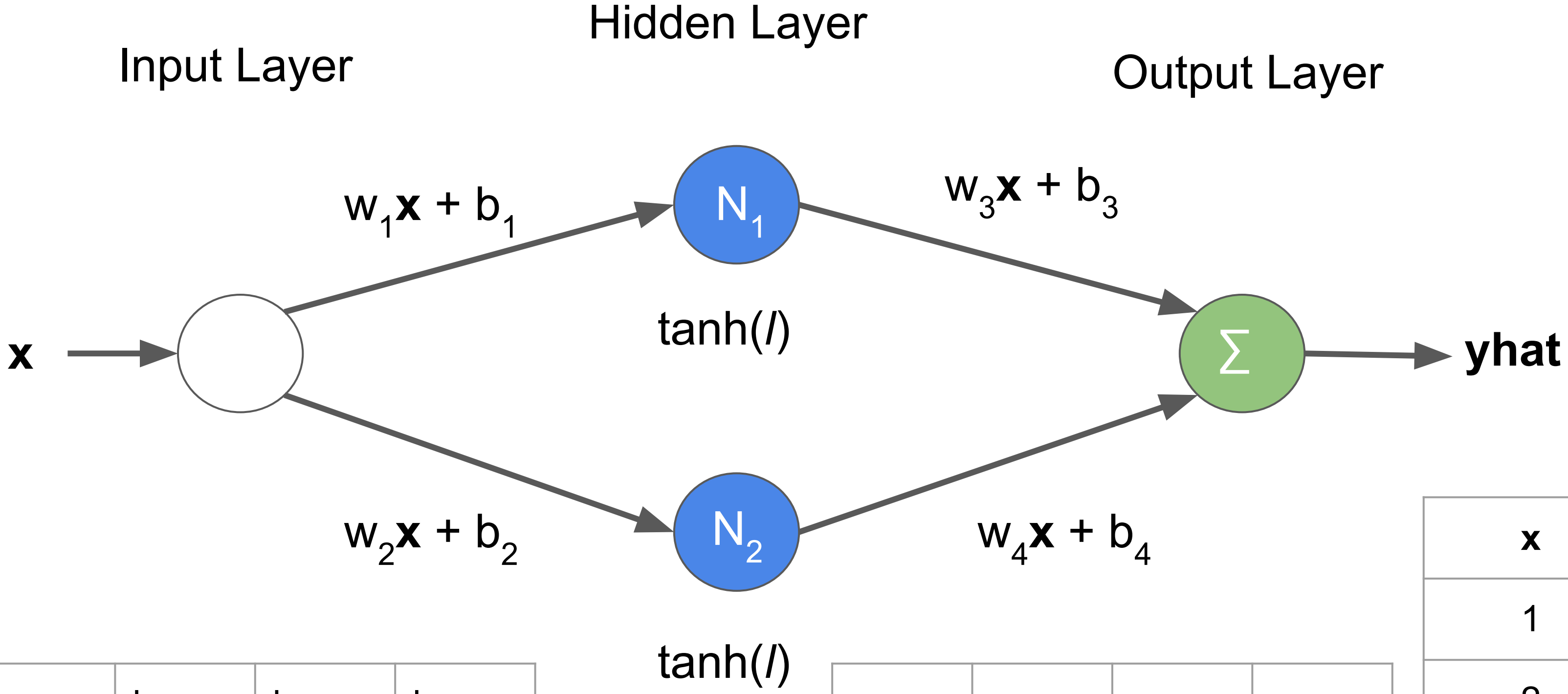




https://www.youtube.com/watch?v=IN2XmBhILt4

Activation function = tanh(x)

