ENCS5341
Machine Learning and Data Science

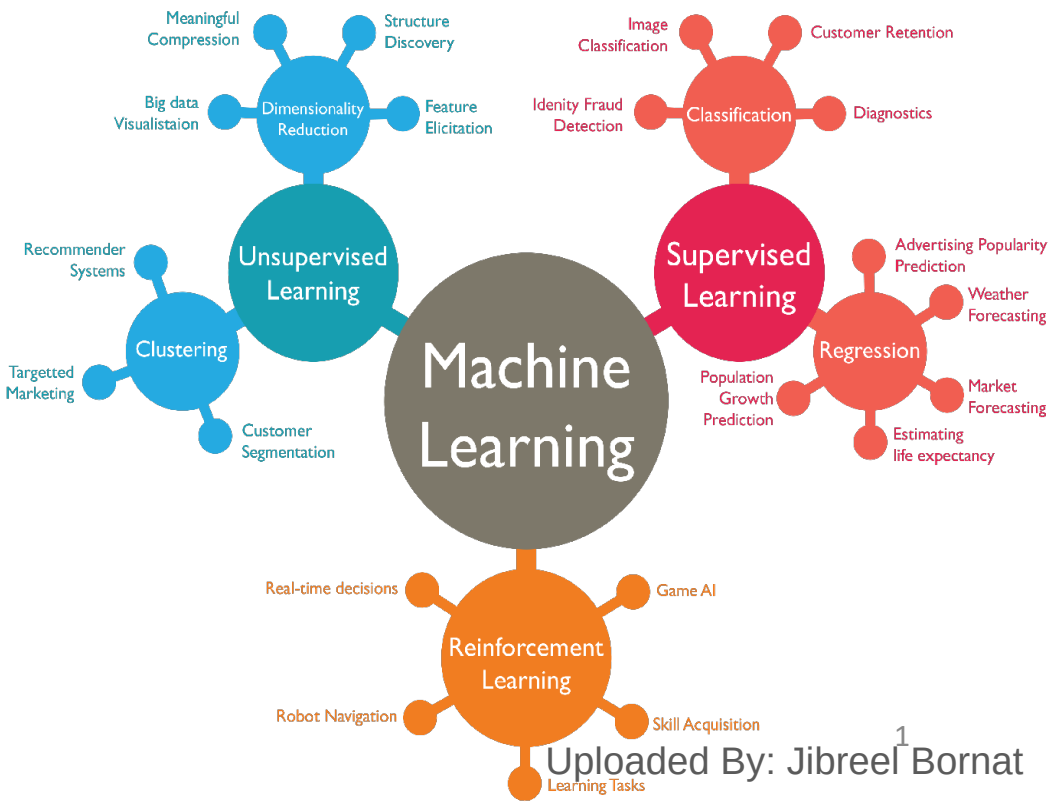
Classification

Introduction

- **Classification** is a supervised learning task where the goal is to predict a categorical (discrete) target label.
  Examples: spam detection, object recognition, … etc.

- If the target label has only two possible values (ex: spam/not spam), then the task is called **binary classification**.
  However, if there are more than two possible values (ex: object recognition) then the task is **multiclass classification**.

- For both binary classification and multiclass classification, each example has only one label. If the example can have more than one label (ex: book genre), then the task is called **multi-label classification**.



Classification Example

- Heart Data: These data contain a binary outcome HD for 303 patients who presented with chest pain. An outcome value of:
  - **Yes** indicates the presence of heart disease based on an angiographic test,
  - **No** means no heart disease.

response variable ***Y*** is Yes/No

Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No

k-NN for Classification

Instance-based learning (IBL)

Idea:

- *Store all training instances x along with their prediction value y:*

  x_1, y_1

  x_2, y_2

  …

- When asked to predict for an unseen instance
  - find stored instance "closest" to it: the "nearest neighbor"
  - and simply predict its stored class value

- *Do not produce an explicit generalization (lazy learning)*

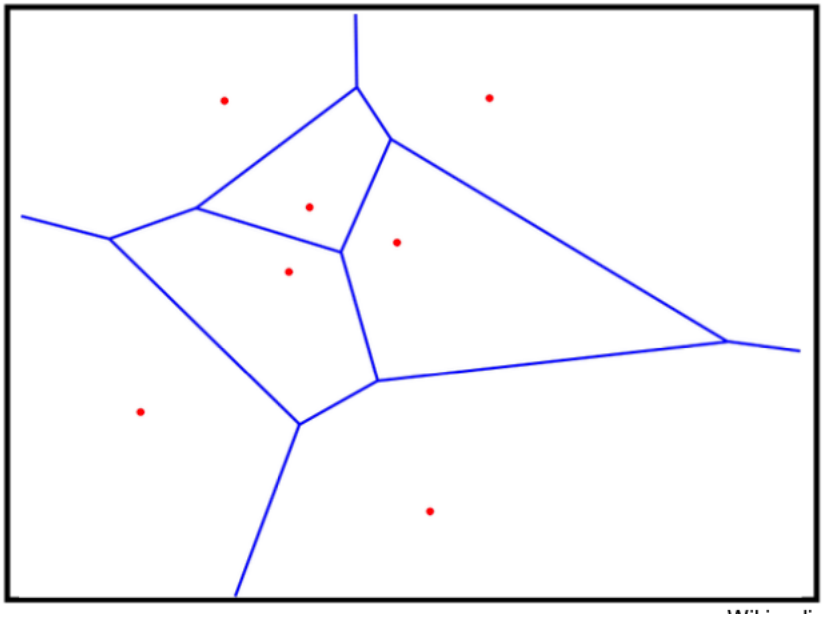
K-nearest neighbor learning:

- *Find k most similar instances and take a (weighted) majority vote*

Voronoi Diagram

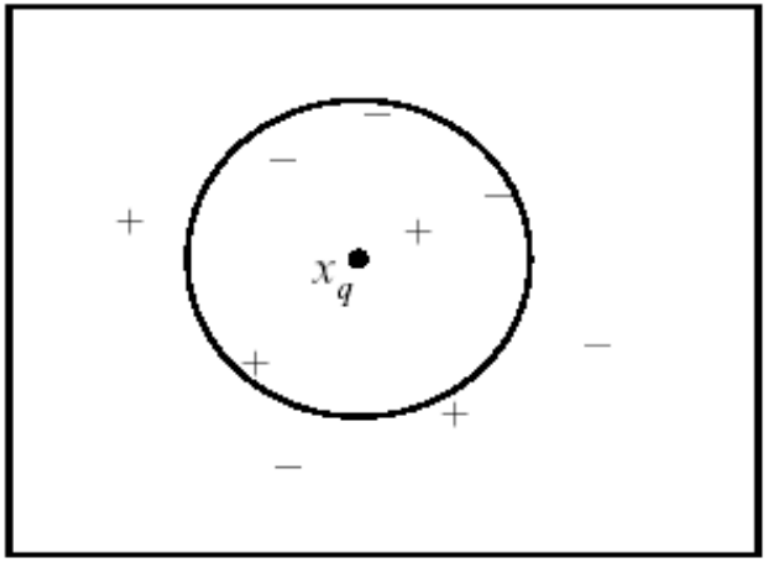
- The decision surface induced by 1-nearest neighbor is a combination of convex polyhedra surrounding each of the training examples (Voronoi Diagram)

- For every training example, the polyhedron indicates the set of query points whose classification will be completely determined by that training example



Example

- Given a set of positive (+) and negative (-) training examples and a new test example (query) x_q as shown in the figure



- What is the prediction of x_q in case of 1-nearest neighbor?

  (+)

