Ahmed Ibrahim

ECE 9039/9309 MACHINE LEARNING

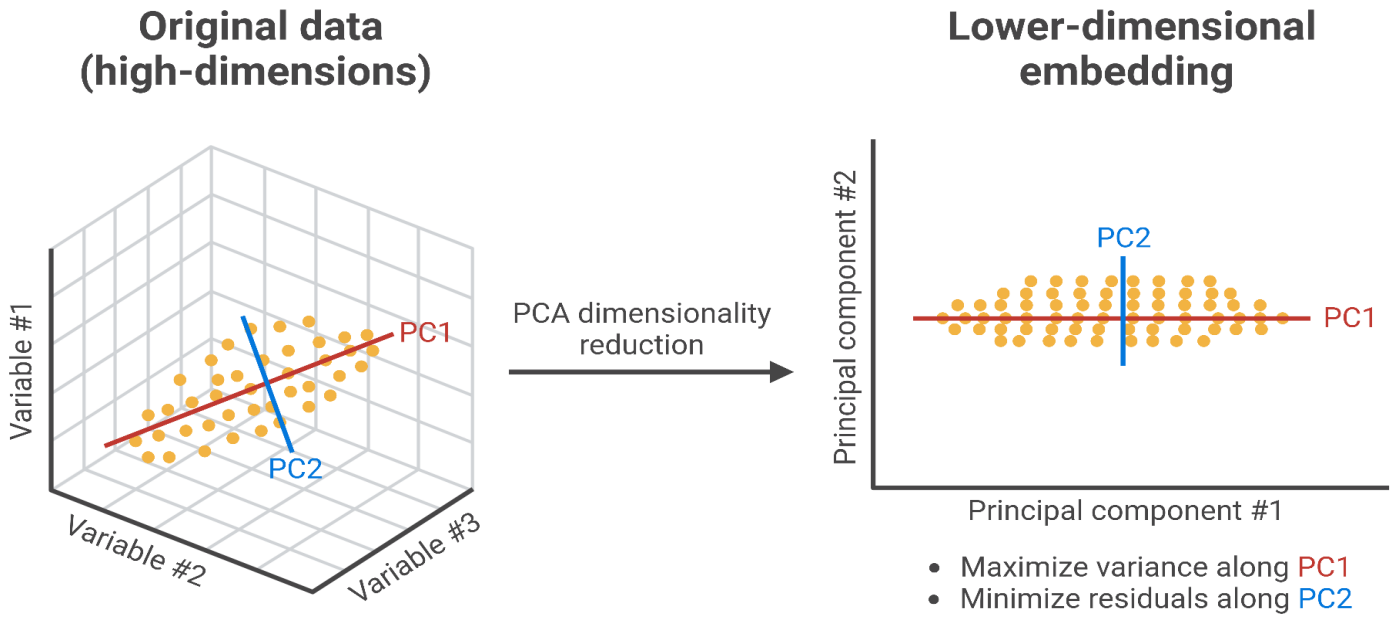
Outline

- Recap
- Principal Component Analysis (PCA) cont.
- Classification
  - How does classification work?
  - Types of Classifiers
  - Linear Classifiers
- Classification Metrics
- Metrics vs Loss
- Logistic Regression

Last Lecture

- Multiple Linear Regression
  - Loss Functions
  - Evaluating Model Performance
  - Sources of Error
- Feature Construction, Manipulation, and Selection
  - Principal Components Analysis (PCA)

Principal Components Analysis (PCA)



- It transforms a set of possibly correlated variables into a smaller number of variables known as principal components
- These principal components capture the maximum variance present in the data in a reduced number of dimensions.

Source:https://www.biorender.com/template/principal-component-analysis-pca-transformation

Recall: Principal Components Analysis (PCA)

- To apply PCA to a dataset, you need to follow the following steps:
  1. Standardize the Data – PCA is affected by scale, so it's important to scale the features in the data before applying PCA.
  2. Compute the Covariance Matrix – This matrix represents the covariance between each pair of features in the data.
  3. Calculate the Eigenvalues and Eigenvectors of the Covariance Matrix – These will determine the principal components.
  4. Sort Eigenvalues and Eigenvectors – Sort the eigenvalues and their corresponding eigenvectors in descending order. The eigenvectors with the highest eigenvalues are the principal components.
  5. Project the Data Onto the Principal Components – This will result in a new dataset of possibly lower dimensions.

Covariance Matrix

- A Covariance Matrix is a square matrix that encapsulates a dataset's covariance between multiple variables.
- **Covariance** is a measure of the **joint variability** of two random variables.
- The covariance matrix for a dataset of two variables is a 2×2 matrix, where each element represents the covariance between a pair of these variables. The following is a general form of a covariance matrix.

$$\Sigma = \begin{bmatrix} \mathrm{Var}(X_1) & \mathrm{Cov}(X_1, X_2) \\ \mathrm{Cov}(X_2, X_1) & \mathrm{Var}(X_2) \end{bmatrix}$$