Neural Net Backpropagation



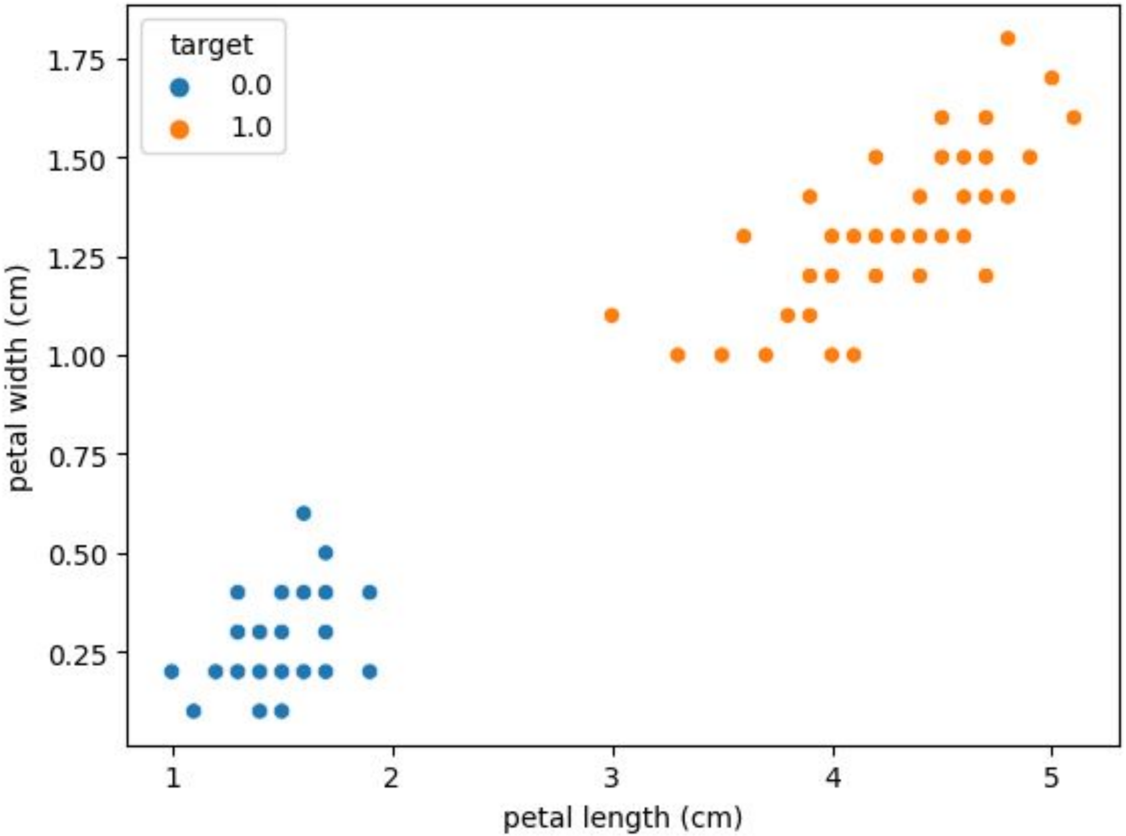
yhat
?
?
?