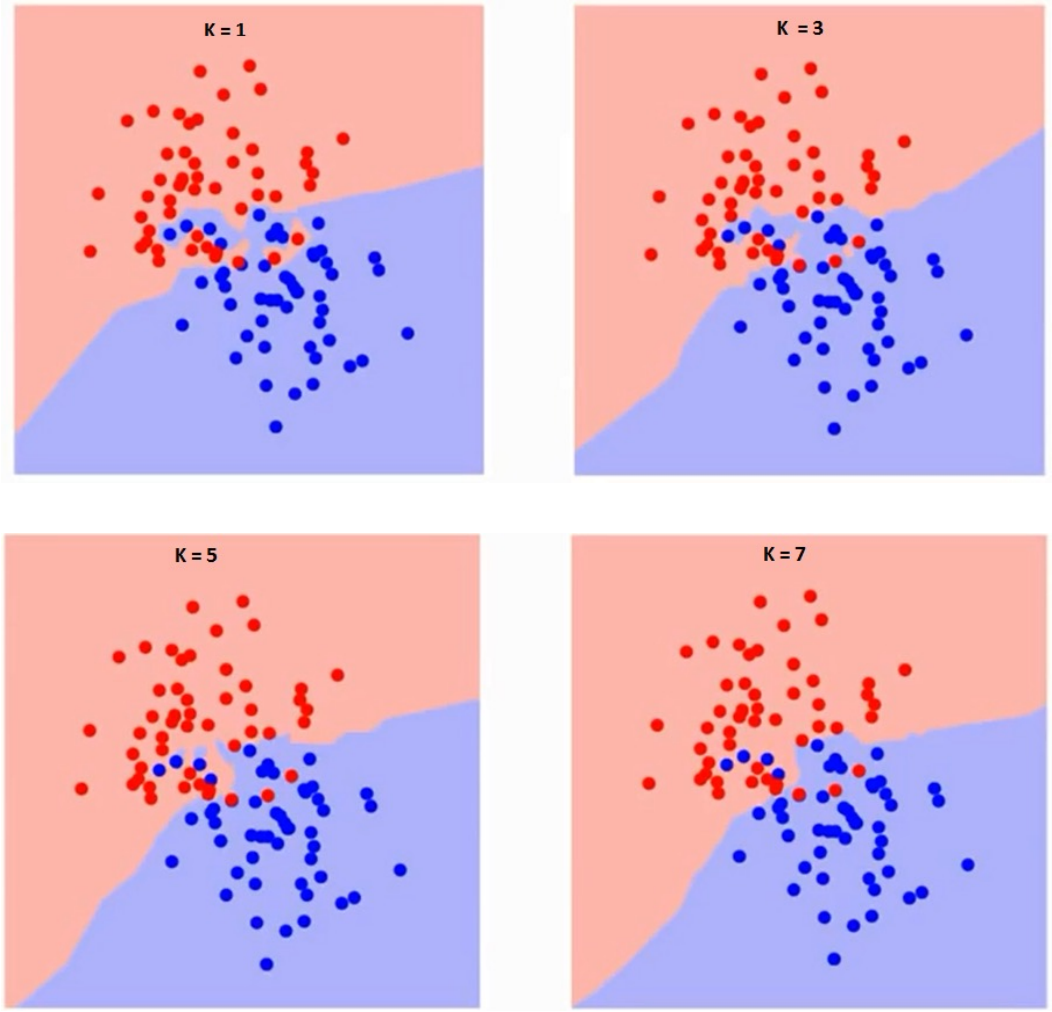
- What is the prediction of x_q in case of 5-nearest neighbor?

  (-)

Selecting k

- boundary becomes smoother with increasing value of k

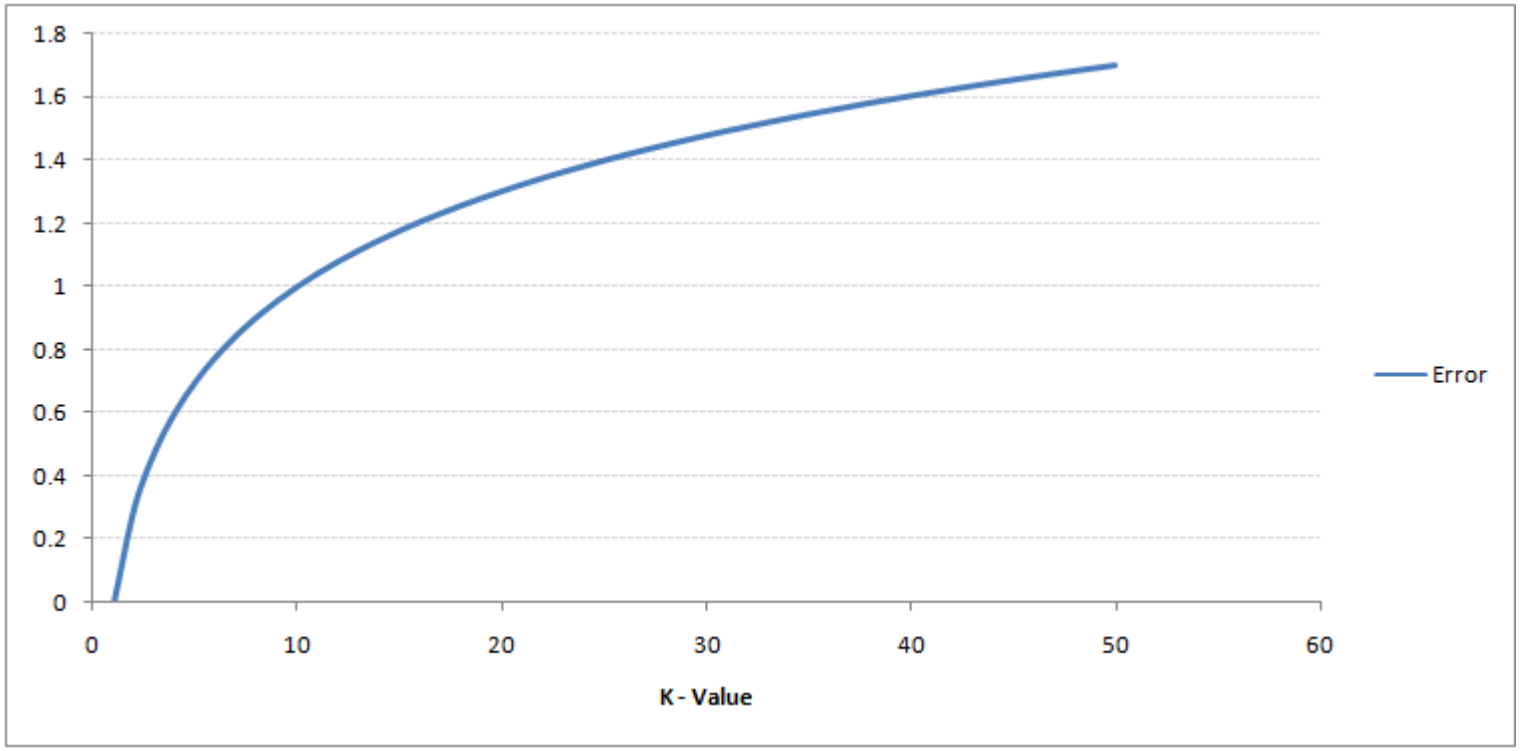
- With k increasing to infinity it finally becomes all blue or all red depending on the total majority.



Selecting k

Training error rate vs. value of k:

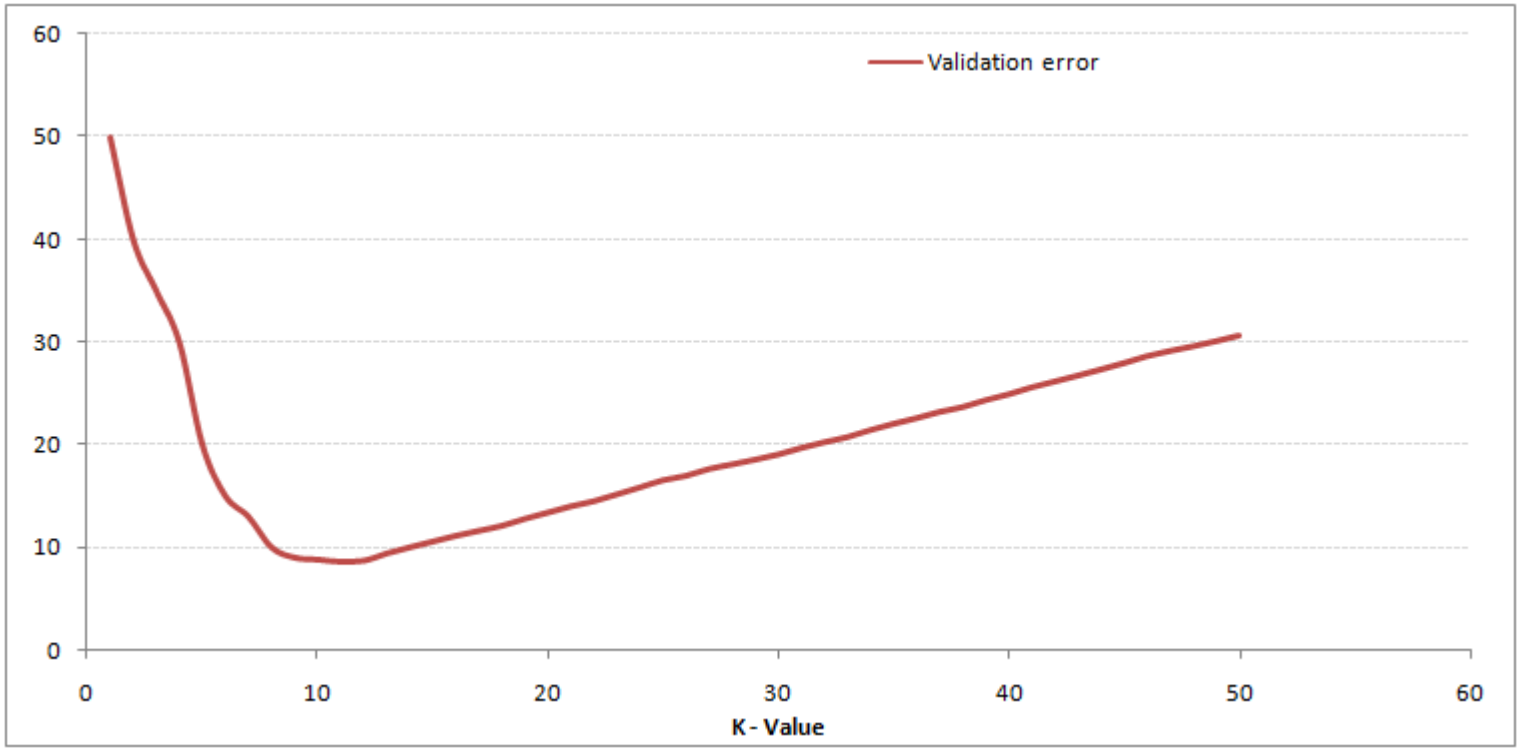
- The error rate at k=1 is always zero for the training sample. This is because the closest point to any training data point is itself.



