Computer Vision: Fall 2022 — Lecture 7

Dr. Karthik Mohan

Univ. of Washington, Seattle

October 21, 2022

Check-In

→ IRIS Dataset

1. Assignment 3 assigned

Check-In

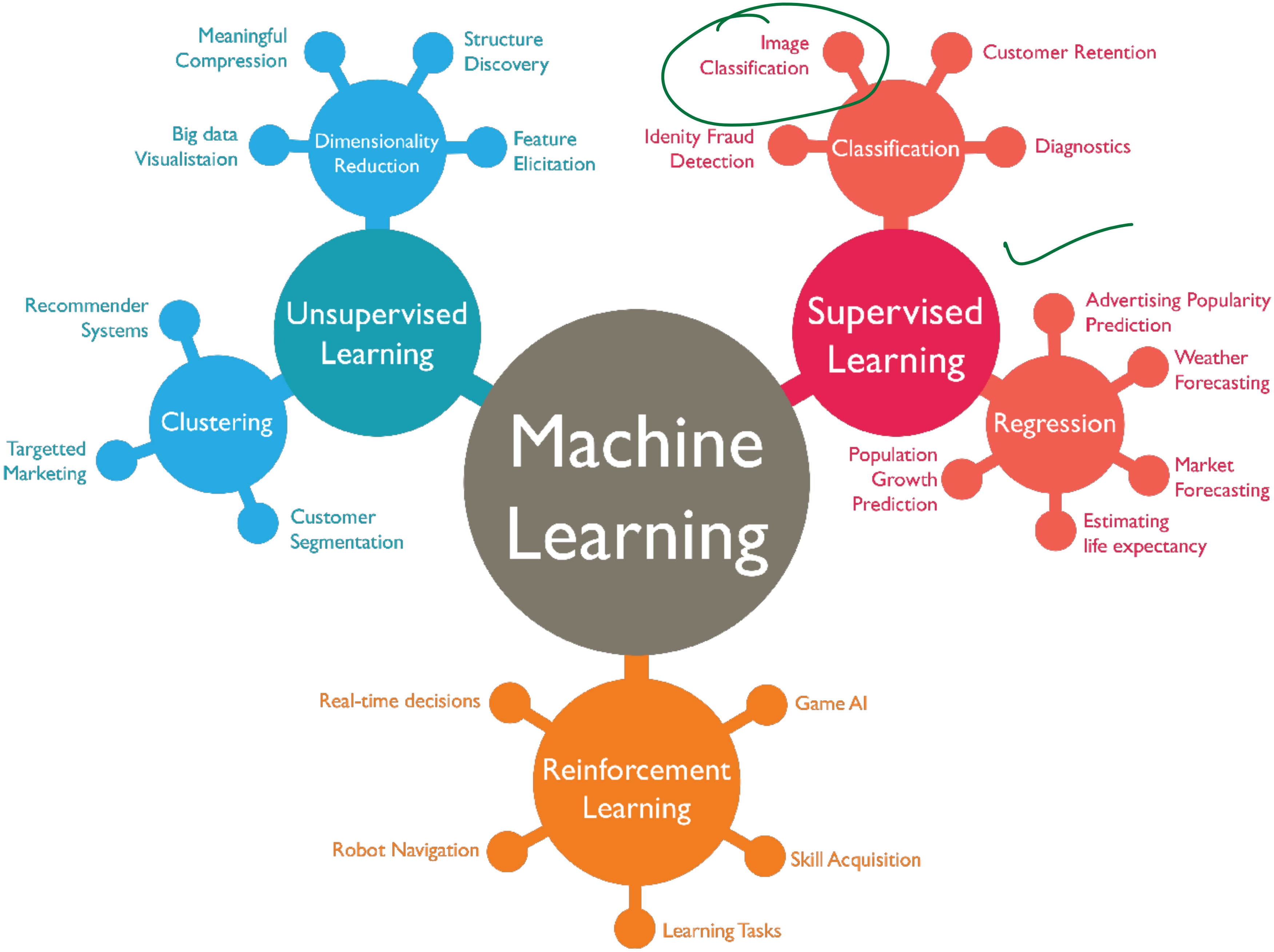
1. Assignment 3 assigned
2. Other thoughts/questions?

Today

1. Binary Classification Recap
2. Classification Metrics
3. Train/Validation/Test Data Sets
4. Over-fitting and Regularization

References

1. Good Book for Machine Learning Concepts


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

Computer Vision Topics

1. Image Processing using convolutions
2. Image De-noising
3. Image Smoothing
4. Image Clustering
5. Image Classification
6. Object Detection
7. Semantic Segmentation
8. Instance Segmentation (maybe)
9. Image Embeddings
10. Image to Text
11. Image Captioning
12. Text to Image (high-level)

Flower Classification

Ex:- $\underset{\in \mathbb{R}^4}{h(x)}$ = [Petal Length, Sepal Length, Petal/Sepal, Petal×Sepal]