x	y
1	8
2	15
3	28

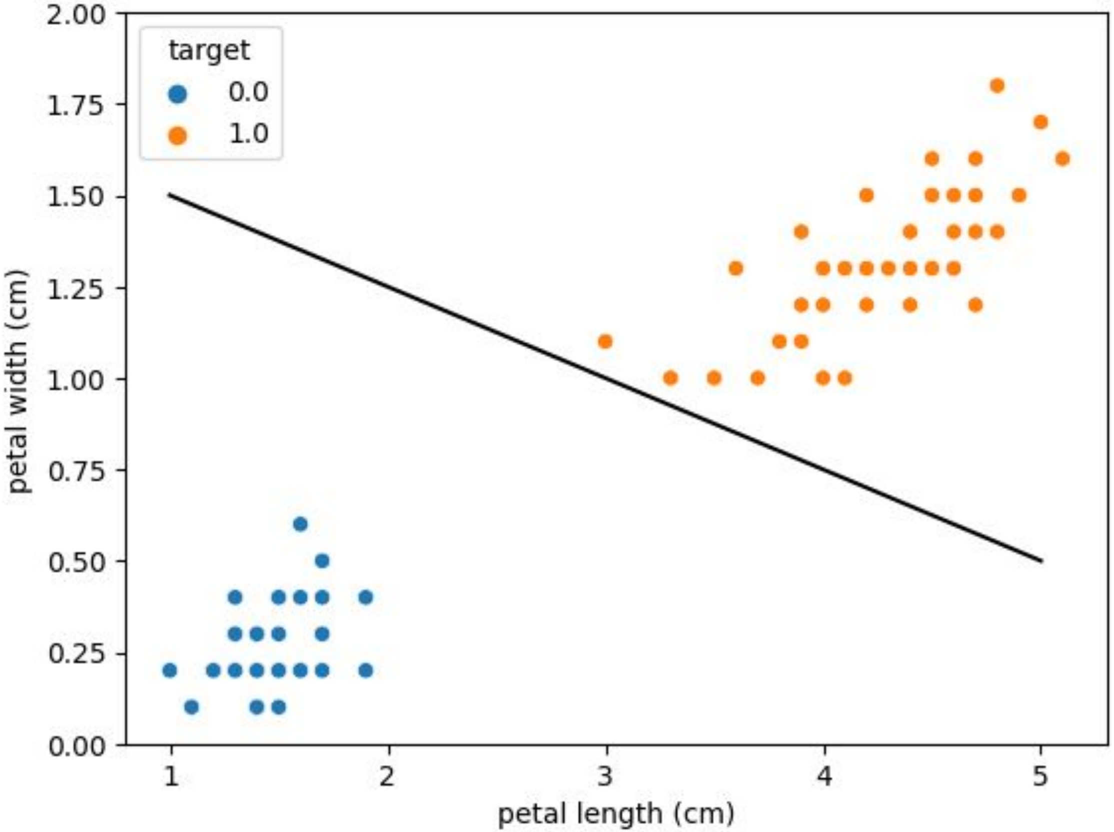
b_1	b_2	b_3	b_4
0	0	0	0

w_1	w_2	w_3	w_4
1	1	1	1

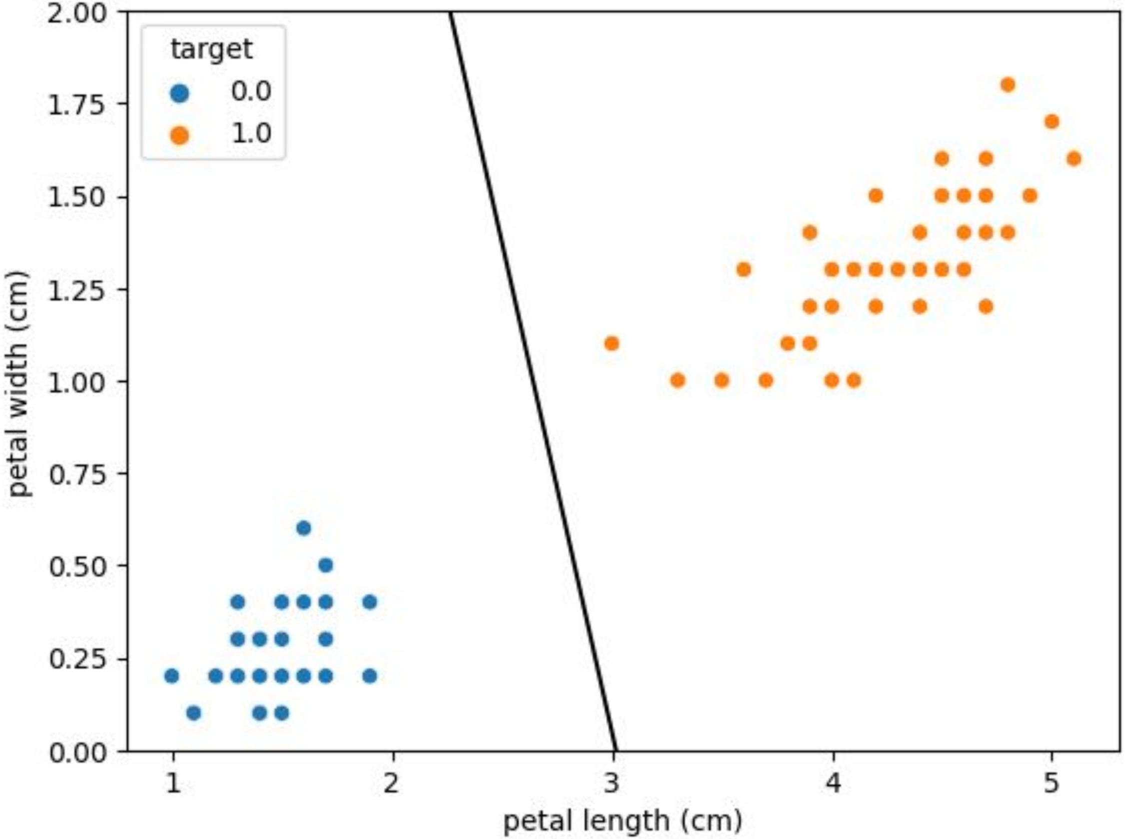