Selecting k

Testing error rate vs. value of k:

- At k=1, we were overfitting the boundaries.
- Hence, error rate initially decreases and reaches a minima.
- After the minima point, it then increase with increasing k.



Majority Voting

- For instance x, let N = $\{n_1 \ldots n_k\}$ the k nearest neighbors of x, where $n_k = (x_k, y_k)$

- Let $w_k := 1 / \text{dist}(x, n_k)^2$, (k in {1...k})

- The Majority voting for discrete prediction problems:

$$h(x) := \text{argmax}_{y \in Y} \; | \{n_k \in N \mid n=(x_k,y)\} | \quad \text{(unweighted)}$$

$$h(x) := \text{argmax}_{y \in Y} \sum_{k=1..k,\, n_k=(x_k,y)} w_k \quad \text{(weighted)}$$

Distances for Real-Valued Spaces (R^m)

Assume we have objects x_r and x_s in R^m.

- l_1-Norm, city-block distance:

$$\sum_{j=1}^{m} \left| x_{rj} - x_{sj} \right|$$

- l_2-Norm, Euclidean distance:

$$\sqrt{\sum_{j=1}^{m} (x_{rj} - x_{sj})^2} = \sqrt{(x_r - x_s)^T (x_r - x_s)}$$

- l_2-Norm, Euclidean distance with weights

$$\sqrt{\sum_{j=1}^{m} \gamma_j (x_{rj} - x_{sj})^2} = \sqrt{(x_r - x_s)^T \Gamma (x_r - x_s)} \quad \text{with} \quad \Gamma = \begin{pmatrix} \gamma_1 & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & \gamma_m \end{pmatrix}$$

Distance Functions (Propositional Case)

For instances $a=\langle a_1,\ldots,a_m\rangle$ and $b=\langle b_1, \ldots, b_m\rangle$, define

$$dist(a,b) := \Sigma_{j=1..m}\, g_j\, dist(a_j,b_j)$$

where $dist(a_j,b_j) :=$

$$\begin{cases} |\, a_j - b_j \,| \,/\, range(att_j) & \text{if } att_i \text{ numeric} \\ d(a_j,b_j) & \text{if } att_i \text{ nominal} \end{cases}$$

- d for nominal attributes can be
  - d trivial: $d(a_j,b_j) := 0$ if $a_j=b_j$, else 1
  - d user-given based on domain properties
- Distance ideally should be a 1-bounded *metric* (and is as defined here). Then we can define *similarity*:

  $sim(a,b) = 1 - dist(a,b)$

Ideal: Distances should be metrics

- *distance* or *metric*: $\delta : X \times X \longrightarrow \mathbb{R}$ satisfying

$$\begin{aligned} \delta(x,y) &\geq 0 \text{ and } \delta(x,y) = 0 \text{ iff } x = y \\ \delta(x,y) &= \delta(y,x) \\ \delta(x,y) &\leq \delta(x,z) + \delta(z,y) \end{aligned}$$

- *trivial* metric: $\delta(x,y) = 0$ if $x = y$, otherwise 1
- *semi-metric*: the triangle inequality does not hold
- r-**bounded** metric, or semi-metric: $\delta(x,y) \leq r$

Scaling of attribute values and weighting of attributes

Scaling:

- If attributes have differing value ranges, the attributes with smaller absolute distances will be effectively “ignored”
    - E.g. one attribute is “income” (from 0 to 1000000€), another is age (from 0 to 125)
- So we must use proper scaling, e.g. to a range between 0 and 1

Weighting:

- K-NN will become confused by large numbers of (potentially irrelevant) attributes
- Must remove attributes that are not relevant (or try an error-based method to select them automatically)
- General problem! (curse of dimensionality)

Curse of Dimensionality

- As the number of dimensions grows, distances between points become more and more similar

- in particular, distances to nearest neighbor becomes similar to distance to most distant neighbor!

- Curse of dimensionality: nearest neighbor is easily mislead when high-dimensional X

Properties of kNN

+ No training time - just reads and stores instances
+ Generally very good accuracy due to locality of generalization
+ Handles numeric problems and categorical problems equally well
+ Deals well with missing data, noise, outliers

– Slow at query time

➔ Use prototype or selected example sets only

– Requires *k* to be set

➔ Use internal optimization on held-out part of the training set

– Can become confused by irrelevant or redundant attributes

➔ Use attribute weight learning (transformation of feature space)

Parametric Modeling: Why not Linear Regression?

Simple Classification Example

- Given a dataset:

$$\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_N, y_N)\}$$

where the y are categorical (sometimes referred to as qualitative), we would like to be able to predict which category y takes on given x.

- A categorical variable y could be encoded to be quantitative. For example, if y represents concentration of BZU undergrads, then y could take on the values:

$$y = \begin{cases} 1 & if \text{ Computer Science (CS)} \\ 2 & if \text{ Statistics} \\ 3 & \text{otherwise} \end{cases} .$$

Linear regression **does not work well**, or is not appropriate at all, in this setting.

Simple Classification Example (cont.)

- A linear regression could be used to predict y from x. What would be wrong with such a model?

- The model would imply a specific ordering of the outcome, and would treat a one-unit change in y equivalent. The jump from y = 1 to y = 2 (CS to Statistics) should not be interpreted as the same as a jump from y = 2 to y = 3 (Statistics to everyone else).

- Similarly, the response variable could be reordered such that y = 1 represents Statistics and y = 2 represents CS, and then the model estimates and predictions would be fundamentally different.

- If the categorical response variable was ***ordinal*** (had a natural ordering, like class year: Freshman, Sophomore, etc.), then a linear regression model would make some sense but is still not ideal.

Even Simpler Classification Problem: Binary Response

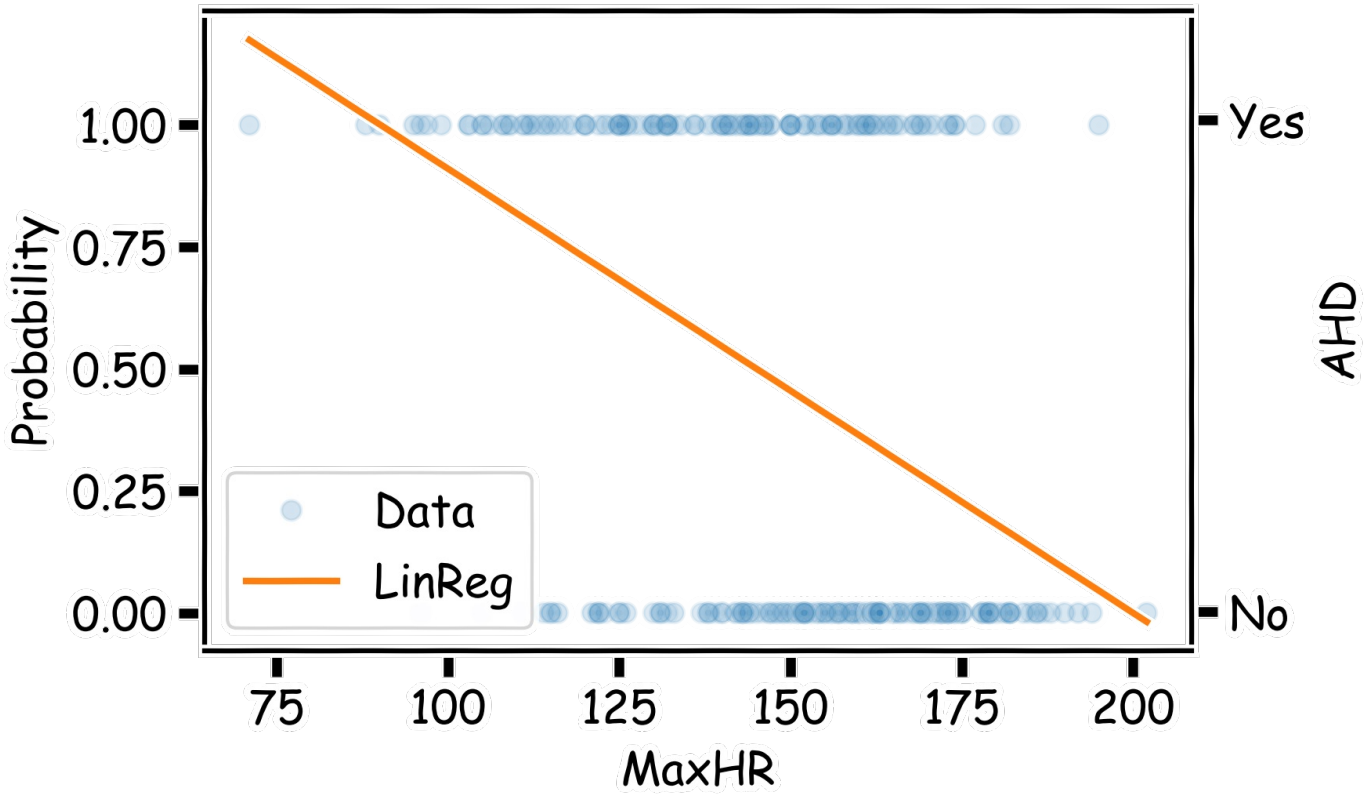
- The simplest form of classification is when the response variable y has only two categories, and then an ordering of the categories is natural. For our example, a patient in the ICU could be categorized as having [atherosclerotic] heart disease (AHD) or not (note, the y=0 category is a "catch-all" so it would involve those patients with lots of other diseases or diagnoses):

$$y = \begin{cases} 1 \text{ if patient has heart disease} \\ 0 \text{ otherwise.} \end{cases}$$