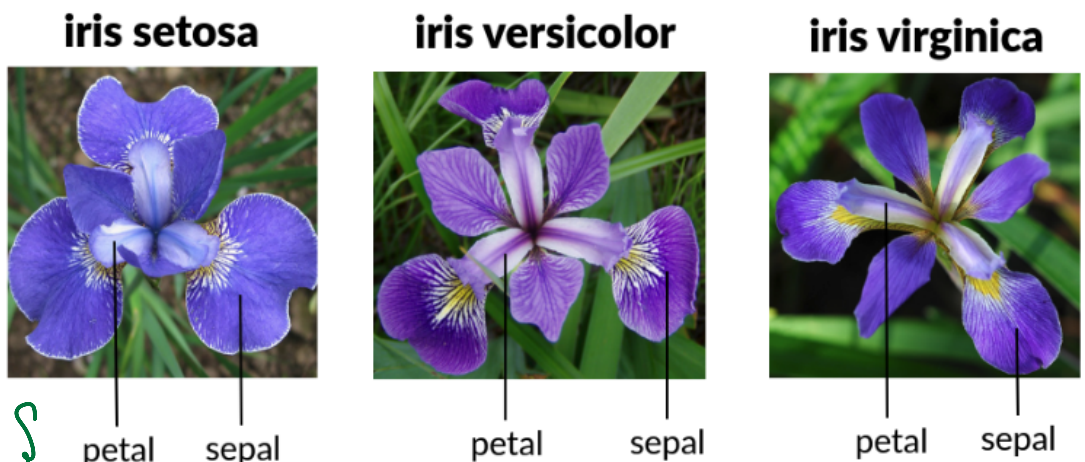
Linear Model

$f(x) = h(x)^T \omega$

Refined Features

Parameter

IRIS



1. SIFT
2. Convolution

MetaData or High-Level Features

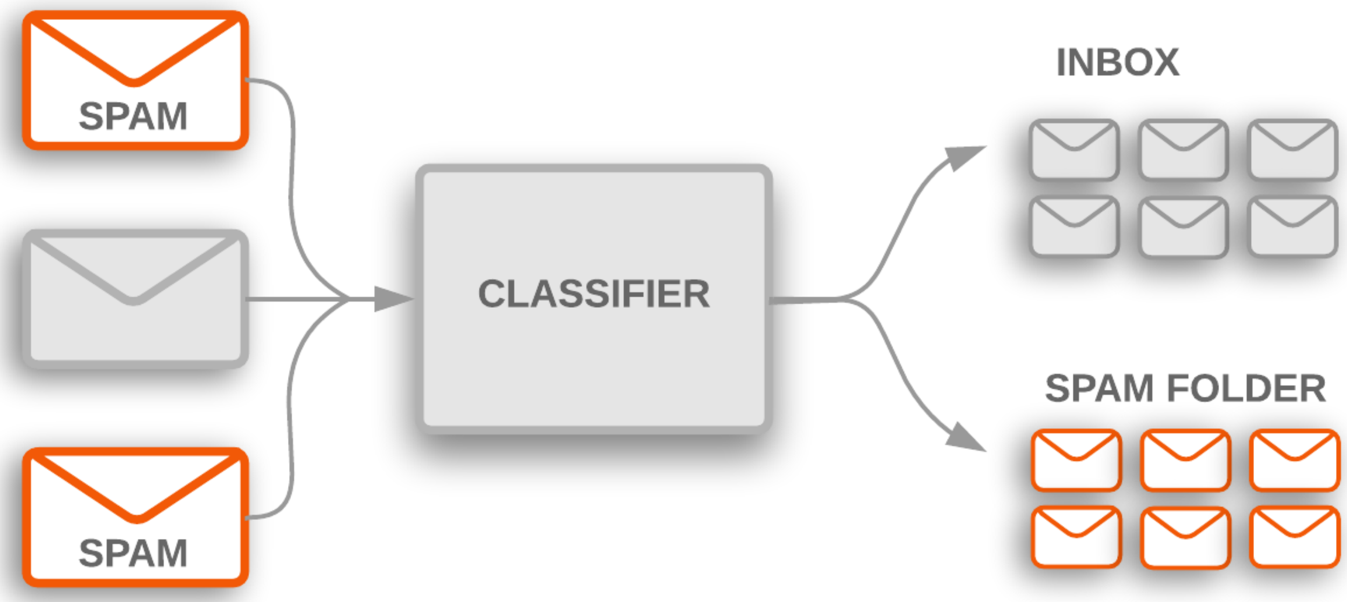
Example:- $\frac{\text{Petal Len}}{\text{Sepal Len}}$ = Ratio

Non-Linear Feature

Feature Engineering] 70% of ML = Feature Engineering (& Data)

Raw Image (Pixel Level Features) → Refined Features (Length of petal) → More Refined Features (Comb of the lengths)

Classification in Machine Learning



Difference between Classification and Regression

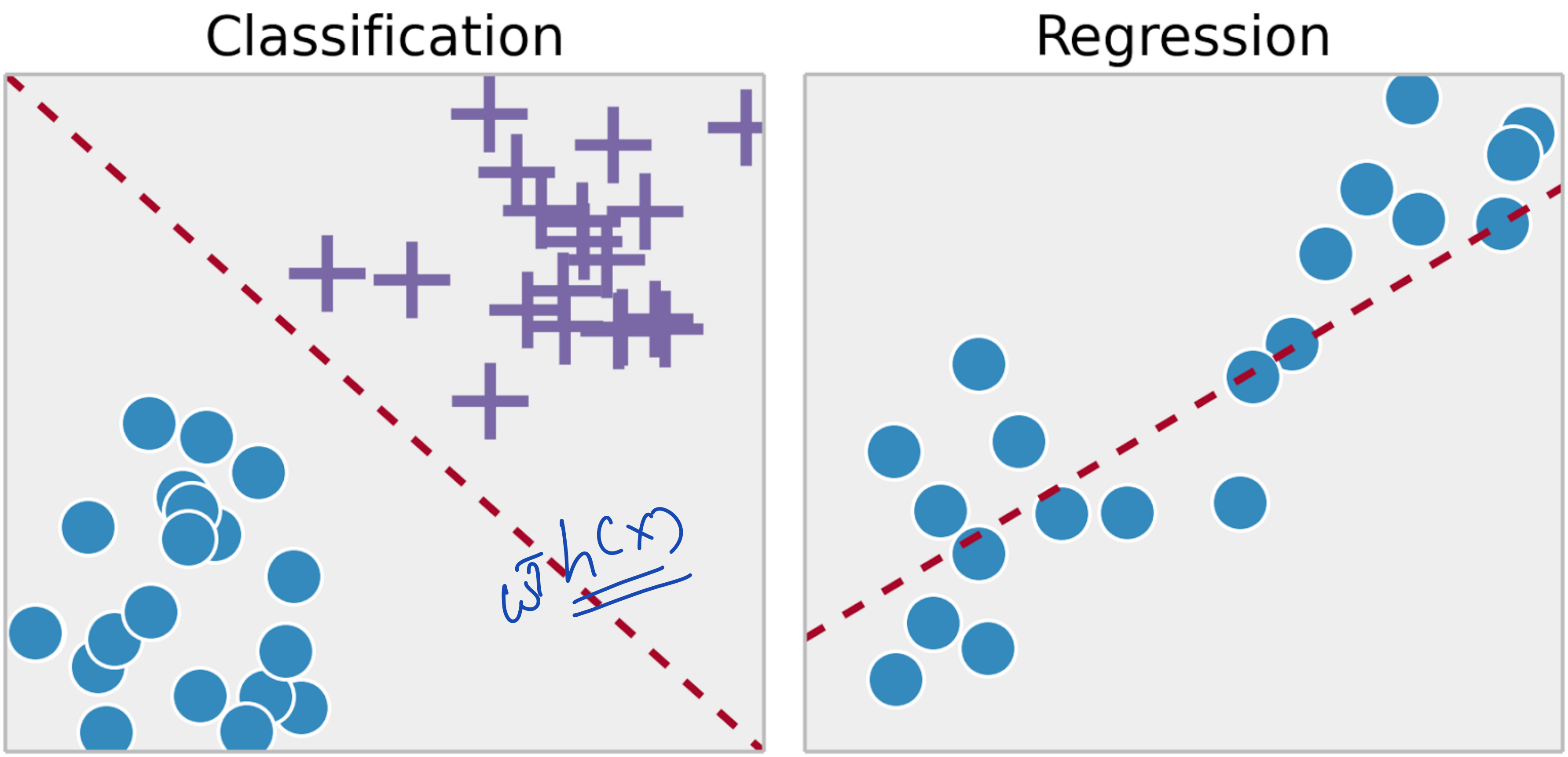
Simple difference

The target type in Regression is **numeric** whereas that in classification is **categorical**

Difference between Classification and Regression

Simple difference

The target type in Regression is **numeric** whereas that in classification is **categorical**



Types of Classification

Binary vs Multi-class classification

With binary categories, its a binary classification problem and with multiple categories, we have a multi-class classification.

Types of Classification

Binary vs Multi-class classification

With binary categories, its a binary classification problem and with multiple categories, we have a multi-class classification.