Tutorial

Supervised Learning:
Classification

Classification

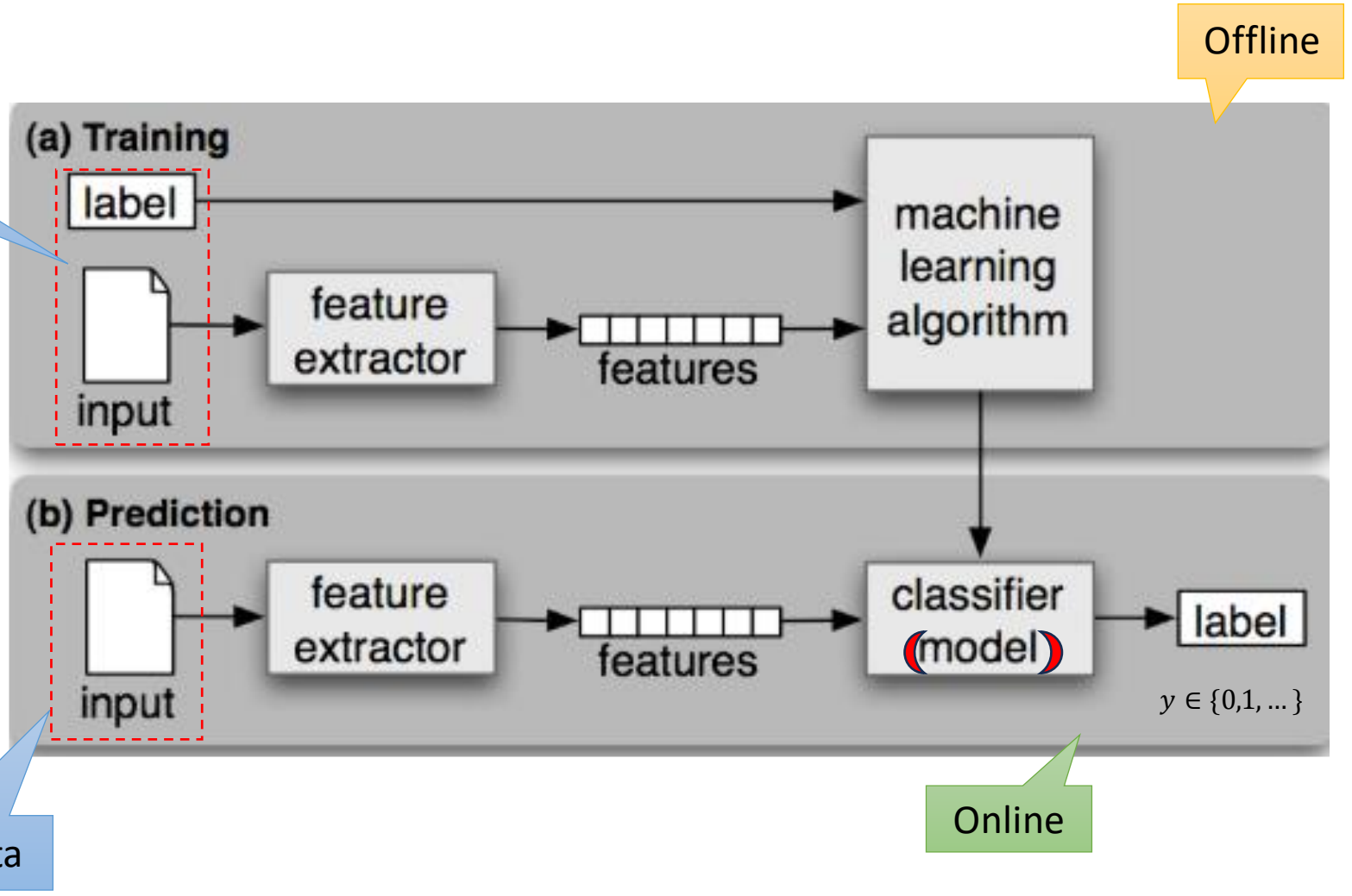
Classification is a **supervised learning** approach in machine learning where the goal is to predict the **categorical class labels** of data based on past observations.

Key Characteristics

- It requires **labeled** training data to **learn** the relationship between input features and the target class.
- The output variable is **discrete** and belongs to a finite set of categories like {yes, no}, {spam, not-spam}, etc.
- Classification can be Linear or Non-linear.

How does classification work?

- Classification operates by **learning** patterns from labeled data and using these **learned patterns** to categorize new, unseen data into specific groups or classes.



Types of Classifiers

Linear Classifiers

- Linear classifiers make a classification decision based on the value of a linear combination of the features.
- Examples:
  - Perceptron
  - Logistic Regression
  - Linear Support Vector Machine (SVM)

Non-Linear Classifiers

- Nonlinear classifiers can model more complex relationships. They do not assume a linear relationship between the features and the target variable.
- Examples:
  - Decision Trees
  - Neural Networks
  - Random Forests

Other Types of Classifiers

Probabilistic Classifiers

- These classifiers, such as **Naive Bayes**, use probability models for classification.
- They calculate the probability of each class given the input data and classify the instance into the class with the **highest probability**.
- Parametric vs. Non-Parametric Classifiers

Instance-Based Classifiers

- Instance-based methods, such as **K-Nearest Neighbors (KNN)**, classify new instances based on the instances of training data closest to the feature space.
- They are often simple to implement and understand but can be computationally intensive, especially with large datasets.
- Instance-Based vs. Model-based Classifiers

Ensemble Methods

- These methods, like Random Forests, Gradient Boosting Machines (GBM), and Extreme Gradient Boosting (XGBoost), combine the predictions of multiple base estimators to improve robustness and accuracy.
- They are powerful in handling various types of data and can improve the performance of both linear and nonlinear base classifiers.