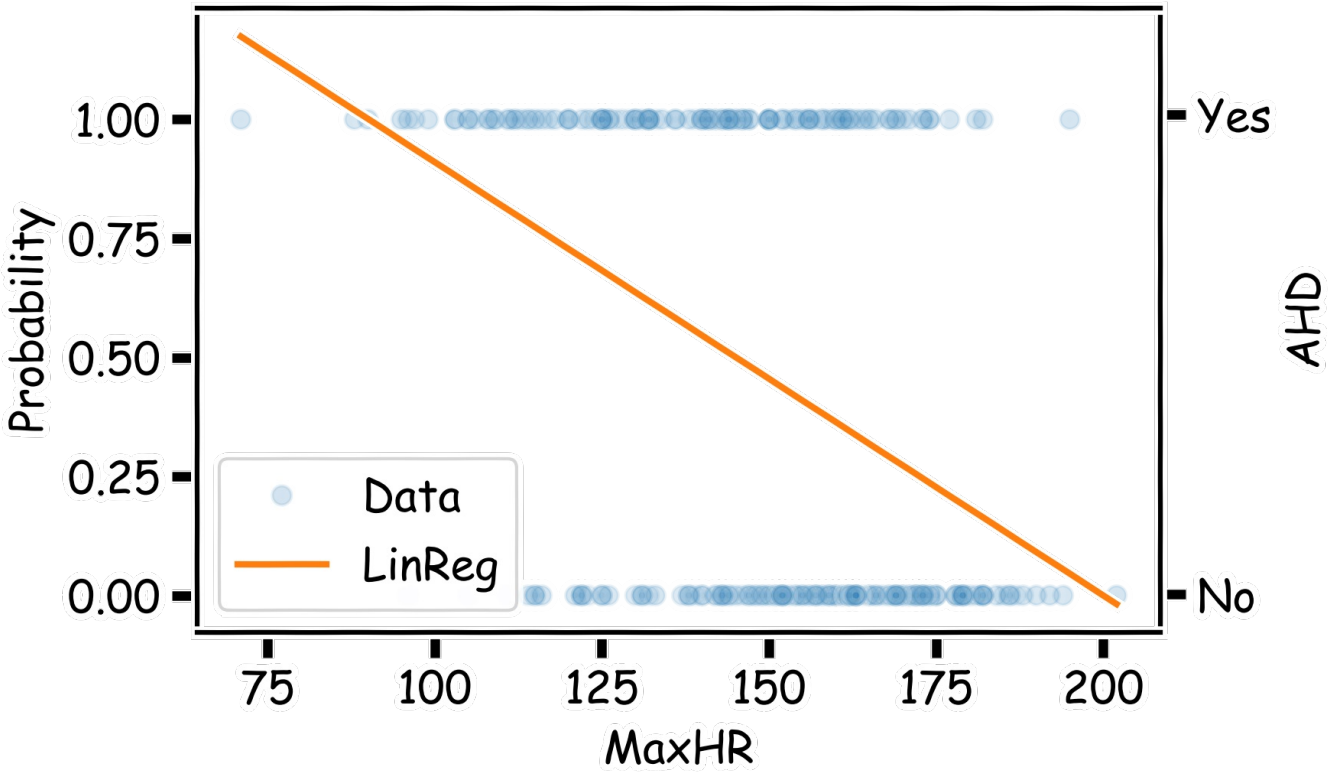
- Linear regression could be used to predict y directly from a set of covariates (like sex, age, resting HR, etc.), and if $\hat{y} \geq 0.5$, we could predict the patient to have AHD and predict not to have heart disease if $\hat{y} < 0.5$.

Even Simpler Classification Problem: Binary Response (cont)

- What could go wrong with this linear regression model?



Even Simpler Classification Problem: Binary Response (cont)

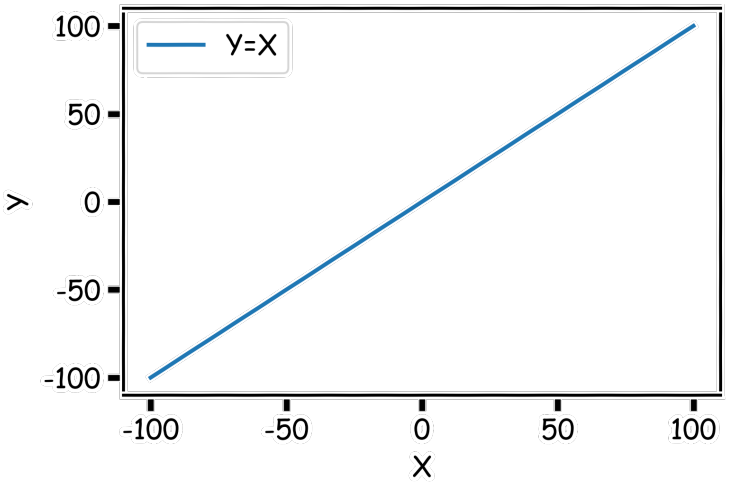


- The main issue is you could get non-sensical values for y. Since this is modeling $P(y{=}1)$, values for $\hat{y}$ below 0 and above 1 would be at odds with the natural measure for y. Linear regression can lead to this issue.

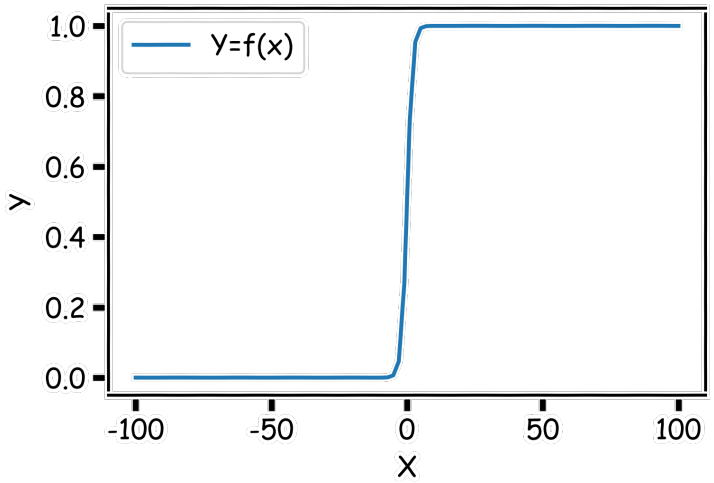
Binary Response & Logistic Regression

Quiz

Think of a function that would do this for us



$$Y = f(x)$$



Logistic Regression

- Logistic Regression addresses the problem of estimating a probability, $P(y=1)$, to be outside the range of [0,1].

- The logistic regression model uses a function, called the ***logistic*** function (or ***sigmoid*** function), to model $P(y=1)$:

$$P(Y=1) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}$$

Logistic Regression Model

- The optimal decisions are based on the posterior class probabilities $P(y|\mathbf{x})$. For binary classification problems, we can write these decisions as

$$y = 1 \text{ if } \quad \log \frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})} > 0$$

  and $y = 0$ otherwise.

- We generally don't know $P(y|\mathbf{x})$ but we can parameterize the possible decisions according to

$$\log \frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})} = f(\mathbf{x};\mathbf{w}) = w_0 + \mathbf{x}^T\mathbf{w}_1$$

Logistic Regression Model

- Our log-odds model

$$\log \frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})} = w_0 + \mathbf{x}^T \mathbf{w}_1$$

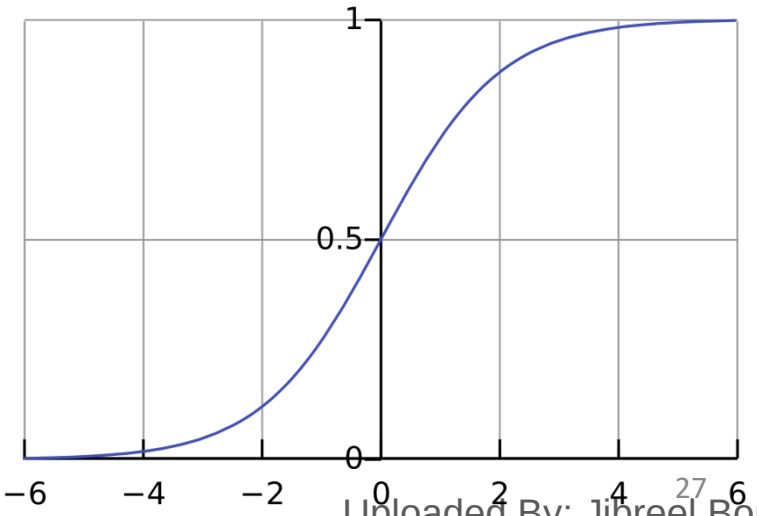
gives rise to a specific form for the conditional probability over the labels (the logistic model):

$$P(y=1|\mathbf{x},\mathbf{w}) = g\left(w_0 + \mathbf{x}^T \mathbf{w}_1\right)$$

- Want $0 \leq h_\theta(x) \leq 1$

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}},$$

$$g(z) = \frac{1}{1+e^{-z}}$$



Logistic Regression - Decision Boundary

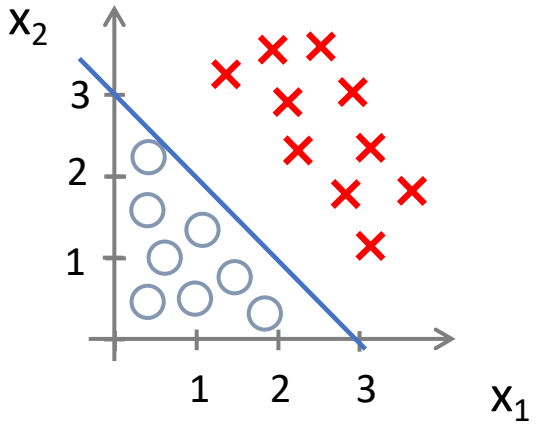
Why Sigmoid?