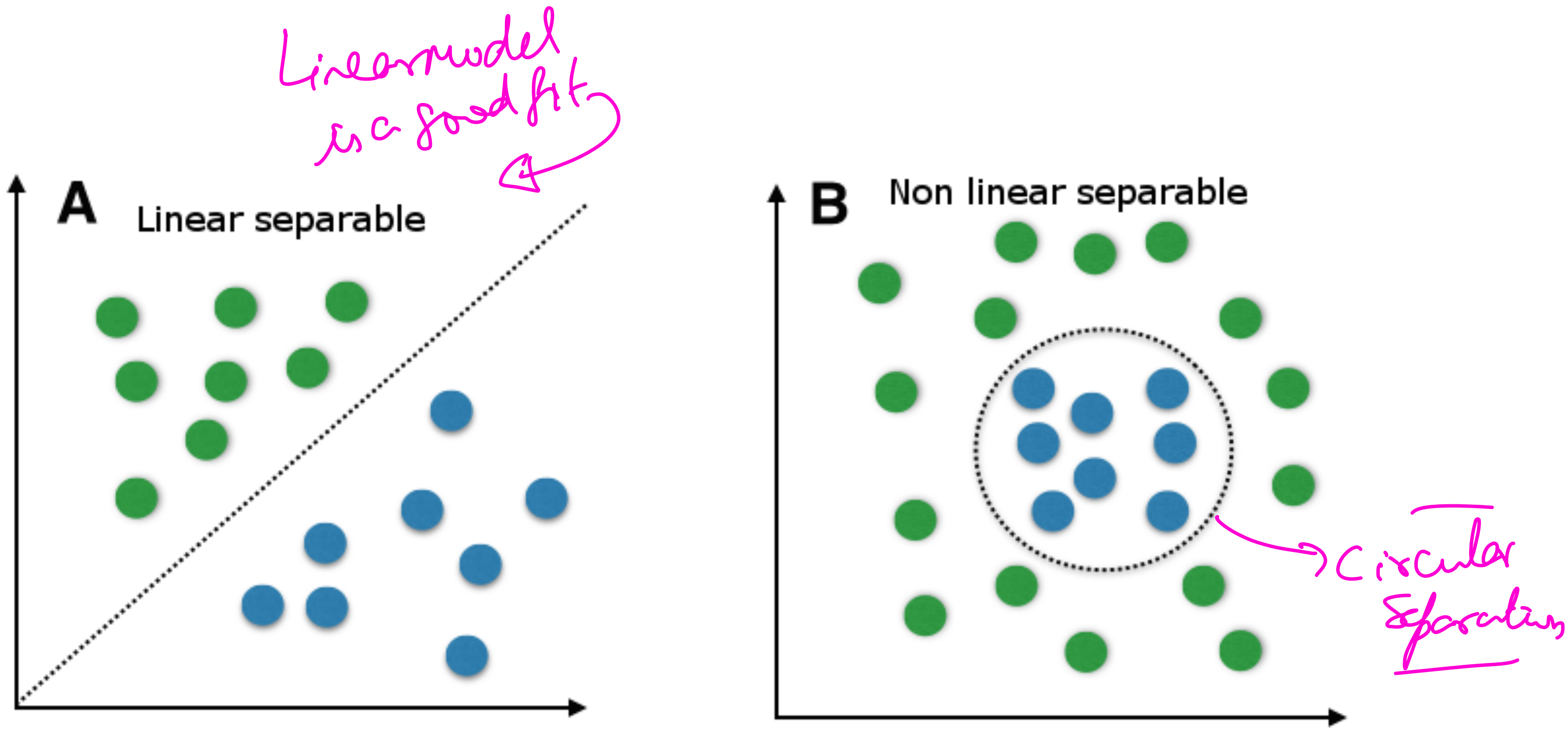
Target is called Label

For binary classification, the convention is to label the target as positive or negative. Example: Positive for spam and negative for not-spam

Spam Classification Example

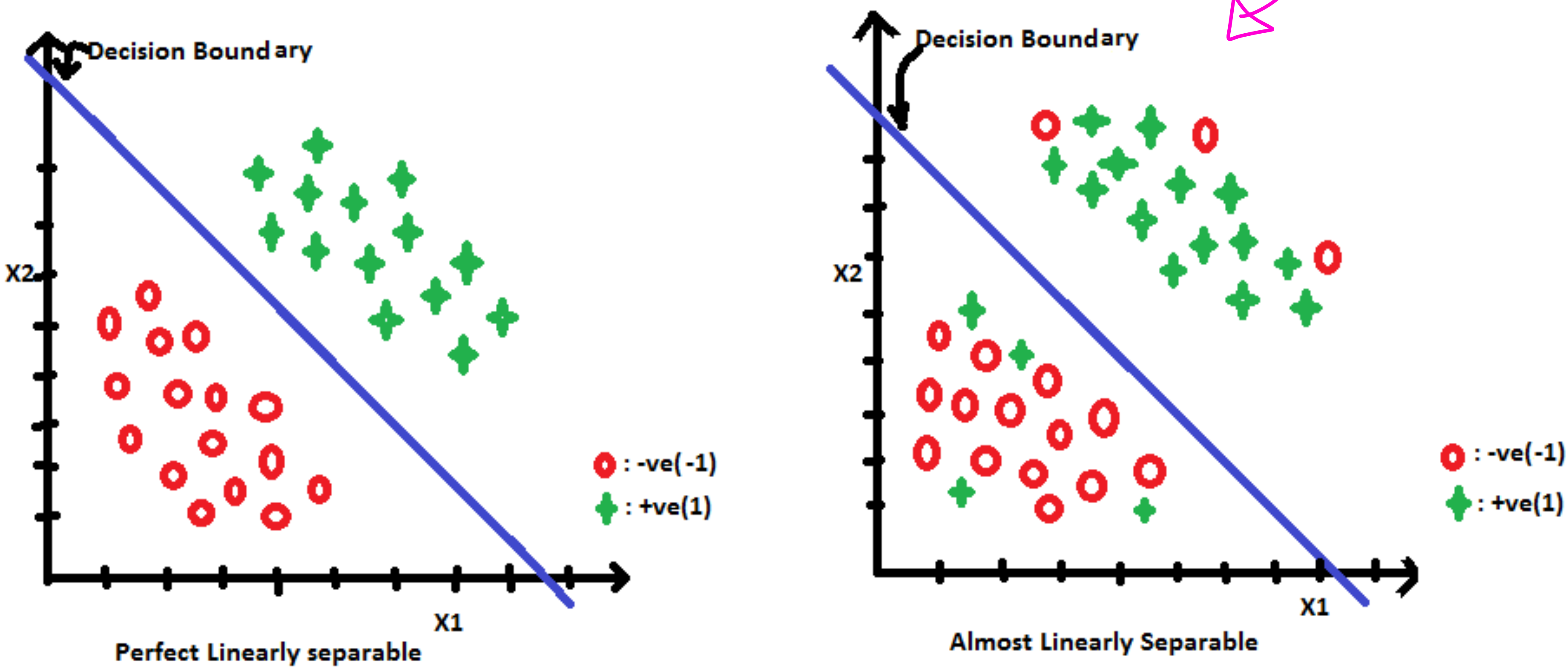
Email excerpt	Type	Label
Could you please respond by tomorrow?	Not-spam	-1
Congratulations!!! You have been selected...	Spam	$+1$
Looking forward to your presentation...	Not-spam	-1
$\dots$	$\dots$	$\dots$