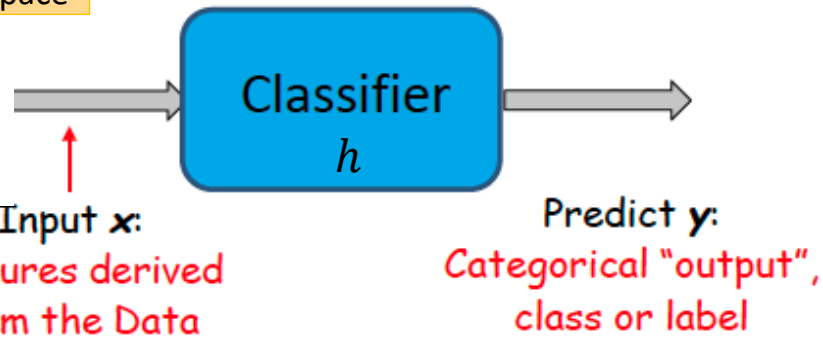
Linear Classifiers

A Linear Classifier

Feature Space

Solution Space

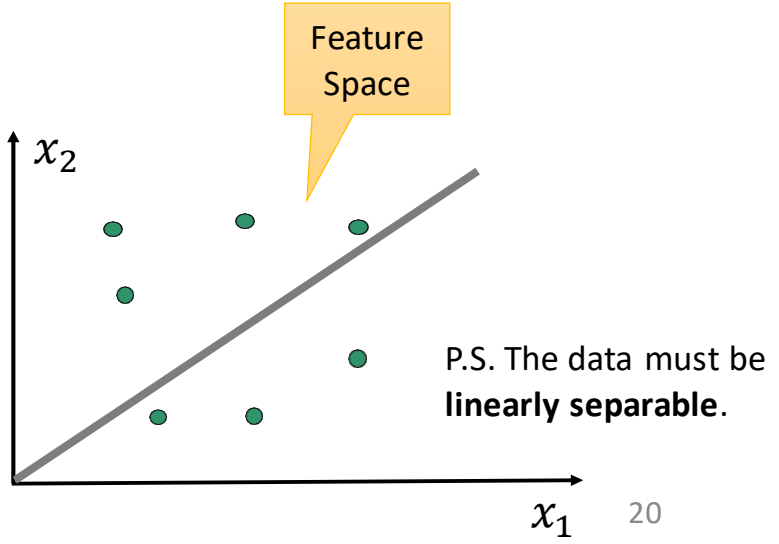
- We are interested in mapping the input x ∈ X to a label t ∈ y, $y \in \{-1, 1\}$
- y is the variable that we are trying to predict using x ∈ X
- Can we do this task using what we have learned in previous lectures? (Simple hack: Ignore that the output is categorical)



$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = h$$

$$h = \text{sign}(\theta^T x + \theta_0) = \begin{cases} +1 & \text{if } \theta^T x + \theta_0 > 0 \\ -1 & \text{otherwise} \end{cases}$$

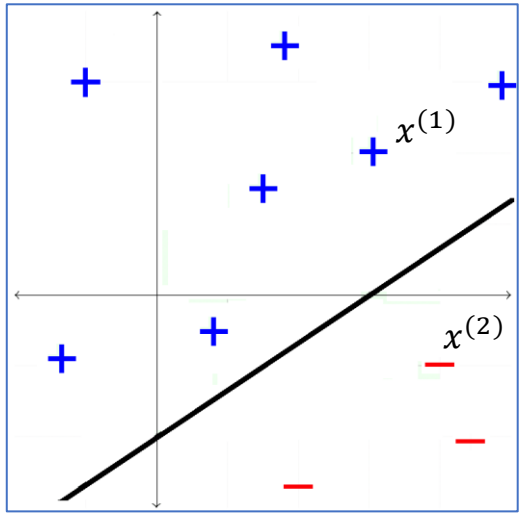
- +1: If the input number is positive, the sign function returns +1.
- -1: If the input number is negative, the sign function returns -1.



P.S. The data must be **linearly separable**.

Example

- Let $h(x)$ be the linear classifier defined by $\theta = \begin{vmatrix} -1 \\ 1.5 \end{vmatrix}$, $\theta_0 = -3$
- The diagram below shows several points classified by h. In particular, let $x^{(1)} = \begin{vmatrix} 3 \\ 2 \end{vmatrix}$ and $x^{(2)} = \begin{vmatrix} 4 \\ -1 \end{vmatrix}$



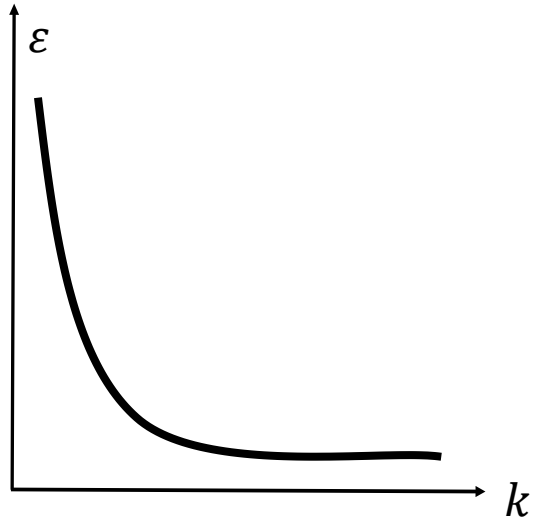
$$y^{(1)} = \text{sign}\left(\begin{bmatrix} -1 & 1.5 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} + 3 \right) = \text{sign}(3) = +1$$

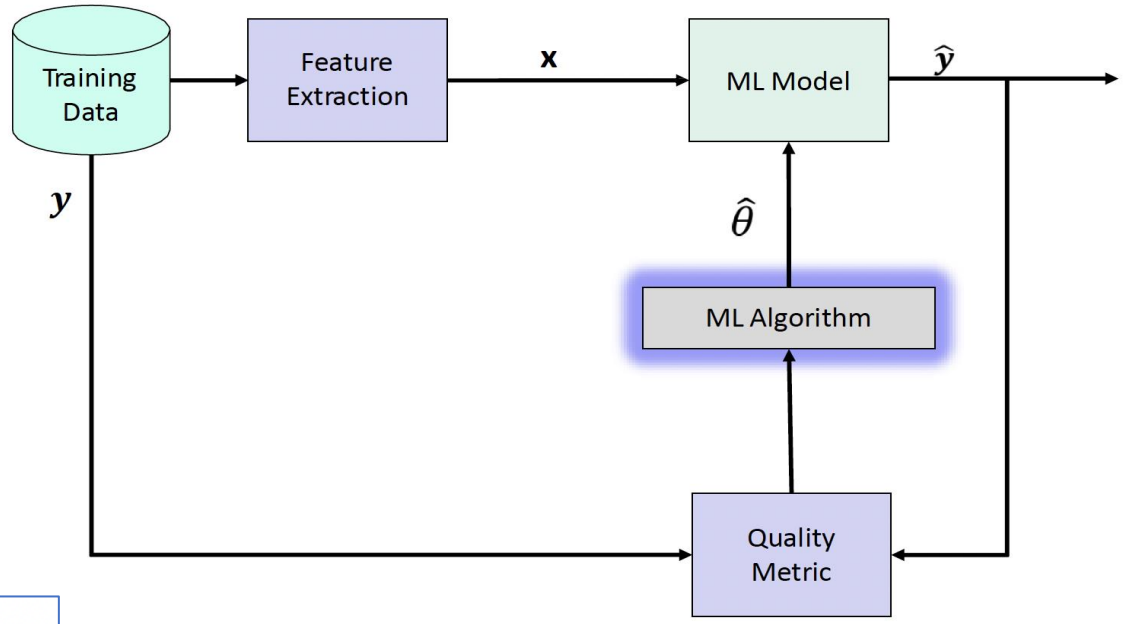
$$y^{(2)} = \text{sign}\left(\begin{bmatrix} -1 & 1.5 \end{bmatrix} \begin{bmatrix} 4 \\ -1 \end{bmatrix} + 3 \right) = \text{sign}(-2.5) = -1$$