- Might seem more "natural" than others
- Activation function that transform linear inputs to nonlinear outputs.
- Bound output to between 0 and 1 so that it can be interpreted as a probability.
- Make computation easier than arbitrary activation functions.
- differentiable

$$
\begin{aligned}
g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= g(z)(1-g(z)).
\end{aligned}
$$

Logistic Regression- Decision Boundary

- Linear decision boundaries

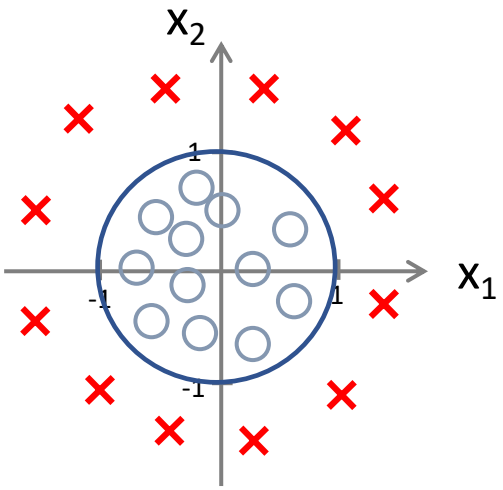


$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

Logistic Regression - Decision Boundary

- Non-linear decision boundaries



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$

Fitting logistic regression models

- As with the linear regression models we can fit the logistic models using the maximum (conditional) log-likelihood criterion

$$l(D; \mathbf{w}) = \sum_{i=1}^{n} \log P(y_i | \mathbf{x}_i, \mathbf{w})$$

where

$$P(y = 1 | \mathbf{x}, \mathbf{w}) = g\left(w_0 + \mathbf{x}^T \mathbf{w}_1 \right)$$

- A number of optimization techniques are available for finding the maximizing parameters
  - Gradient descent: Neural Networks
  - Newton's method – Fisher scoring: Features Selection.

Fitting logistic regression models ML Estimator

- Let us assume that

$$\begin{aligned} P(y = 1 \mid x; \theta) &= h_\theta(x) \\ P(y = 0 \mid x; \theta) &= 1 - h_\theta(x) \end{aligned}$$

- Note that this can be written more compactly as

$$p(y \mid x; \theta) = (h_\theta(x))^y \, (1 - h_\theta(x))^{1-y}$$

- Assuming that the **m** training examples were generated independently, we can then write down the likelihood of the parameters as

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X; \theta) \\ &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

Fitting logistic regression models using gradient ascent

- As with Linear Regression, it will be easier to maximize the log likelihood:

$$\begin{aligned}\ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))\end{aligned}$$

- we can use gradient ascent. Written in vectorial notation, our updates will therefore be given by

$$\theta := \theta + \alpha \nabla_\theta \ell(\theta)$$

where

$$\begin{aligned}\frac{\partial}{\partial \theta_j}\ell(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial \theta_j} g(\theta^T x) \\ &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right) g(\theta^T x)(1 - g(\theta^T x)\frac{\partial}{\partial \theta_j}\theta^T x \\ &= \left(y(1 - g(\theta^T x)) - (1-y)g(\theta^T x)\right) x_j \\ &= (y - h_\theta(x))\, x_j\end{aligned}$$

- This therefore gives us the stochastic gradient ascent rule

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

Gradient Descent for Logistic Regression

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

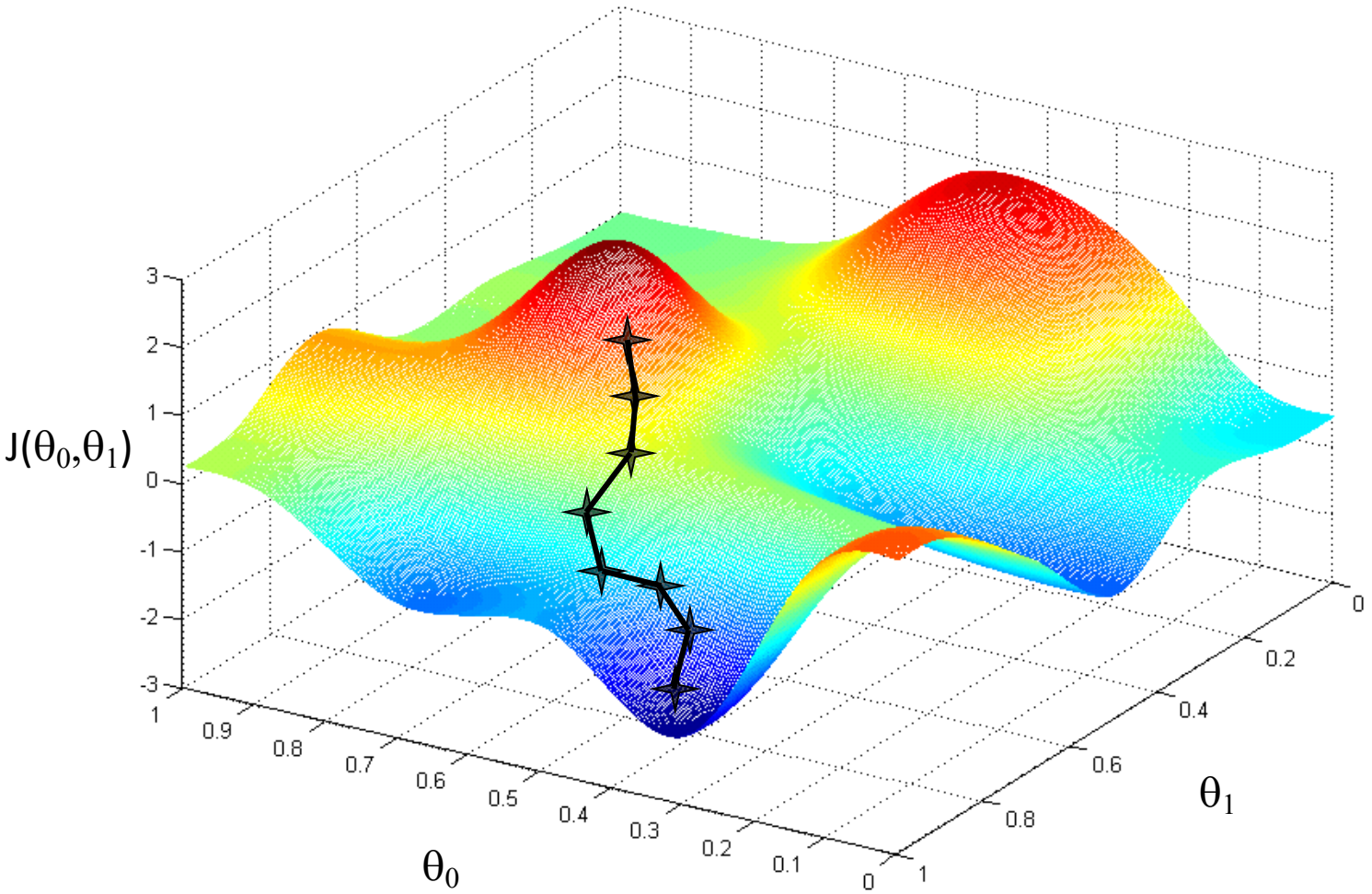
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

(simultaneously update all)

$\}$

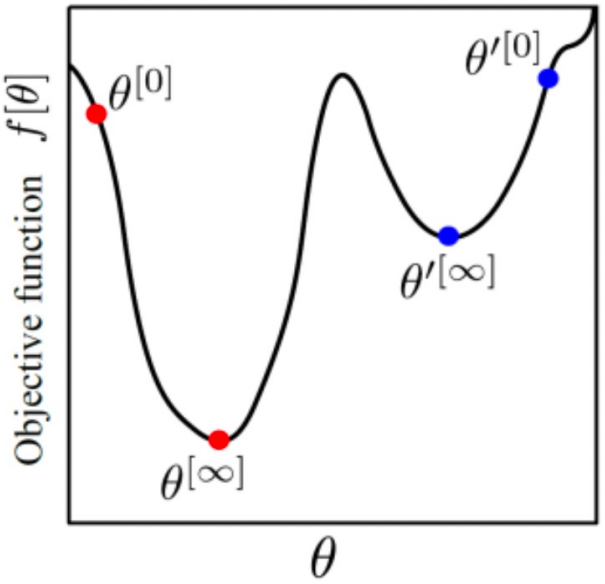
Start from some random initial point and move in the ‘opposite gradient direction' in order to decrease the cost function. We move step by step until there is no decrease in the cost function.