Linear Separability

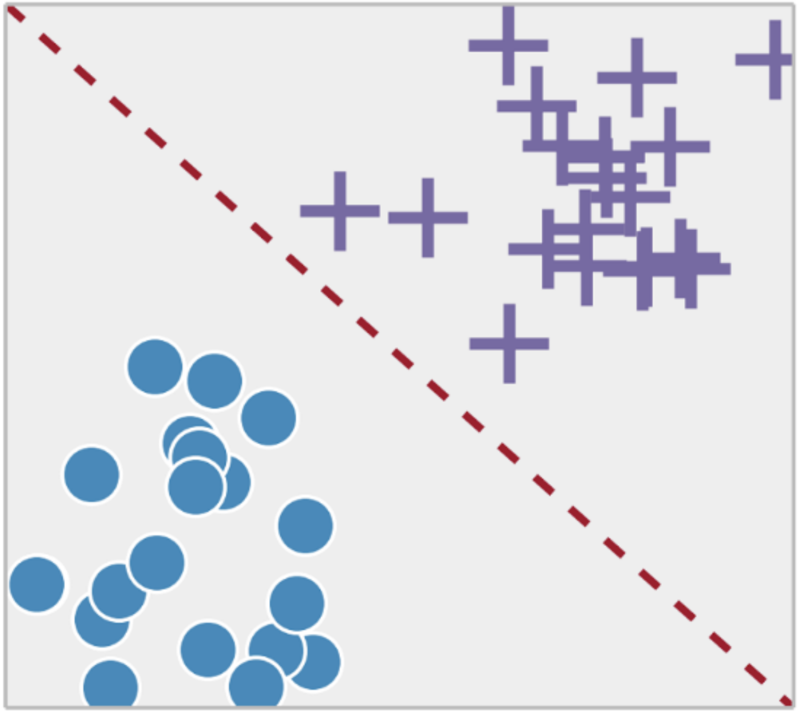


Approximate Linear Separability

Linear model is ok (good)



Logistic Regression



LR fundamentals

(SVM (support vector machines)

- Linear Model
- Want score $w^T x^i > 0$ for $y_i = +1$ and $w^T x_i < 0$ for $y_i = -1$!
- If linearly separable data, above is feasible. Else, minimize error in separability!!

Logistic Regression

Probability for a class

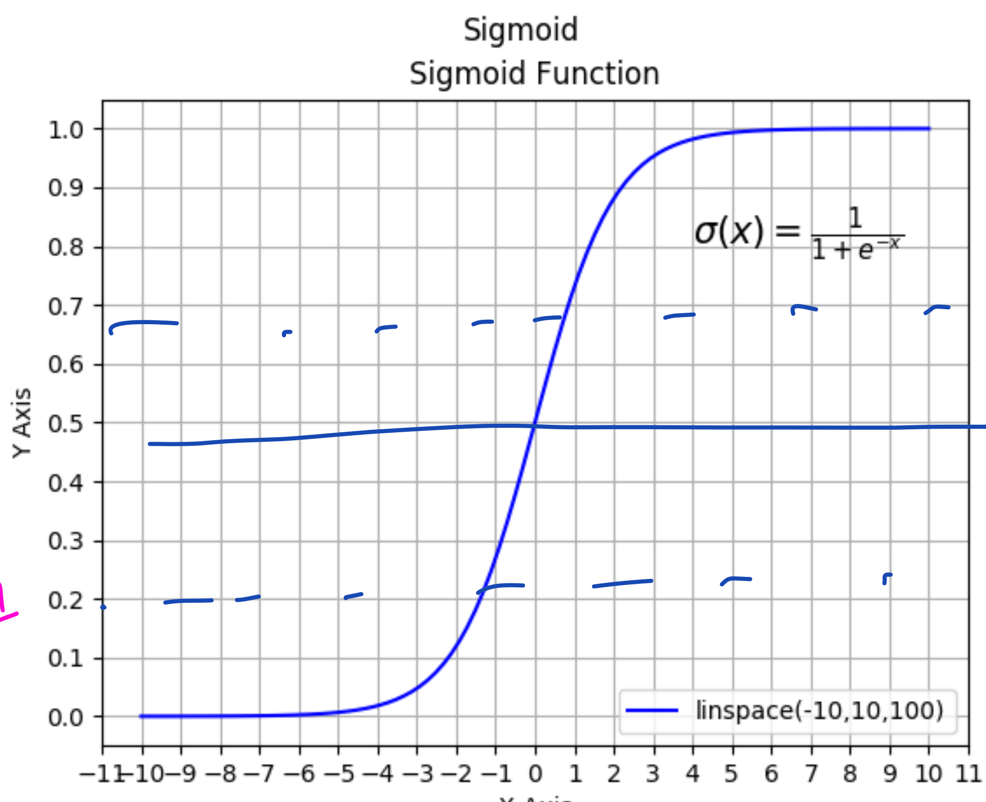
In LR, the score, $w^T x$ is converted to a probability through the sigmoid function. So we can talk about $P(\hat{y}^i = +1)$ or $P(\hat{y}^i = -1)$