What is Classification



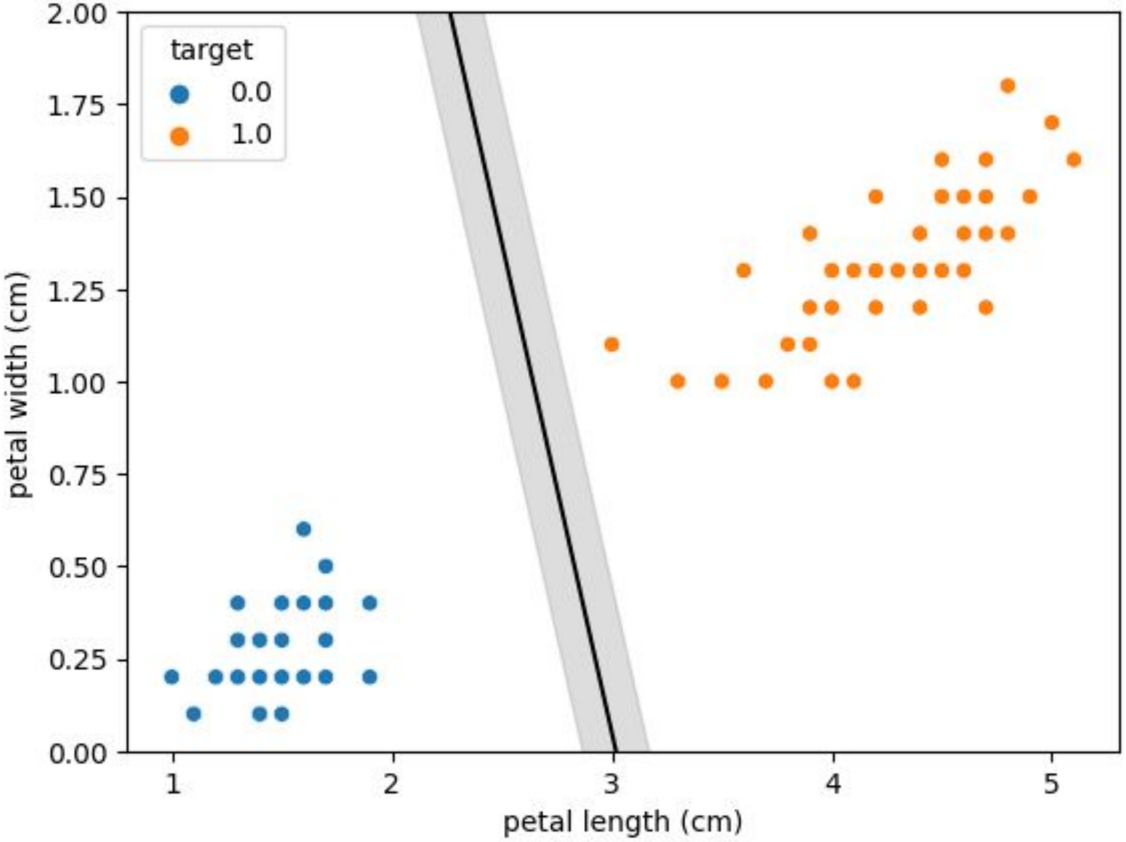
Using a line to define a boundary



Which is a good fit?