- In supervised learning we are given a training data set of the form:

$$\mathcal{D}_n = \left\{ \left(x^{(1)}, y^{(1)} \right), \ldots, \left(x^{(n)}, y^{(n)} \right) \right\}$$

A Linear Classifier (cont.)





$$\hat{y} = Sign(\boldsymbol{\theta}^T \boldsymbol{x})$$
$$-1 or +1$$

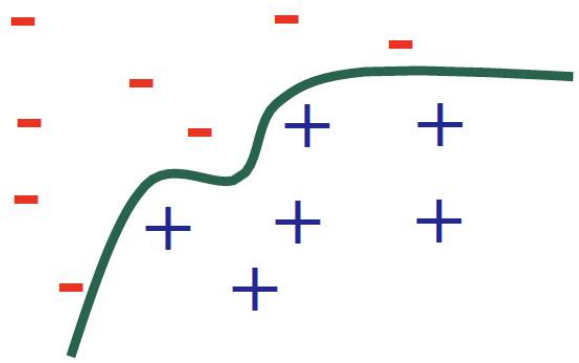
RANDOM-LINEAR-CLASSIFIER($\mathcal{D}_n$, k, d)

1 **for** j = 1 **to** k
2 randomly sample $\left(\theta^{(j)}, \theta_0^{(j)}\right)$ from $(\mathbb{R}^d, \mathbb{R})$
3 $j^* = \arg\min_{j\in\{1,\ldots,k\}} \mathcal{E}_n\left(\theta^{(j)}, \theta_0^{(j)}\right)$
4 **return** $\left(\theta^{(j^*)}, \theta_0^{(j^*)}\right)$

Given a training set D_n and a classifier h, we can define the *empirical classification error* or *training error* of h to be

$$\varepsilon(h) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} 1 & h\left(x^{(i)}\right) \neq y^{(i)} \\ 0 & otherwise \end{cases}$$

Decisions Boundaries



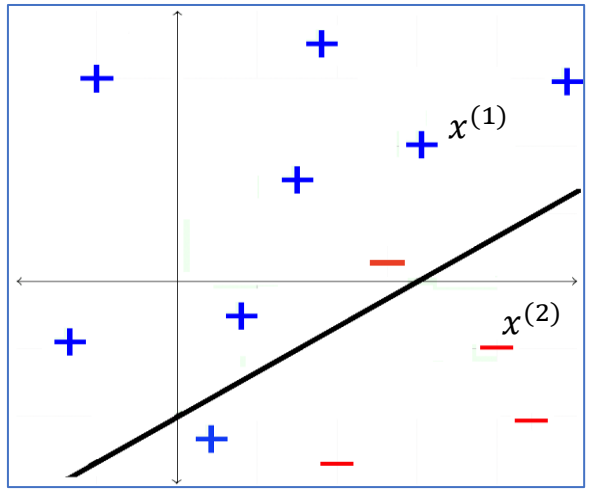
Non-Linear Decision Boundary

- Linear Classifiers
  - When only 2 coefficients are non-zero
    - The decision boundary is a line
  - When only 3 coefficients are non-zero
    - The decision boundary is a **Plane**
  - When many coefficients are non-zero
    - The decision boundary is a **hyperplane**
- For more general classifiers
  - Decision boundaries will take complicated shapes

Confusion Matrix

- A confusion matrix is a table often used in classification tasks in ML to visualize the performance of an algorithm.
- The matrix is usually set up with actual values in rows and predicted values in columns, or vice versa.
- This matrix helps in understanding not just the errors of the model but, more importantly, the type of errors that are being made.

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True Positive: 7	False Negative: 1
	Negative	False Positive: 1	True Negative: 3



Example

- Imagine we have a dataset of hospital patients:
  - COVID-19 **Positive** Cases (Disease Present): **100 samples**
  - COVID-19 **Negative** Cases (Disease Absent): **100 samples**
- Test Results:
  - True Positives (TP): The test correctly identifies 80 out of 100 disease cases.
  - False Negatives (FN): The test misses 20 out of 100 disease cases.
  - True Negatives (TN): The test correctly identifies 90 out of 100 healthy cases.
  - False Positives (FP): The test incorrectly identifies 10 out of 100 healthy cases as diseased.