Sigmoid Function

$P(z) = \underbrace{\frac{1}{1+e^{-z}}}_{\text{prob.}}$

score

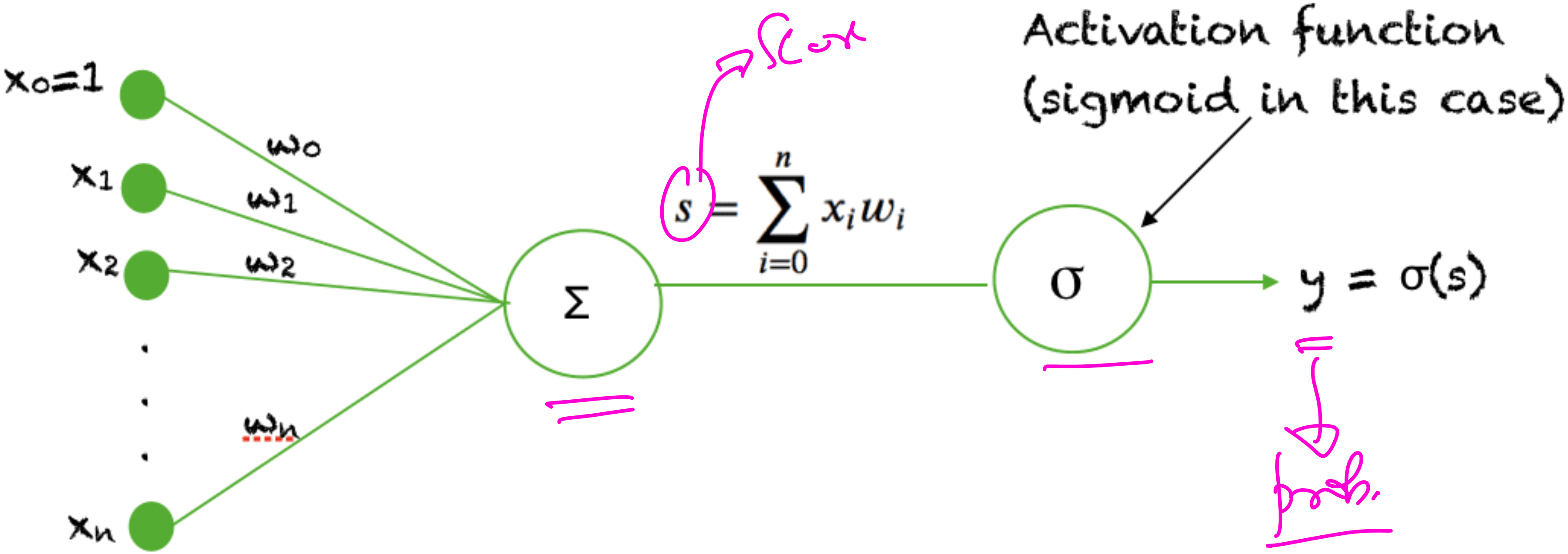
$0 \leq \quad \leq 1$



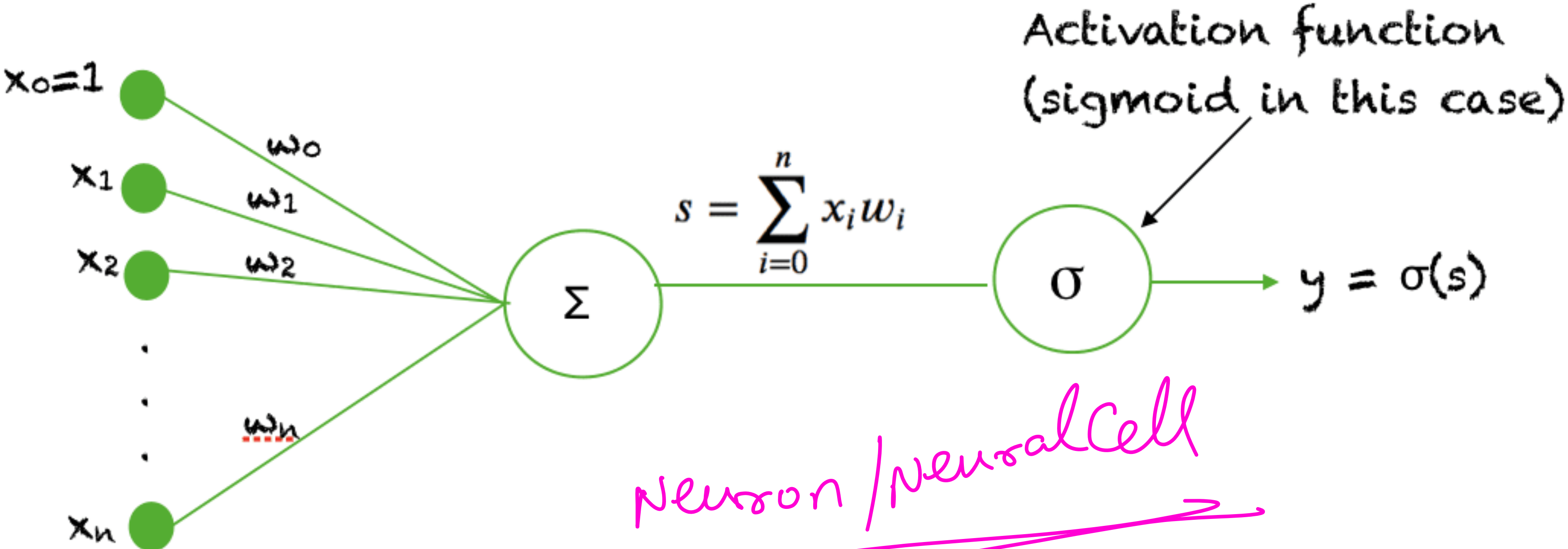
P +ve

P −ve

LR represented Graphically



LR vs Neural Networks/Deep Learning



Neuron / Neural Cell

Logistic Regression

LR Prediction

prob

$$\hat{y}_i = \frac{1}{1 + e^{-\hat{w}^T x^i}}$$

Raw Data

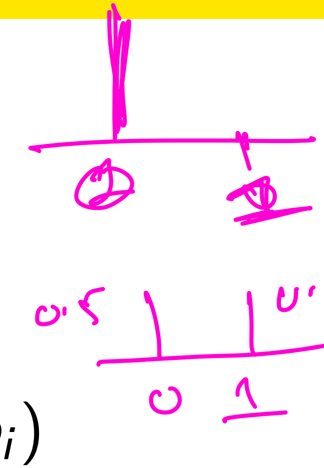
$\hat{w}^T h(x^i)$

Feature

LR Loss function - a.k.a what function do you optimize to learn a classifier?

LR Loss is based on the **cross-entropy** function

Entropy Function



Lowest Entropy

Highest Entropy

Entropy

What is it a measure of?

$$H(\mathbf{p}) = \sum_i -p_i \log(p_i)$$

where $\mathbf{p} \in^K$ is a probability distribution over K objects (e.g. K classes)

Binary Cross Entropy

Binary Cross Entropy is a measure of distance between two binary probability distributions!

$$H(p, q) = -p \log(q) - (1 - p) \log(1 - q)$$

Binary Cross Entropy Loss Function

LR Loss

Assume that $y_i = 0$ or $y_i = 1$ (i.e. the negative class has a label 0). Then the binary cross-entropy loss applies to LR:

$$\min_w \sum_i -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)$$

$p(y_i) = y_i$

$p(1-y_i) = 1-y_i$

$q(0) = 1-\hat{y}_i$

$q(1) = \hat{y}_i$

Binary Cross Entropy Loss Function

LR Loss

Assume that $y_i = 0$ or $y_i = 1$ (i.e. the negative class has a label 0). Then the binary cross-entropy loss applies to LR:

$$\min_w \sum_i \underbrace{-y_i \log(\hat{y}_i) - (1 - y_i)\log(1 - \hat{y}_i)}$$