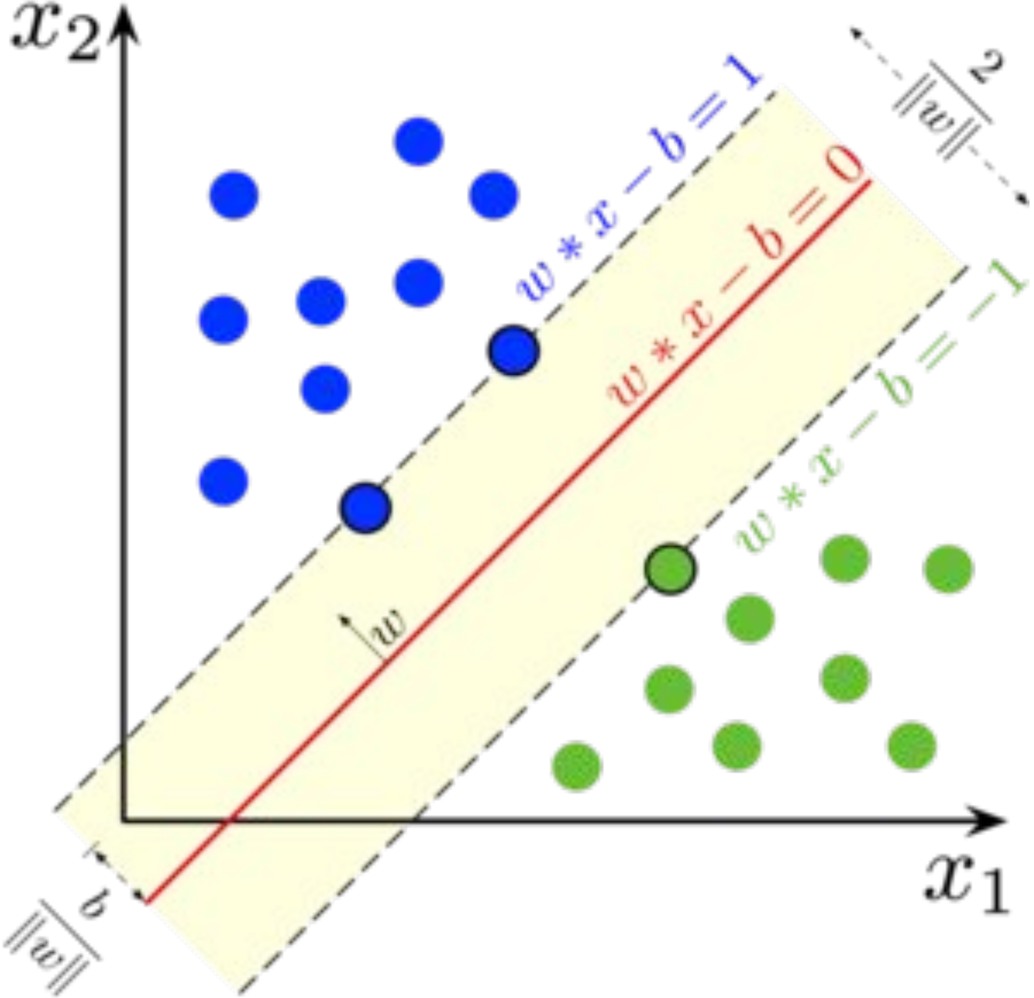


Adding a Margin



What is Classification

Wikipedia

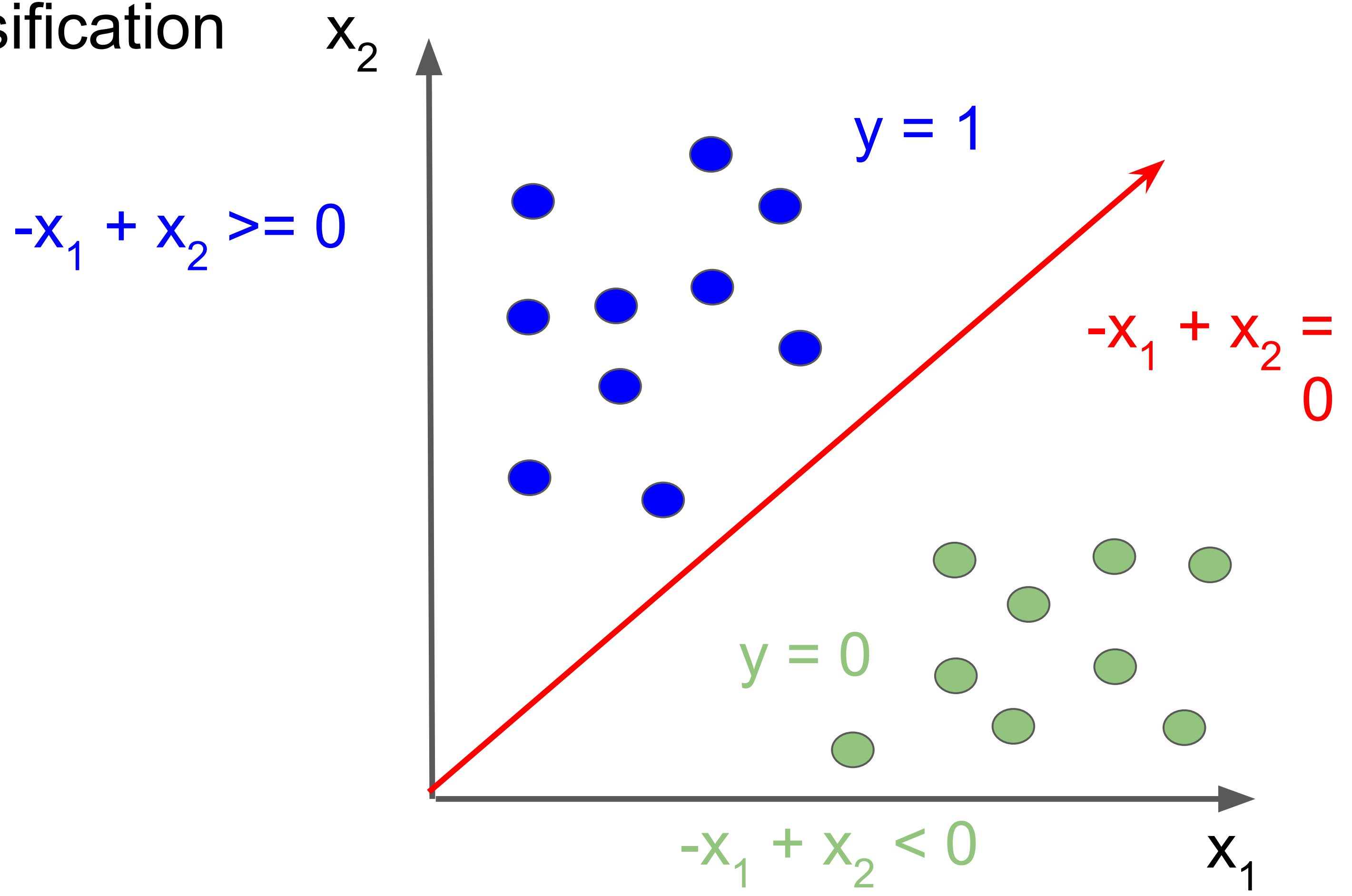


Predict $y = 1$ given $x_1w_1 + x_2w_2 + w_0$

Logistic Classification