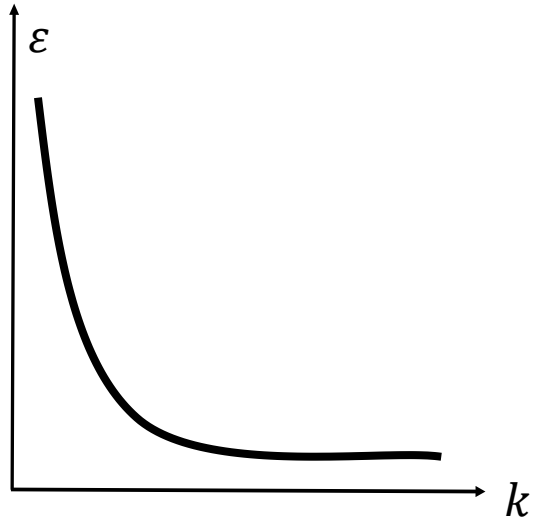
Confusion Matrix

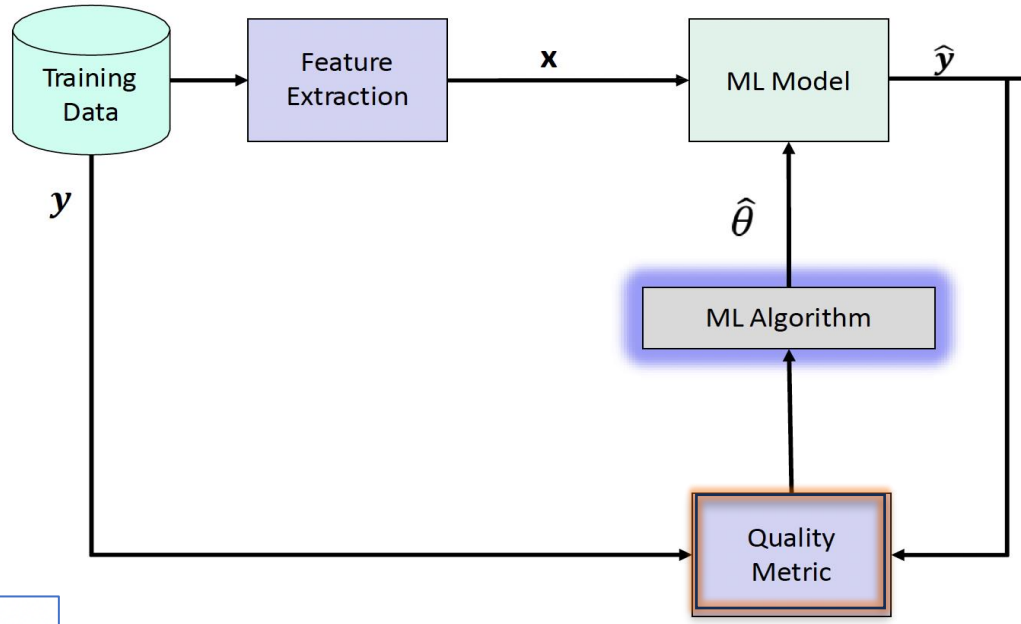
		Predicted	
		Positive	**Negative**
Actual	**Positive**	TP: 80	FN: 20
	Negative	FP:10	TN:90

Calculating Positive Rates (metrics):

- **True Positive Rate (TPR)**: TP / (TP + FN) = 80 / (80 + 20) = 0.8
- **False Positive Rate (FPR)**: FP / (FP + TN) = 10 / (10 + 90) = 0.1

Recall: A Linear Classifier (cont.)





$$\hat{y} = Sign(\boldsymbol{\theta}^T \boldsymbol{x})$$
$$-1 or +1$$

RANDOM-LINEAR-CLASSIFIER($\mathcal{D}_n$, k, d)
1 **for** j = 1 **to** k
2 randomly sample $\left(\theta^{(j)}, \theta_0^{(j)}\right)$ from $(\mathbb{R}^d, \mathbb{R})$
3 $j^* = \arg\min_{j\in\{1,\ldots,k\}} \mathcal{E}_n\left(\theta^{(j)}, \theta_0^{(j)}\right)$
4 **return** $\left(\theta^{(j^*)}, \theta_0^{(j^*)}\right)$

Given a training set D_n and a classifier h, we can define the *empirical classification error* or *training error* of h to be

$$\varepsilon(h) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} 1 & h\left(x^{(i)}\right) \neq y^{(i)} \\ 0 & otherwise \end{cases}$$

Precision & Recall Metrics

- **Recall –** It's the ratio of positive predictions to the actual number of positives.

$$R = \frac{TP}{TP + FN} = \frac{TP}{\text{all groundtruth instances}}$$

- **Precision –** It's the ratio of positive predictions to the total number of positive predictions.

$$P = \frac{TP}{TP + FP} = \frac{TP}{\text{all predicted}}$$

Recall: Example

- Imagine we have a dataset of hospital patients:
  - COVID-19 **Positive** Cases (Disease Present): **100 samples**
  - COVID-19 **Negative** Cases (Disease Absent): **100 samples**
- Test Results:
  - True Positives (TP): The test correctly identifies 80 out of 100 disease cases.
  - False Negatives (FN): The test misses 20 out of 100 disease cases.
  - True Negatives (TN): The test correctly identifies 90 out of 100 healthy cases.
  - False Positives (FP): The test incorrectly identifies 10 out of 100 healthy cases as diseased.

$$R = \frac{TP}{TP + FN} = \frac{TP}{\text{all groundtruth instances}}$$

$$\text{Recall} = \frac{80}{80+20} = \frac{80}{100} \approx 0.8$$

$$P = \frac{TP}{TP + FP} = \frac{TP}{\text{all predicted}}$$

$$\text{Precision} = \frac{80}{80+10} = \frac{80}{90} \approx 0.889$$

F1 score Metric

- The F1 score is a statistical measure used to evaluate the accuracy of a classification model.
- It's harmonic mean of **precision (*P*)** and **recall (*R*)**.

$$F1 = 2\frac{P \cdot R}{P + R}$$

- The best value of the F1 score is 1, and the worst is 0.
- It is particularly useful when the **balance** between precision and recall is important.