Minimizing the LR loss

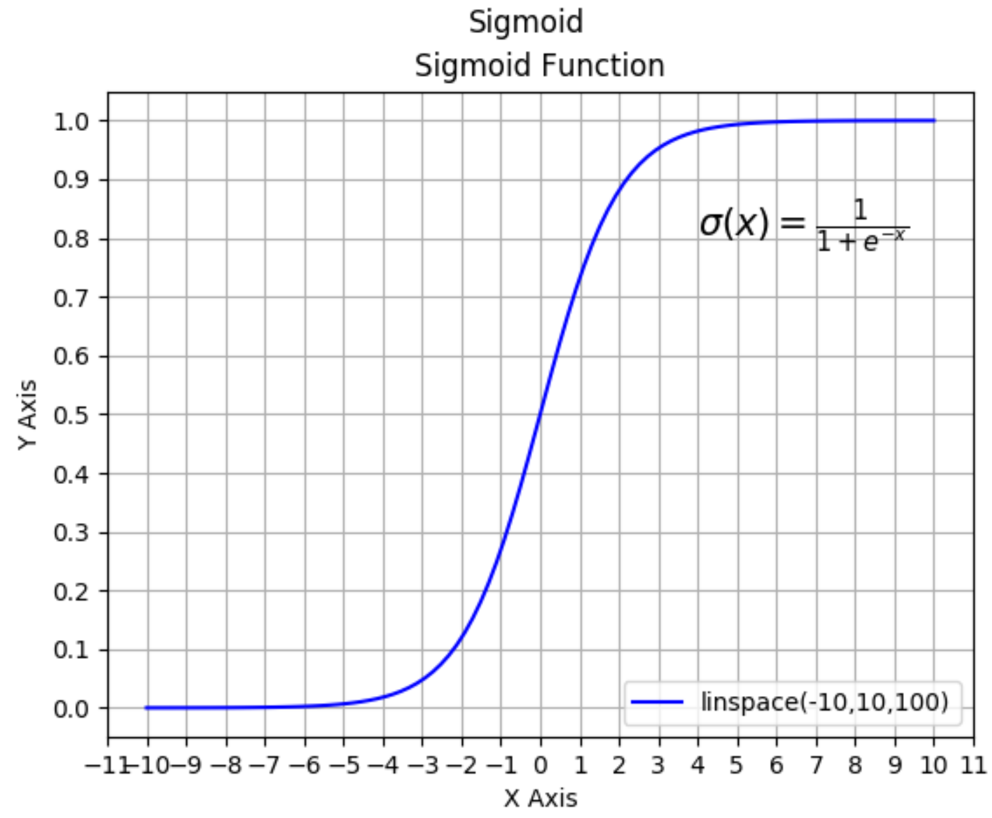
What distribution minimizes the LR loss function?

$$\min_{\hat{y}_i} \; -y_i \log(\hat{y}_i) - (1-y_i)\log(1-\hat{y}_i)$$

$$\frac{d}{d\hat{y}_i}(\;) = 0 \quad -\frac{y_i}{\hat{y}_i} + \frac{(1-y_i)}{1-\hat{y}_i} = 0 \Rightarrow \hat{y}_i = y_i$$

Score to a Probability

Sigmoid Function



Logistic Regression || Probabilistically Speaking

(Prediction)

Probability of a class

$$P(\hat{y}_i = 1) = \frac{1}{1 + e^{-\hat{w}^T x^i}}$$

Data pt. i

optimized

ICE #1 (2 mins)

Handling the math

Let $P(\hat{y}_i = 1) = \frac{1}{1+e^{-\hat{w}^T x^i}}$. What's the log probability of $P(\hat{y}_i = 0)$? (Working out the math on paper is recommended here!)

a) $\frac{e^{-\hat{w}^T x^i}}{1+e^{-\hat{w}^T x^i}}$

b) $\log\left(\frac{e^{-\hat{w}^T x^i}}{1+e^{-\hat{w}^T x^i}}\right)$

c) $\log\left(\frac{1}{1+e^{\hat{w}^T x^i}}\right)$

d) $\log\left(\frac{1}{1+e^{-\hat{w}^T x^i}}\right)$

Summary on Logistic Regression

1. Uses a linear model just like Linear Regression.

$s_i = \hat{w}^T h(x_i)$

Summary on Logistic Regression

1. Uses a linear model just like Linear Regression.
2. Assumes linear separability or approximate linear separability.

Summary on Logistic Regression

1. Uses a linear model just like Linear Regression.
2. Assumes linear separability or approximate linear separability.
3. For linear regression, $\hat{y}_i = \underbrace{\hat{w}^T x^i}_{\text{Score}}$. For LR, $\hat{y}_i = \underbrace{\frac{1}{1+e^{-\hat{w}^T x^i}}}_{\text{Prob}}$

Summary on Logistic Regression

1. Uses a linear model just like Linear Regression.
2. Assumes linear separability or approximate linear separability.
3. For linear regression, $\hat{y}_i = \hat{w}^T x^i$. For LR, $\hat{y}_i = \frac{1}{1+e^{-\hat{w}^T x^i}}$
4. Linear regression predicts numeric values that can range in $(-\infty, \infty)$. Logistic Regression predicts a probability of a class that ranges between $[0, 1]$.

Summary on Logistic Regression

1. Uses a linear model just like Linear Regression.
2. Assumes linear separability or approximate linear separability.
3. For linear regression, $\hat{y}_i = \hat{w}^T x^i$. For LR, $\hat{y}_i = \frac{1}{1+e^{-\hat{w}^T x^i}}$
4. Linear regression predicts numeric values that can range in $(-\infty, \infty)$. Logistic Regression predicts a probability of a class that ranges between $[0, 1]$.
5. Logistic Regression uses the Sigmoid or S-shaped function to go from a score to a probability!

Summary on Logistic Regression

1. Uses a linear model just like Linear Regression.
2. Assumes linear separability or approximate linear separability.
3. For linear regression, $\hat{y}_i = \hat{w}^T x^i$. For LR, $\hat{y}_i = \frac{1}{1+e^{-\hat{w}^T x^i}}$
4. Linear regression predicts numeric values that can range in $(-\infty, \infty)$. Logistic Regression predicts a probability of a class that ranges between $[0, 1]$.
5. Logistic Regression uses the Sigmoid or S-shaped function to go from a score to a probability!
6. Logistic Regression uses the log-loss or cross-entropy loss whereas Linear Regression uses the quadratic loss

$\|X\theta - y\|_2^2$

$\|Xw - y\|_2^2$

Summary on Logistic Regression

1. Uses a linear model just like Linear Regression.
2. Assumes linear separability or approximate linear separability.
3. For linear regression, $\hat{y}_i = \hat{w}^T x^i$. For LR, $\hat{y}_i = \frac{1}{1+e^{-\hat{w}^T x^i}}$
4. Linear regression predicts numeric values that can range in $(-\infty, \infty)$. Logistic Regression predicts a probability of a class that ranges between $[0, 1]$.
5. Logistic Regression uses the Sigmoid or S-shaped function to go from a score to a probability!
6. Logistic Regression uses the log-loss or cross-entropy loss whereas Linear Regression uses the quadratic loss
7. Logistic Regression loss can be derived as a MLE - So its well grounded in statistics.

Evaluating Classifiers!

ICE #2

Let's say you own an email server and want to provide a service to your email customers to help sort their emails into spam vs not-spam. So you go ahead and build a spam classifier on a training data set. Your data set has 100 spam emails and 900 non-spam emails. You notice that your classifier has 90% accuracy on the training data set and also your validation data set. Should you be happy with your classifier?

a) Yes
b) No
c) Maybe!
d) Something's fishy!