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

$$\mathbf{w} = \begin{bmatrix} -1 \\ +1 \end{bmatrix}$$

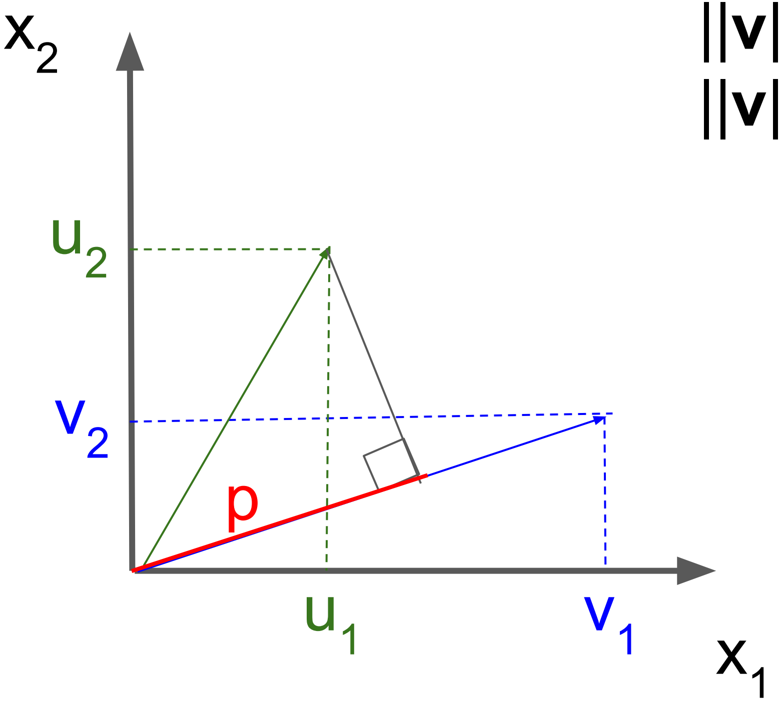


Adding Margin (Recall Inner Products)