Another Example

- Imagine you have a machine learning model designed to determine whether a credit card transaction is fraudulent.
- The predictive performance of this model is summarized in the following confusion matrix:
- You can calculate Precision and Recall as below:

  Precision = 50 / 50+10 = 0.83

  Recall = 50 / 50+5 = 0.91

- Thus, the F1 score is calculated as:

  F1 Score = 2 * (0.83 * 0.91) / 0.83 + 0.91

  = 0.87

Confusion Matrix

Total of instances: 165	Actual Positives	Actual Negatives
Predicted Positives	TP: 50	FP: 10
Predicted Negatives	FN: 5	TN: 100

Therefore, the model has an F1 Score of 0.87, which is close to 1

Accuracy

- Accuracy is a commonly used metric to evaluate the performance of a classification model. It is calculated as the ratio of the number of correct predictions to the total number of predictions made. The formula for calculating accuracy is:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions Made}} \quad \Rightarrow \quad \text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- While accuracy is a straightforward metric, it may not always be the best indicator of a model's performance, especially in cases where the dataset is **imbalanced** (i.e., one class is significantly more frequent than the other).
- In such cases, other metrics like **precision**, **recall**, **F1 score**, or ROC-AUC (explain it later) might provide a more detailed understanding of the model's performance.

Metrics vs Loss

- We are mainly interested in how well a classifier works on a dataset, which is measured by its metrics.
- Usually, we can't **directly** make a classifier better based on its metrics.
- Our loss function needs to be reflective of the problem we're trying to solve. Then, we'll cross our fingers and hope it produces models that perform well on our dataset.

Attendance

You can use the provided link if you don't have a mobile phone or if your phone lacks a QR-Code reader – https://rebrand.ly/ECEJan30

Tutorial

Can we always separate the classes?

Causes of non-perfect separation:

- Model is too simple
- Noise in the inputs
- Simple features that do not account for all variations
- Errors in data targets (mis-labeling)

Probabilistic Classifiers

Conditional Probability Review

- A **conditional probability** is the probability of an event A, given that another event B has already occurred.
- The conditional probability of *A* given *B* is written as P(A|B) and is read as "the probability of A given B".
- **Example** – Suppose a study of speeding violations and drivers who use cell phones produced the following fictional data:
  - What is the probability that a randomly selected person is a cell phone user, given that they had no speeding violations in the last year?
- P(A|B) ≠ P(B|A).

Restricted sample space

In the last year	Speeding violation	No speeding violation	Total
Cell phone user	25	280	305
Not a cell phone user	45	405	450
Total	70	685	755

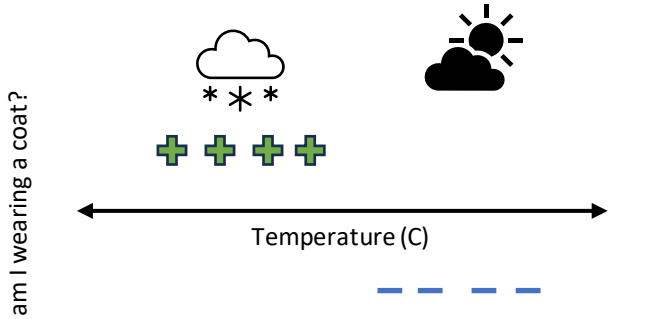
P(cell phone|no violations) =

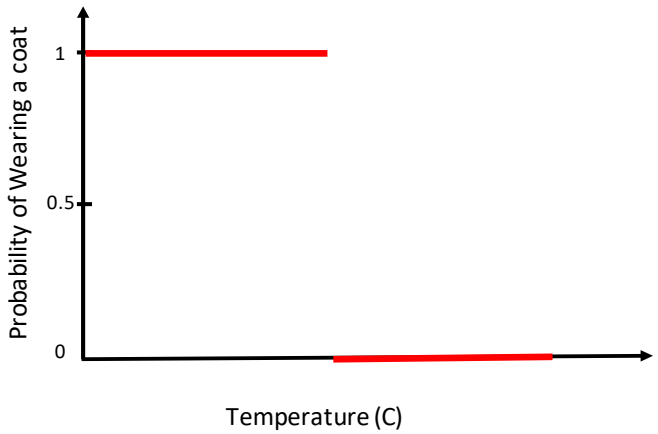
$$\frac{\text{number of cell phone users in restricted sample space}}{\text{total number in restricted sample spac}}$$

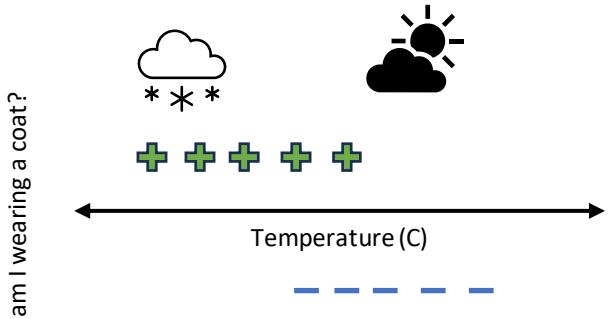
$$= \frac{280}{685} = 0.41$$

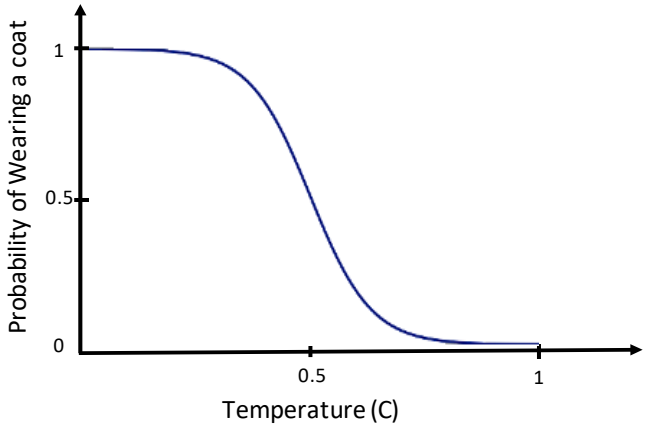
$$\Rightarrow P(\text{cell phone}|\text{no violations}) = \frac{P(\text{cell phone} \cap \text{no violation})}{P(\text{no violation})}$$