Evaluating classifiers

Can Impact metrics

Can Impact Model's Learning

Class imbalance

The above data set is an example of class imbalance. What can go wrong here?

Evaluating classifiers

Class imbalance

The above data set is an example of class imbalance. What can go wrong here?

Better metric than accuracy

→ 95%

Consider the **confusion matrix** for above Spam classification example with the trivial classifier (predict everything as non-spam).

	Predicted Positive	Predicted Negatives
Positives	0	100
Negatives	0	900

100 (spam)
900 (not spam)

1000

Evaluating classifiers

Better metric than accuracy

Consider the confusion matrix for above Spam classification example with the trivial classifier (predict everything as non-spam).

	Predicted Positive	Predicted Negatives
Positives	0	100
Negatives	0	900

Evaluating classifiers

Better metric than accuracy

Consider the confusion matrix for above Spam classification example with the trivial classifier (predict everything as non-spam).

	Predicted Positive	Predicted Negatives
Positives	0	100
Negatives	0	900

Better metric than accuracy

Accurcay is how many data points the classifier got right divided by the total data points. What's accuracy here?

Evaluating classifiers

Better metric than accuracy

Consider the confusion matrix for above Spam classification example with the trivial classifier (predict everything as non-spam).

	Predicted Positive	Predicted Negatives
Positives (P)	0	100
Negatives (N)	0	900

Evaluating classifiers

Better metric than accuracy

Consider the confusion matrix for above Spam classification example with the trivial classifier (predict everything as non-spam).

	Predicted Positive	Predicted Negatives
Positives (P)	0	100
Negatives (N)	0	900

Accuracy, Precision, Recall and F1-score

	Predicted Positive	Predicted Negatives
Positives (P)	TP	FN
Negatives (N)	FP	TN

Evaluating classifiers

Better metric than accuracy

Consider the confusion matrix for above Spam classification example with the trivial classifier (predict everything as non-spam).

	Predicted Positive	Predicted Negatives
Positives (P)	0	100
Negatives (N)	0	900

Evaluating classifiers

Better metric than accuracy

Consider the confusion matrix for above Spam classification example with the trivial classifier (predict everything as non-spam).

	Predicted Positive	Predicted Negatives
Positives (P)	(TP) 0	100 (FN)
Negatives (N)	(FP) 0	900 (TN)

100

Accuracy, Precision, Recall and F1-score

Precision (Pr) = TP/(TP + FP) = 0/0 ≈ 100% → How often do you make mistakes?

Recall (R) = TP/(TP + FN) = TP/P = 0% → coverage

F1-score = $\frac{2 \times Pr \times R}{Pr + R}$ (Harmonic Mean of Precision & Recall) = 0%

Accuracy (Acc) = $(TP + TN)/(P + N)$ = 90%

ICE #3

More Confusion!

Let's say we computed a **Confusion Matrix** for another Spam Classifier on a different data set and we obtained:

	Predicted Positive	Predicted Negatives
Positives (P)	50	50
Negatives (R)	100	400

100
500

Metrics!

Accuracy, **Pr**, **R** and **F1** are as follows:

a) $75\%, 0.2, 0.5, 0.285$
b) $80\%, 0.3, 0.4, 0.285$
c) $80\%, 0.5, 0.3, 0.1875$
d) $75\%, 0.3, 0.5, 0.1875$

$ACC = \frac{TP+TN}{total}$ $F1 = HM(Pr, R)$

$Pr = \frac{TP}{TP+FP}$

$R = \frac{TP}{TP+FN}$

(AUC)

Training the Binary Classification Model

Features

Target

x_1	x_2	x_3	x_4	x_5	x_6	x_7	y

N Rows

Data Table

Example: 70 : 10 : 20 Train-Val-Test data split

Choose 70% train data at random

x_1	x_2	x_3	x_4	x_5	x_6	x_7	y

7 rows (70%)

Example: 70 : 10 : 20 Train-Val-Test data split

Add 20% test data at random

x_1	x_2	x_3	x_4	x_5	x_6	x_7	y

Test Data
(20%)

Example: 70 : 10 : 20 Train-Val-Test data split

Remainder becomes validation data

x_1	x_2	x_3	x_4	x_5	x_6	x_7	y

← validation data

(15%)

Why Split Data into Train/Validation/Test ?

Training Dataset (Usually 70%)

We train on the training dataset - Learning from the data happens here.

Why Split Data into Train/Validation/Test ?

Training Dataset (Usually 70%)

We train on the training dataset - Learning from the data happens here.

Test Dataset (Usually 20%)

We evaluate our model on the test data set. Test data mimicks "unseen" data - So we don't bias our evaluation on data we have seen before.

Why Split Data into Train/Validation/Test ?

Training Dataset (Usually 70%)

We train on the training dataset - Learning from the data happens here.

Test Dataset (Usually 20%)

We evaluate our model on the test data set. Test data mimicks "unseen" data - So we don't bias our evaluation on data we have seen before.

Validation Dataset (Usually 10%)

We tune our hyper-parameters on the validation data set - So we get the best performing model on the validation set, which can then be tested on the test data set.

Why Split Data into Train/Validation/Test ?

Example: ℓ_1 regularized Logistic Regression

$$\min_{\omega} \underbrace{\ell(\omega)}_{\text{Binary Cross Entropy Loss}} + \lambda \underbrace{\|\omega\|_1}_{\ell_1 \text{ norm of } w}$$

λ Hyper-parameter

- Tune on validation Data Set

ICE #4

High Accuracy

You trained your favorite Image Classifier model to differentiate cat images from dog images and obtained a high precision and also high recall on your training data set. Does this mean your model is:

1. Performing well
2. Not performing well
3. Likely performing well but needs checking
4. Likely not performing well but needs checking

The phenomenon of Overfitting

Overfitting

Overfitting is when your model performs great on training data but doesn't match up on test data. To account for overfitting, we also have a validation data set.