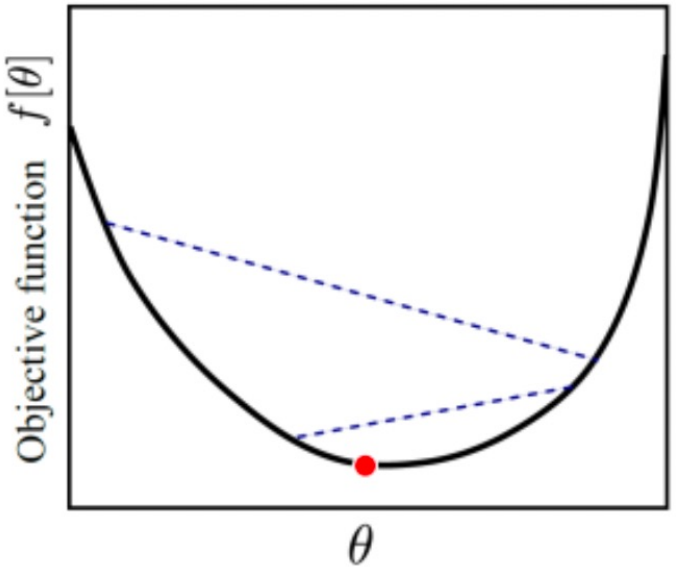
Gradient descent algorithm



Local Minima

- Gradient descent is a local search method for minimizing a function.
- This makes gradient descent prone to getting stuck in local minima, rather than reaching the global minimum.
- If the function is convex, then it has only a single minimum.





Multiclass Classification using Logistic Regression

- one-vs-all (one-vs-rest):
  - Train one classifier F_m for each class m
    - All examples with label m are positive examples.
    - All other examples are negative examples
  - Classify by finding maximal response

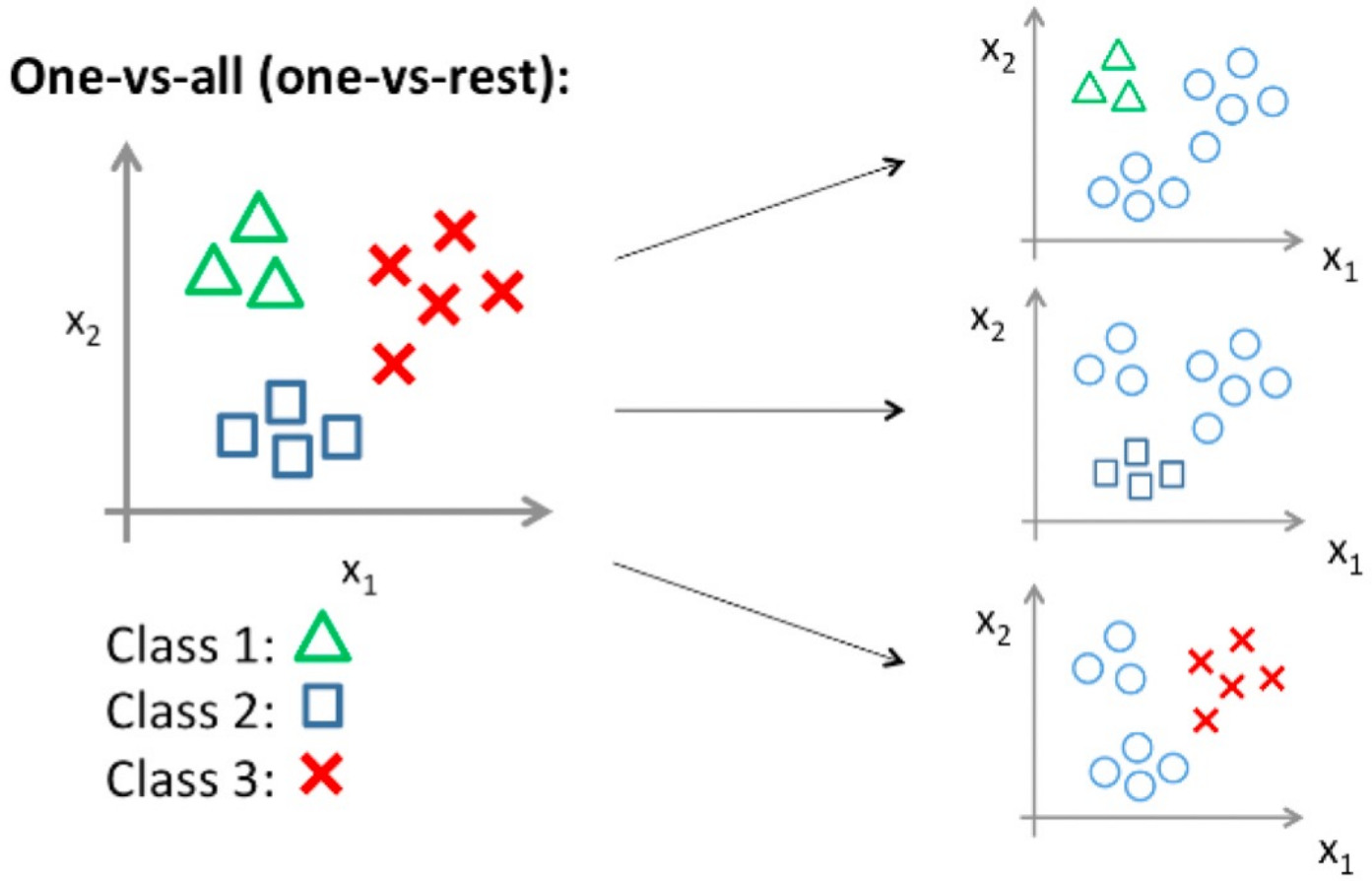
$$f(x) = \underset{m=1,\ldots,M}{\operatorname{argmax}} F_m(x)$$

- all-vs-all
  - Train a classifier for each pair of classes

$$F_{ij} \text{ for each } 1 \leq i < j \leq M, \text{ total } \frac{m(m-1)}{2}$$

  - Classify by voting

Multiclass Classification using Logistic Regression



Basic idea – Transfer multi-class classification into binary classification problem

Multiclass Classification using Softmax Regression

- Softmax regression is a generalization of logistic regression for multiclass classification problems.
- For example, rather than classifying email into the two classes, we might want to classify it into three classes, such as spam, personal mail, and work-related mail.

- Data $\mathbf{x} \in \mathbb{R}^d$
- Labels $y \in \{1, 2, \ldots, k\}$
- Model parametrized by $\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_k} \in \mathbb{R}^d$

- $P(y = j \mid x) = \frac{e^{\langle x, w_j \rangle}}{\sum_{i=1}^{k} e^{\langle x, w_i \rangle}}$

- Prediction: given a point x, predict label $\text{argmax}_j \, P(y = j \mid x)$

Application: Sentiment Analysis

- Problem: Given a sentence predict if the sentiment is positive or negative.
- Also known as opinion mining or emotion AI.
- Data set:
  - Sentences from reviews on Amazon, Yelp, IMDB.
  - Each labeled as positive or negative.
  - 2500 training sentences, 500 test sentences

Sentiment Analysis

Examples:

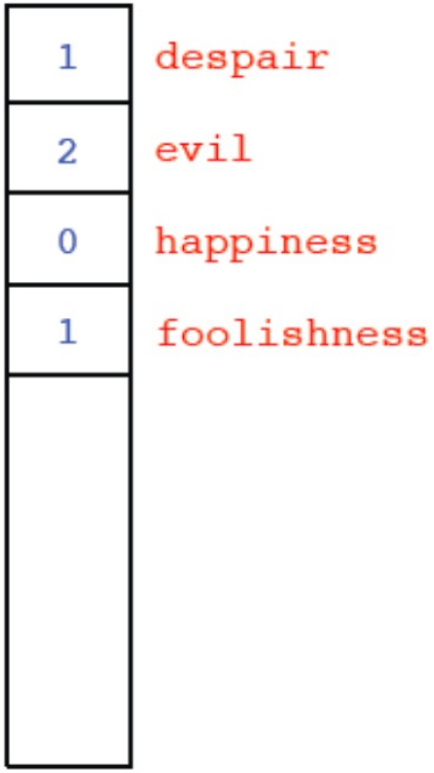
- Needless to say, I wasted my money.
- He was very impressed when going from the original battery to the extended battery.
- Decent volume.
- Will order from them again!

Handling text data

- How do we use sentences as features?

- One Solution: Bag-of-words: vectorial representation of text sentences.

It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way – in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.