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$||\mathbf{v}||$ = Length of $\mathbf{v}$

$$||\mathbf{v}|| = \sqrt{v_1^{\,2} + v_2^{\,2}}$$



$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$\mathbf{v}^T\mathbf{u} = ||\mathbf{v}||\ p$

Where p is the projection of u onto v

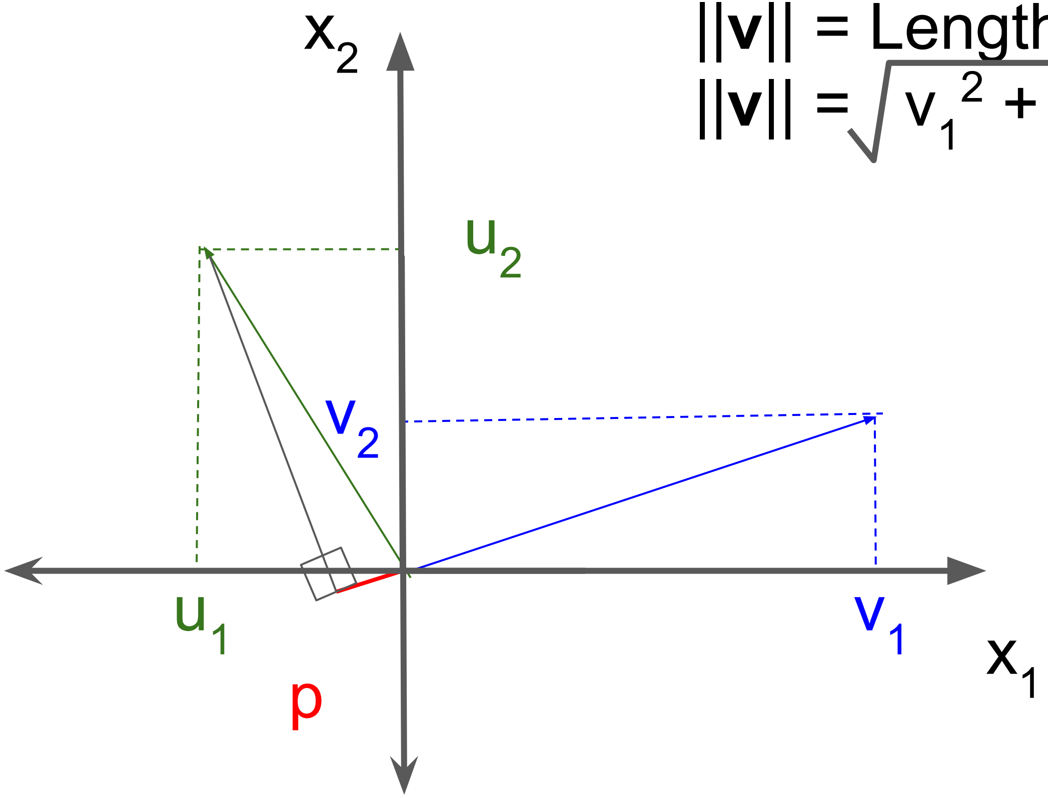
Adding Margin (Recall Inner Products)

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$\|\mathbf{v}\|$ = Length of $\mathbf{v}$