The phenomenon of Overfitting

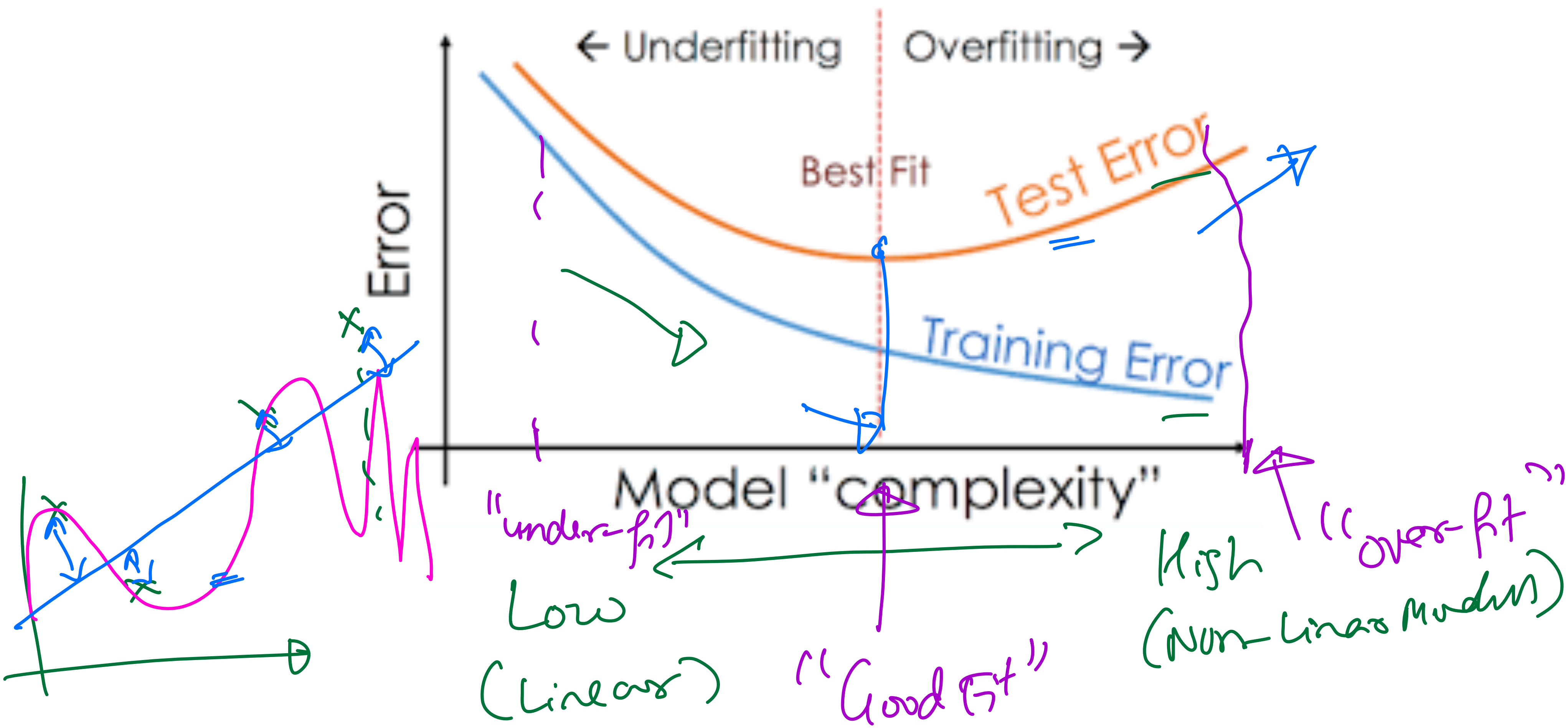
Overfitting

Overfitting is when your model performs great on training data but doesn't match up on test data. To account for overfitting, we also have a validation data set.

Overfitting Example

Your data set for classification of images has all roses red and all jasmines in yellow. In test data, you see a yellow rose and your model predicts it's a jasmine! Here, the model has overfit to the color yellow. This is also an issue with data coverage. In overfitting, the training error is low but test error is way higher. So data augmentation and data coverage can cover for issues like this!

The figure to remember for over-fitting!



ICE #5

Image Classifier

Consider that you have an image classifier model that takes a raw image as input and identifies if there is a cat present in the image or not. The image size is 500×500 pixels and the size of your training data is 10,000 images with their labels (cat or no cat). You decide to use a logistic regression model with a parameter corresponding to each pixel value. Is this model likely to:

1. Under-fit
2. Over-fit
3. Just fit
4. No fit

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!
- Consider the linear system $Xw = y$. This system is under-determined when $N < d$ (number of examples ¡ feature dimension)

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!
- Consider the linear system $Xw = y$. This system is under-determined when $N < d$ (number of examples ¡ feature dimension)
- Infinitely many solutions when $N < d$!

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!
- Consider the linear system $Xw = y$. This system is under-determined when $N < d$ (number of examples ¡ feature dimension)
- Infinitely many solutions when $N < d$!
- ICE #0: Find a solution for $1^T w = 1$ where $w \in^2$.

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!
- Consider the linear system $Xw = y$. This system is under-determined when $N < d$ (number of examples ¡ feature dimension)
- Infinitely many solutions when $N < d$!
- ICE #0: Find a solution for $1^T w = 1$ where $w \in^2$.
- Unique solution when $N > d$ and when X has full rank!

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!
- Consider the linear system $Xw = y$. This system is under-determined when $N < d$ (number of examples ¡ feature dimension)
- Infinitely many solutions when $N < d$!
- ICE #0: Find a solution for $1^T w = 1$ where $w \in^2$.
- Unique solution when $N > d$ and when X has full rank!
- Over-fitting happens when number of data points comparable to the number of attributes/features (order)

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!
- Consider the linear system $Xw = y$. This system is under-determined when $N < d$ (number of examples ¡ feature dimension)
- Infinitely many solutions when $N < d$!
- ICE #0: Find a solution for $1^T w = 1$ where $w \in^2$.
- Unique solution when $N > d$ and when X has full rank!
- Over-fitting happens when number of data points comparable to the number of attributes/features (order)
- `Solution A` to overfitting: Increase number of examples so that $N >> d$

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!
- Consider the linear system $Xw = y$. This system is under-determined when $N < d$ (number of examples ¡ feature dimension)
- Infinitely many solutions when $N < d$!
- ICE #0: Find a solution for $1^T w = 1$ where $w \in^2$.
- Unique solution when $N > d$ and when X has full rank!
- Over-fitting happens when number of data points comparable to the number of attributes/features (order)
- `Solution A` to overfitting: Increase number of examples so that $N >> d$
- `Solution B`: Decrease number of features so that $d << N$

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!
- Consider the linear system $Xw = y$. This system is under-determined when $N < d$ (number of examples ¡ feature dimension)
- Infinitely many solutions when $N < d$!
- ICE #0: Find a solution for $1^T w = 1$ where $w \in^2$.
- Unique solution when $N > d$ and when X has full rank!
- Over-fitting happens when number of data points comparable to the number of attributes/features (order)
- `Solution A` to overfitting: Increase number of examples so that $N >> d$
- `Solution B`: Decrease number of features so that $d << N$
- `Solution C`: Regularization! (Perhaps accomplish B as well along the way)

Regularization for classification

Large weights

Is a sign of over-fitting. With large weights - Small changes in feature value can throw the predictions off.

Regularization for classification

Large weights

Is a sign of over-fitting. With large weights - Small changes in feature value can throw the predictions off.

l_2 Regularization

Regularized loss (objective function):

$$\min_{w} l(w) + \lambda \|w\|_2^2$$

Regularization for classification

Large weights

Is a sign of over-fitting. With large weights - Small changes in feature value can throw the predictions off.

l_2 Regularization

Regularized loss (objective function):

$$\min_{w} l(w) + \lambda \|w\|_2^2$$

l_1 Regularization

Regularized loss (objective function):

$$\min_{w} l(w) + \lambda \|w\|_1$$

Summary so far

1. Supervised Learning and Binary Classification
2. Logistic Regression
3. Metrics for measuring goodness of a classifier
4. Confusion Matrix