Capturing Uncertainty



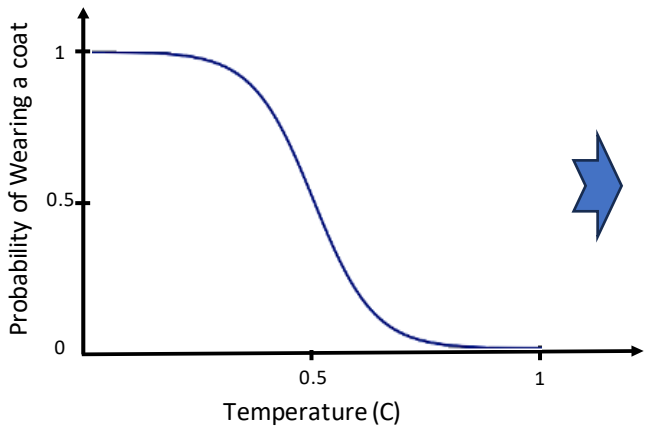






Source: Tamara Broderick

Capturing Uncertainty

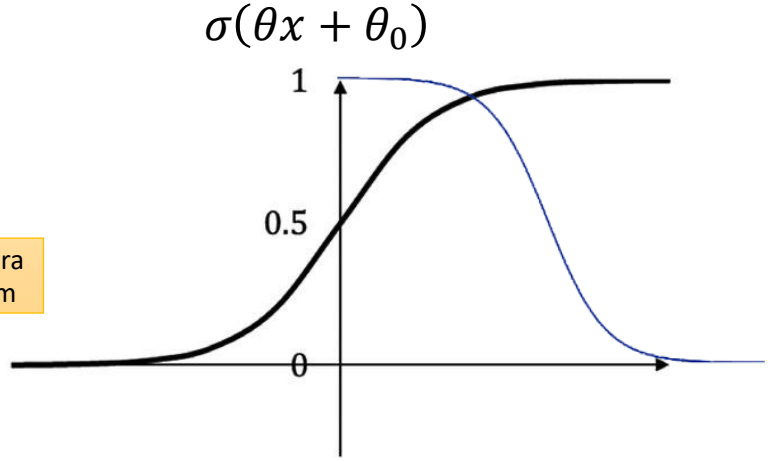


$$\sigma(\theta x + \theta_0) = \frac{1}{1 + e^{-(\theta x + \theta_0)}}$$

$$\hat{P}(y|x,\theta) = \frac{1}{1 + e^{-(\theta x + \theta_0)}}$$

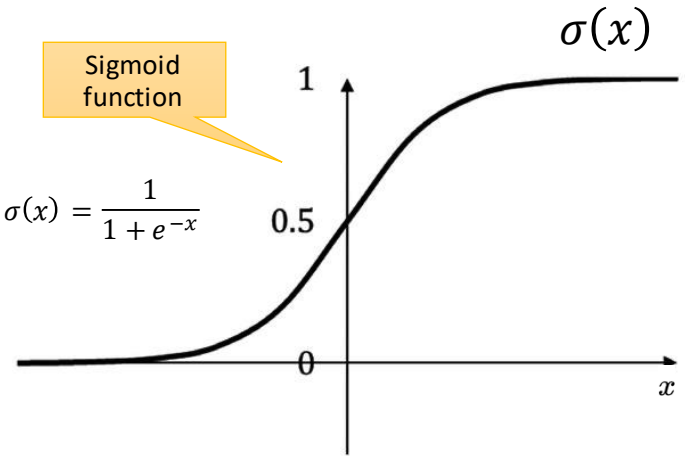
$$\hat{P}(y|x,\theta) = \frac{1}{1 + e^{-(\theta^T x + \theta_0)}}$$