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2}$$

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$



$$\mathbf{v}^T\mathbf{u} = \|\mathbf{v}\|\, p$$

Where p is the projection of u onto v

Predict $y = 1$ given $x_1w_1 + x_2w_2 + w_0$

SVM with Margin

We want large **p**

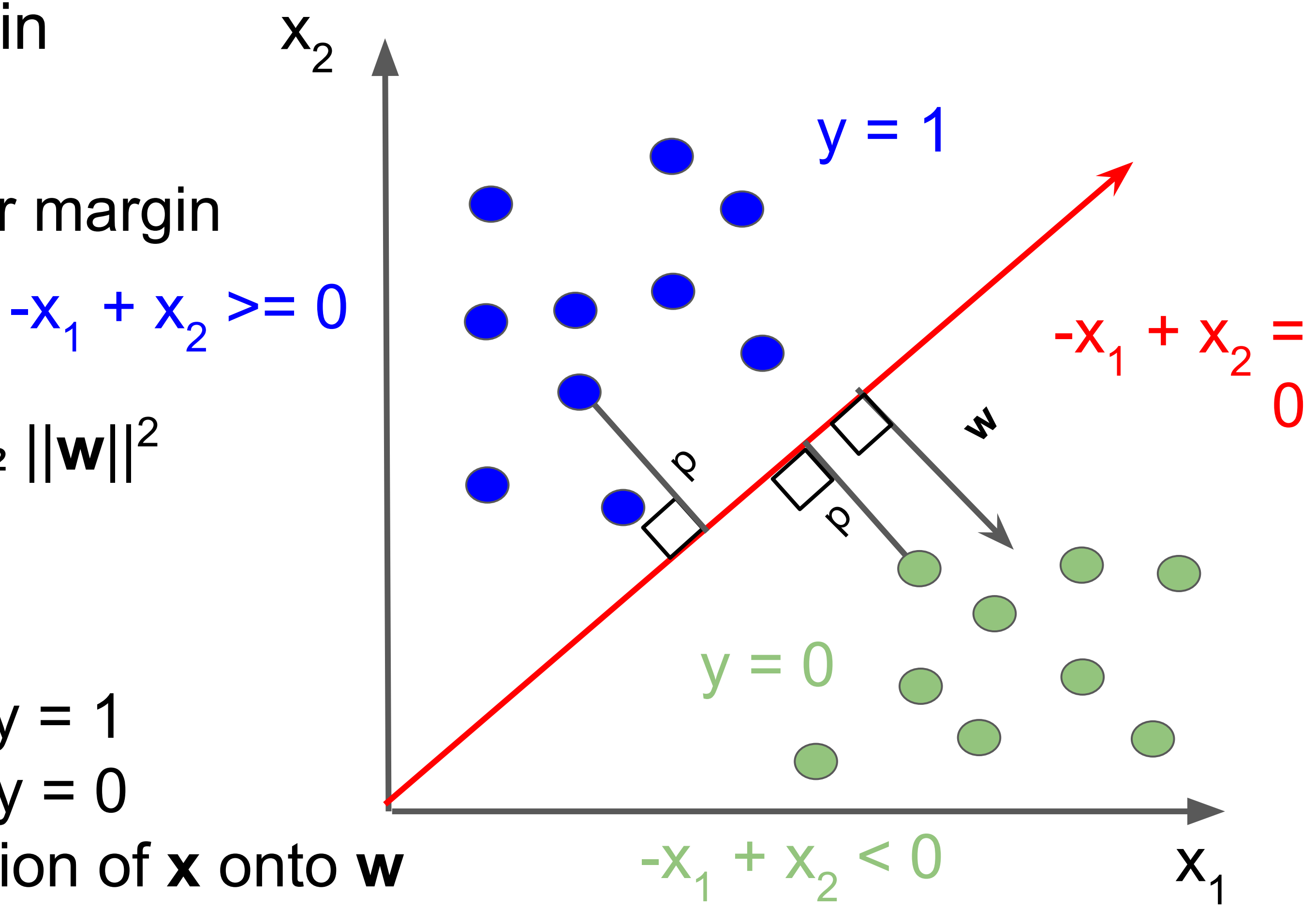
Since it wil be our margin

$$\min_w \frac{1}{2}\sum_{j=1}^{m} w_j^2 = \frac{1}{2}\,||\mathbf{w}||^2$$

$\mathbf{p}\ ||\mathbf{w}|| => 1$ if $y = 1$

$\mathbf{p}\ ||\mathbf{w}|| <= -1$ if $y = 0$

Where **p** is projection of **x** onto **w**



SVM Classification

Wikipedia