Bag-of-words

- Fix V = Some vocabulary

- Treat each sentence (or document) as a vector of length |V|:

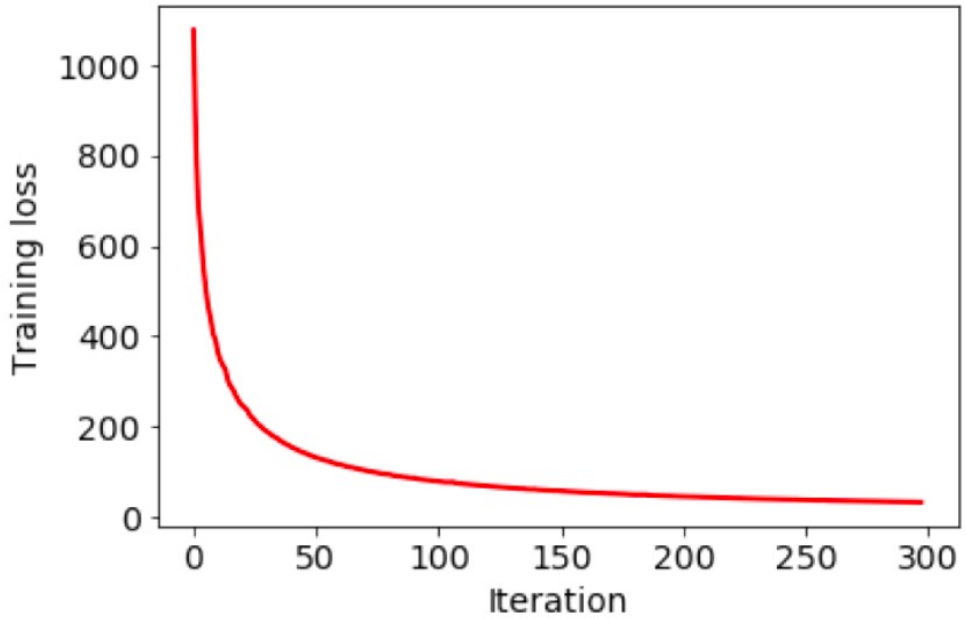
$$x = (x_1, x_2, \ldots, x_{|V|})$$

where x_i = # of times the i-th word appears in the sentence

- Then we normalize each vector by the sum of x_i

A logistic regression approach

- Code positive as class 1 and negative as class 0.
- $P(y = 1 \mid x) = \frac{1}{1 + e^{-<x,w>}}$
- Given training data, minimize the loss/cost



Interpreting the model

Words with the most positive coefficients:

- 'sturdy', 'able', 'happy', 'disappoint', 'perfectly', 'remarkable', 'animation', 'recommendation', 'best', 'funny', 'restaurant', 'job', 'overly', 'cute', 'good', 'rocks', 'believable', 'brilliant', 'prompt', 'interesting', 'skimp', 'comfortable', 'amazing', 'tasty', 'wonderful', 'excellent', 'pleased', 'beautiful', 'fantastic'

Words with the most negative coefficients:

- 'disappointment', 'poor', 'not', 'doesn', 'worst', 'average', 'garbage', 'bit', 'looking', 'avoid', 'roasted', 'broke', 'starter', 'disappointing', 'dont', 'waste', 'why', 'slow', 'none', 'directing', 'stupid', 'lazy'

Classification Evaluation Metrics

Classification metrics

- For classification, we usually represent the predictions by a confusion matrix, from which we derive all metrics.
- Confusion Matrix
  - C by C matrix (C is the number of classes).
  - Rows correspond to true classes, columns to predicted classes.
  - Count how often samples belonging to class c are classified as c or any other class.
  - For binary classification, we label these true negative (TN), true positive (TP), false negative (FN), false positive (FP).

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Classification Evaluation Metrics

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

$$TPR(recall\ or\ sensitivity) = \frac{TP}{TP + FN}$$

$$TNR\ (specificity) = \frac{TN}{TN + FP}$$

$$FPR = \frac{FP}{TN + FP}$$

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

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

		Predicted: Negative	Predicted: Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

Classification Evaluation Metrics

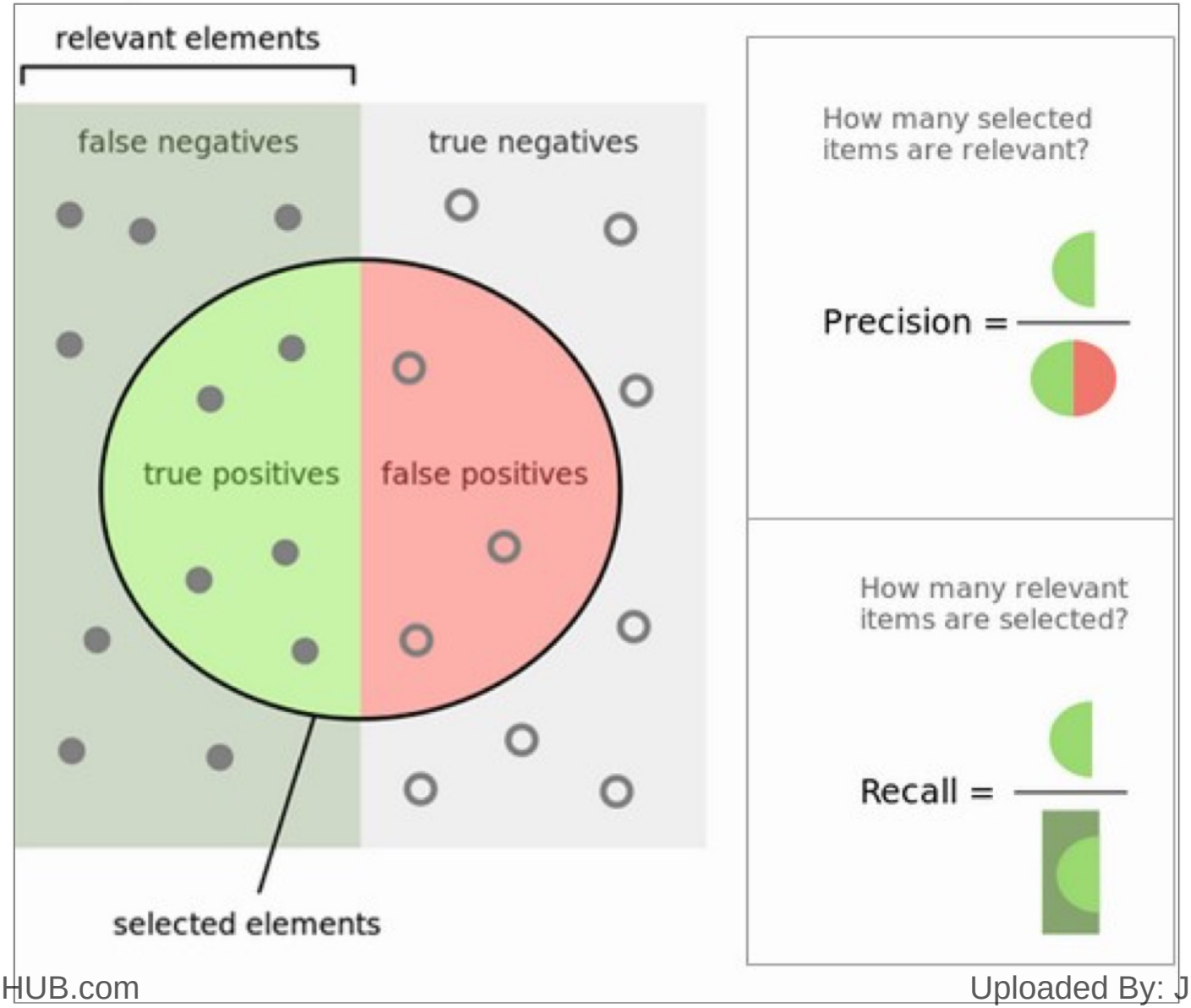
Remarks

- Accuracy is not the best measure when the classes are highly skewed, i.e., number of samples of one class is significantly higher than the number of samples of other class.

- Precision is used when the goal is to limit FPs. E.g. clinical trials (you only want to test drugs that really work), search engine (you want to avoid bad search results).

- Recall is used when the goal is to limit FNs. E.g. cancer diagnosis (you don't want to miss a serious disease), search engine (you don't want to miss important hits).

Precision vs. Recall



F1-score

- Trade off precision and recall

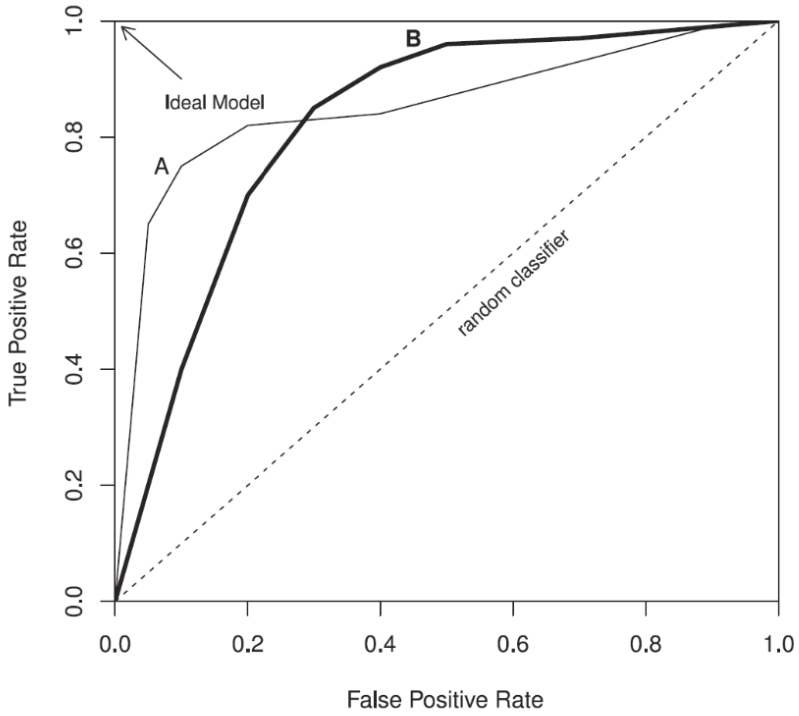
$$\mathrm{F1} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