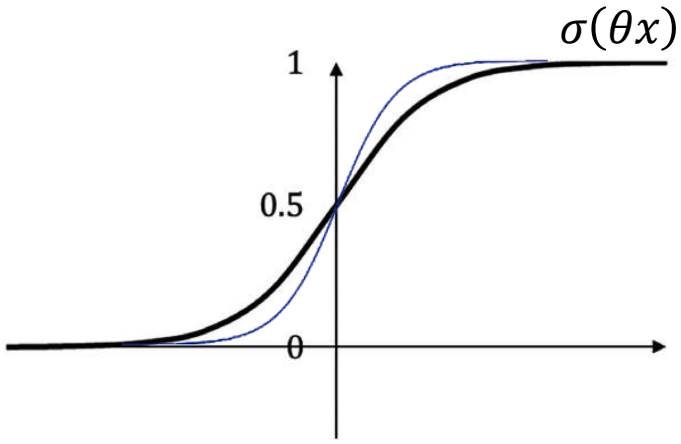
Genera l form

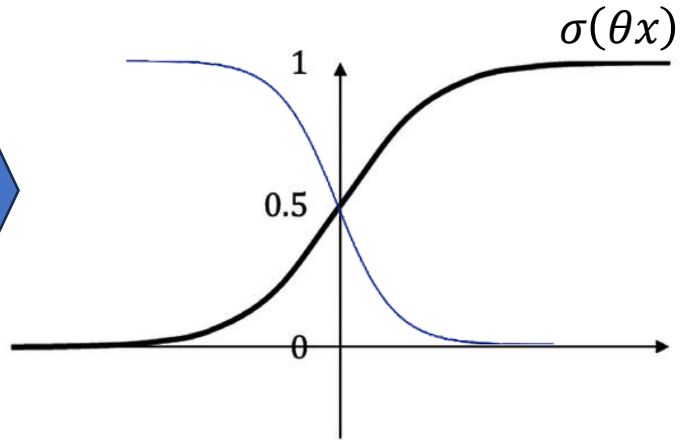


Sigmoid function

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







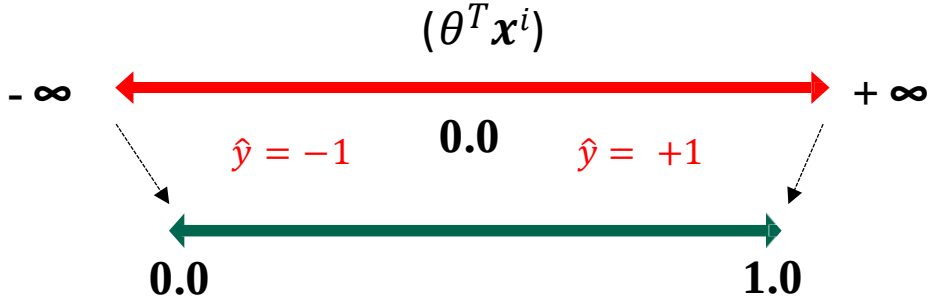
Using Probability in Classification

$$\hat{P}(y|x,\theta) = \frac{1}{1+e^{-(\theta^T x+\theta_0)}}$$

Simplify -> $\hat{P}(y|x,\theta) = \frac{1}{1+e^{-(\theta^T x)}}$

$(\theta^T x) \rightarrow +\infty$, $\hat{P}$= 1

$(\theta^T x) \rightarrow -\infty$, $\hat{P}$= 0

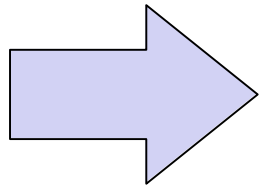


Logistic Regression

- $\hat{P}$ -> estimate of class probability
- Learning parameters of logistic regression:

Training Data: *m* Observations (x^i, y)

x^1	x^2	y
2	1	+1
1	1	-1
0	1	-1
2	0	+1
3	1	+1



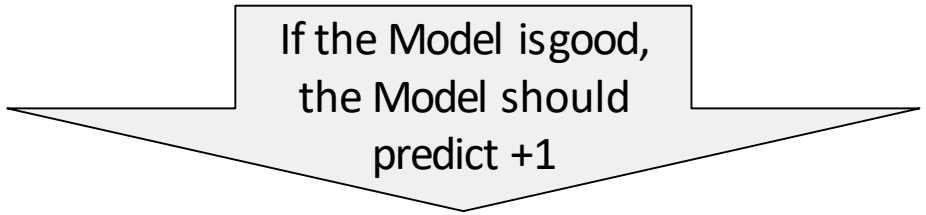
To learn $\hat{\theta}$, we need to define empirical classification error

Likelihood Function

- Empirical classification error => Likelihood function
- No $\hat{\theta}$ achieves perfect prediction (normally)
- Likelihood $l(\theta)$ => Measures the quality fit for the model with coefficients θ.

Likelihood Function (cont.)

x[1]	x[2]	y
3	1	+1



Pick θ to maximize

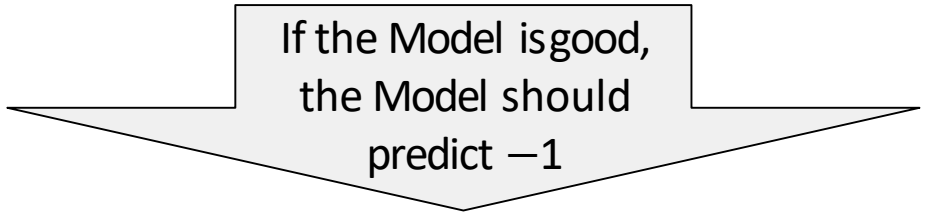
$$P(y = +1|\boldsymbol{x}, \theta) = P(y = +1|\boldsymbol{x}[1] = 3, \boldsymbol{x}[2] = 1, \theta)$$

- Your classifier is extremely good if:

$$P(y = +1|\boldsymbol{x}, \theta) \sim 1$$

Likelihood Function (cont.)

x[1]	x[2]	y
0	2	−1



Pick θ to maximize

$$P(y = -1|\boldsymbol{x}, \theta) = P(y = -1|\boldsymbol{x}[1] = 0, \boldsymbol{x}[2] = 2, \theta)$$

- Your classifier is extremely good if:

$$P(y = -1|\boldsymbol{x}, \theta) \sim 0$$

Likelihood Function (cont.)

x[1]	x[2]	y	Choose θ to maximize
2	1	+1	$P\ y = +1\ \boldsymbol{x}\ 1 = 2, \boldsymbol{x}\ 2 = 1, \theta$
1	1	-1	$P\ y = -1\ \boldsymbol{x}\ 1 = 1, \boldsymbol{x}\ 2 = 1, \theta$
0	1	-1	$P\ y = -1\ \boldsymbol{x}\ 1 = 0, \boldsymbol{x}\ 2 = 1, \theta$
2	0	+1	$P\ y = +1\ \boldsymbol{x}\ 1 = 2, \boldsymbol{x}\ 2 = 0, \theta$
3	1	+1	$P\ y = +1\ \boldsymbol{x}\ 1 = 3, \boldsymbol{x}\ 2 = 1, \theta$
2	2	-1	$P\ y = -1\ \boldsymbol{x}\ 1 = 2, \boldsymbol{x}\ 2 = 2, \theta$
4	4	-1	$P\ y = -1\ \boldsymbol{x}\ 1 = 4, \boldsymbol{x}\ 2 = 4, \theta$
…	..	…	…

How do we combine them into one single measure of Quality?

$$l(\theta) = P\left(y^{(1)} \middle| \boldsymbol{x}^{(1)}, \theta\right) . P\left(y^{(2)} \middle| \boldsymbol{x}^{(2)}, \theta\right) \ldots\ldots P\left(y^{(m)} \middle| \boldsymbol{x}^{(m)}, \theta\right)$$

$$l(\theta) = \prod_{i=1}^{m} P\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \theta\right)$$

Likelihood Function (cont.)

- In probability theory, multiplying probabilities is a fundamental concept used to find the joint probability of independent events occurring.
- Goal=> Choose the coefficients θ that maximizes the likelihood. This is done by multiplying probabilities.

$$l(\theta) = \prod_{i=1}^{m} P\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \theta\right)$$

- The Math is simplified by using log-likelihood: Taking the natural log

$$\ln\left(l(\theta)\right) = ll(\theta) = ln\left(\prod_{i=1}^{m} P\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \theta\right)\right)$$

Next

- Logistic Regression
- More Supervised Learning Algorithms


thank you