Receiver operating characteristics (ROC)

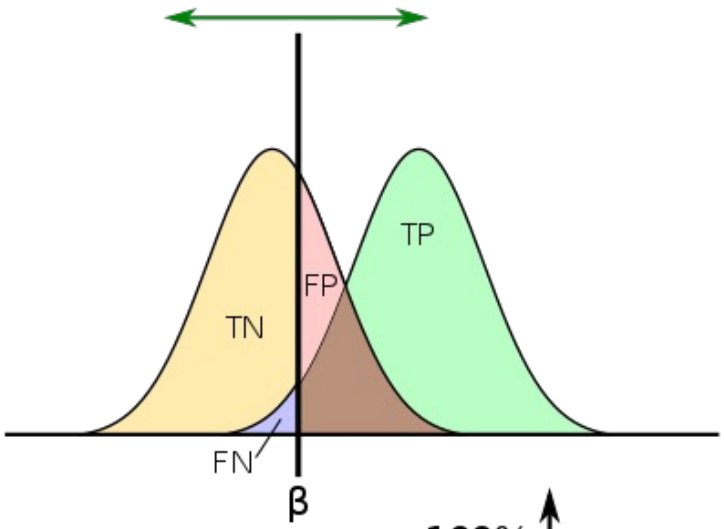
- Trade off true positive rate with false positive rate

$$TPR = \frac{TP}{TP+FN} \qquad FPR = \frac{FP}{FP+TN}$$

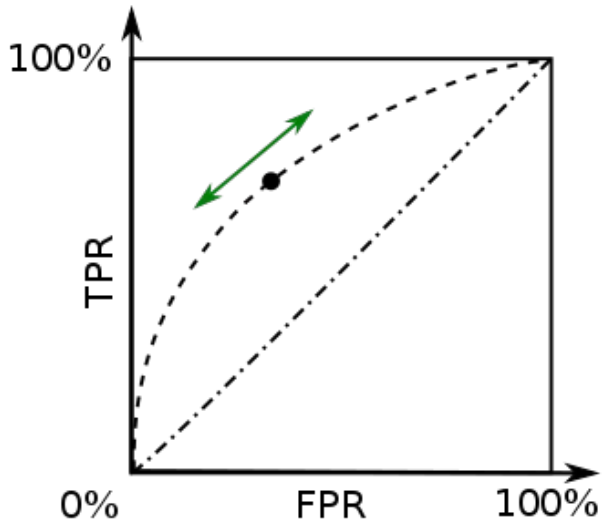
- Plotting TPR against FPR for all possible thresholds yields a Receiver Operating Characteristics curve.
  - Change the thresholds until you find a sweet spot in the TPR-FPR trade-off.
  - Lower thresholds yield higher TPR (recall), higher FPR, and vice versa.
- The area under the ROC curve gives the best overall model.



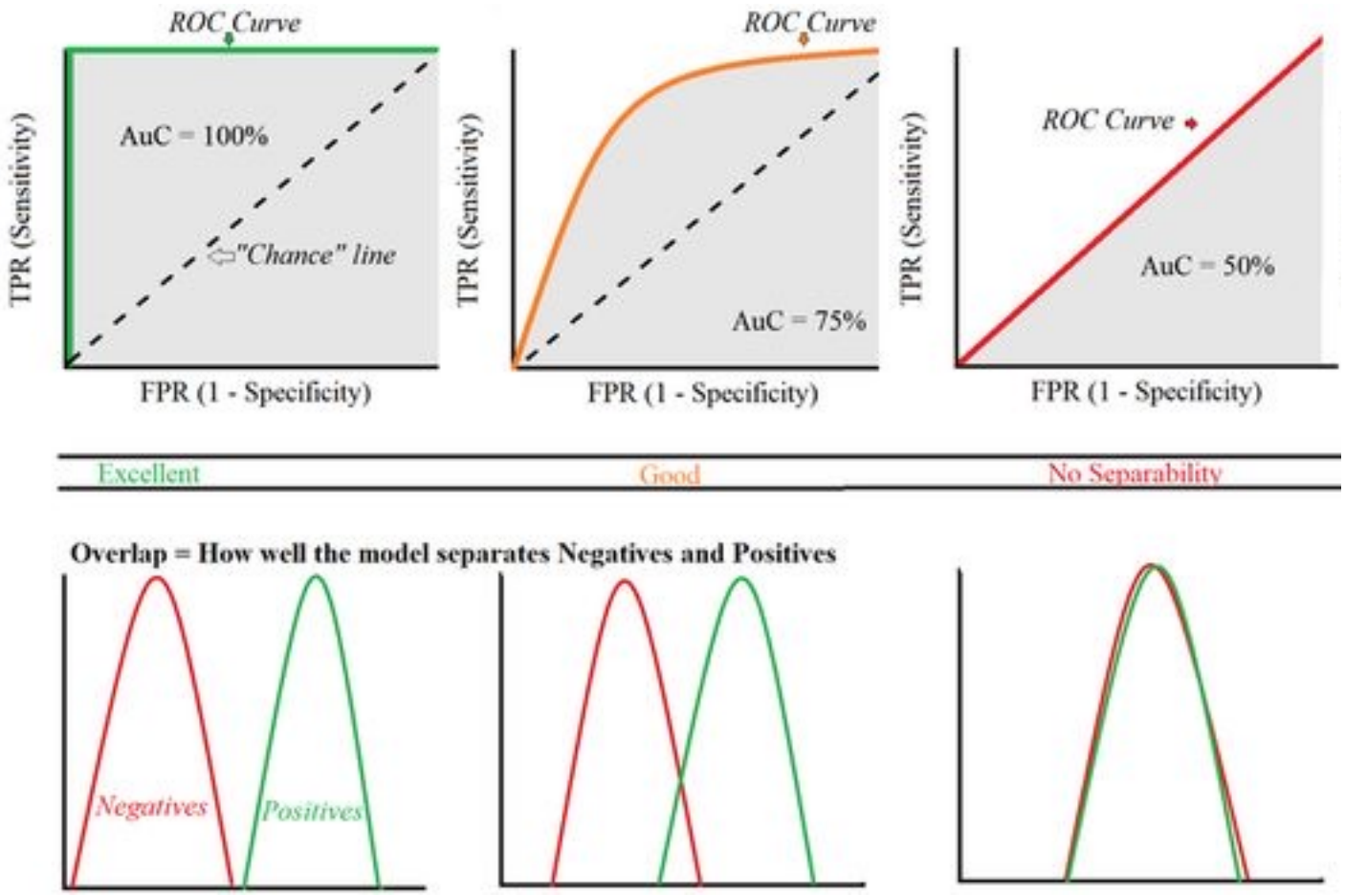
Receiver operating characteristics (ROC)



TP	FP
FN	TN



Receiver operating characteristics (ROC)

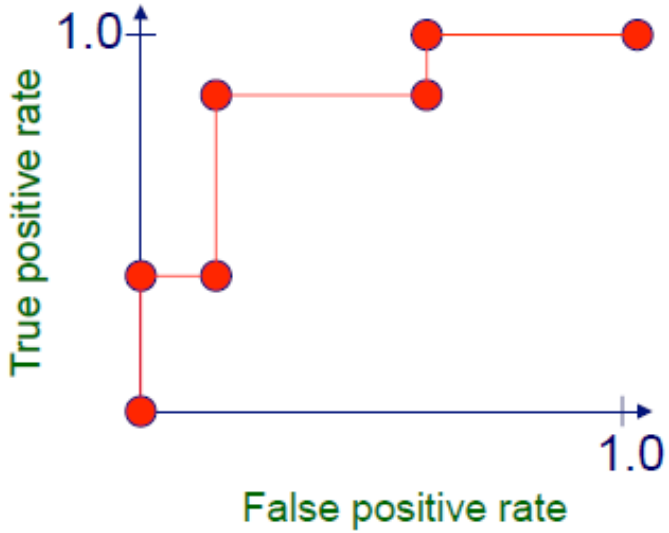


How to Construct an ROC curve

- Use classifier that produces posterior probability for each test instance P(+|A)
- Sort the instances according to P(+|A) in decreasing order
- Apply threshold at each unique value of P(+|A)
- Count the number of TP, FP, TN, FN at each threshold
- TP rate, TPR = TP/(TP+FN)
- FP rate, FPR = FP/(FP + TN)

How to Construct an ROC curve

instance	confidence positive		correct class
Ex 9	.99		+
Ex 7	.98	TPR= 2/5, FPR= 0/5	+
Ex 1	.72	TPR= 2/5, FPR= 1/5	-
Ex 2	.70		+
Ex 6	.65	TPR= 4/5, FPR= 1/5	+
Ex 10	.51		-
Ex 3	.39	TPR= 4/5, FPR= 3/5	-
Ex 5	.24	TPR= 5/5, FPR= 3/5	+
Ex 4	.11		-
Ex 8	.01	TPR= 5/5, FPR= 5/5	-



ROC Curve Interpretation

- Test A and B have nearly equal area but cross each other. Test A performs better than test B where high sensitivity is required, and test B performed better than A when high specificity is needed.