Unsupervised Learning:

Unsupervised Learning

- Supervised learning used labeled data pairs $(\mathbf{x}, \mathrm{y})$ to learn a function $f : \mathrm{X} \rightarrow \mathrm{Y}$
  - But, what if we don't have labels?

- No labels = **unsupervised learning**

- Learning algorithm must find patterns from features alone
- Discover hidden patterns, exploratory analysis
- Examples:
  - K-means clustering , Gaussian Mixture Models
  - Dimensionality Reduction: Transform an initial feature representation into a more concise representation

Clustering

Clustering: group together similar points and represent them with a single token

Key Challenges:

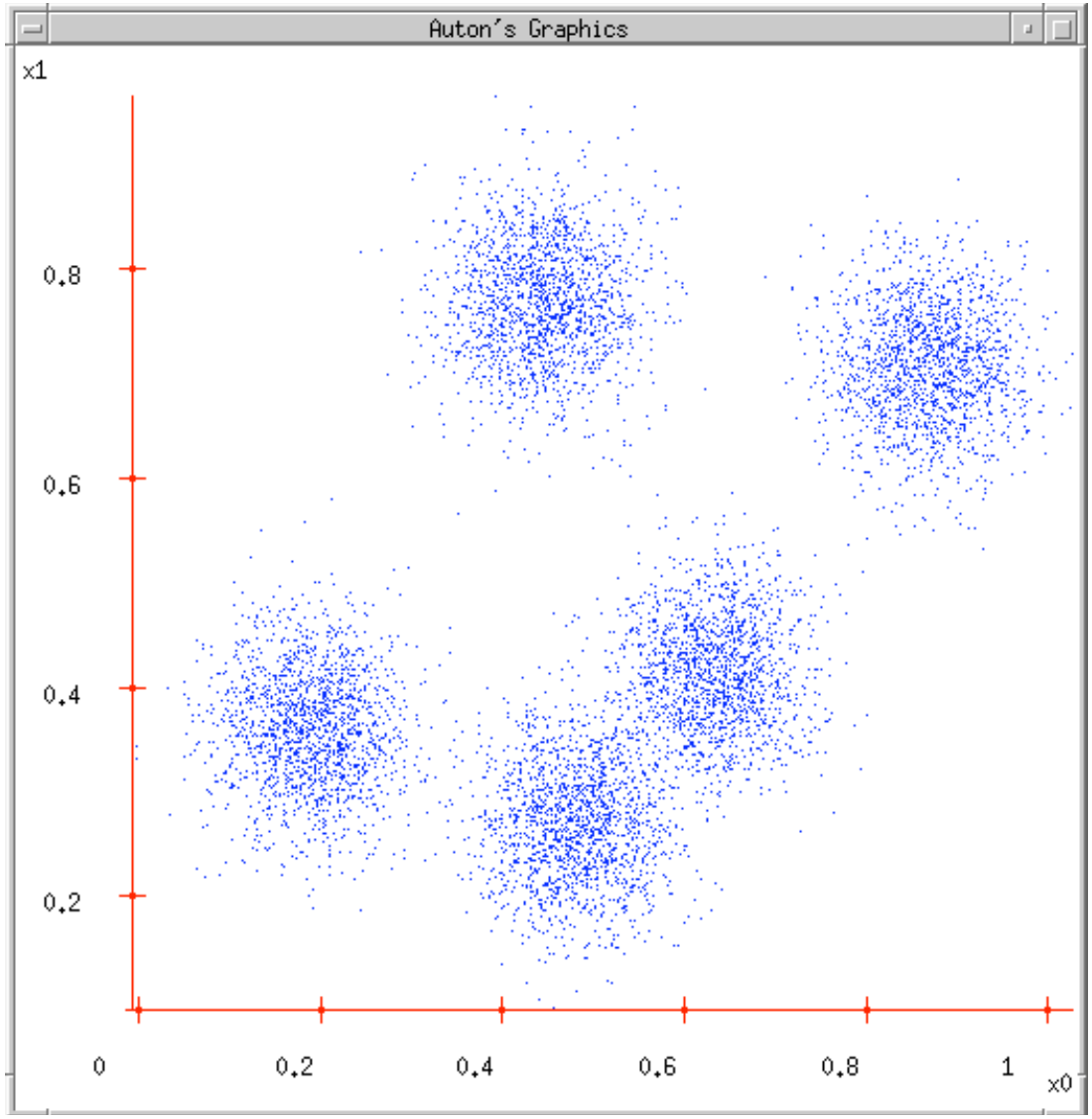
1) What makes two data points similar?

2) How do we compute an overall grouping from pairwise similarities?

How might we cluster?

- K-means
  - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters

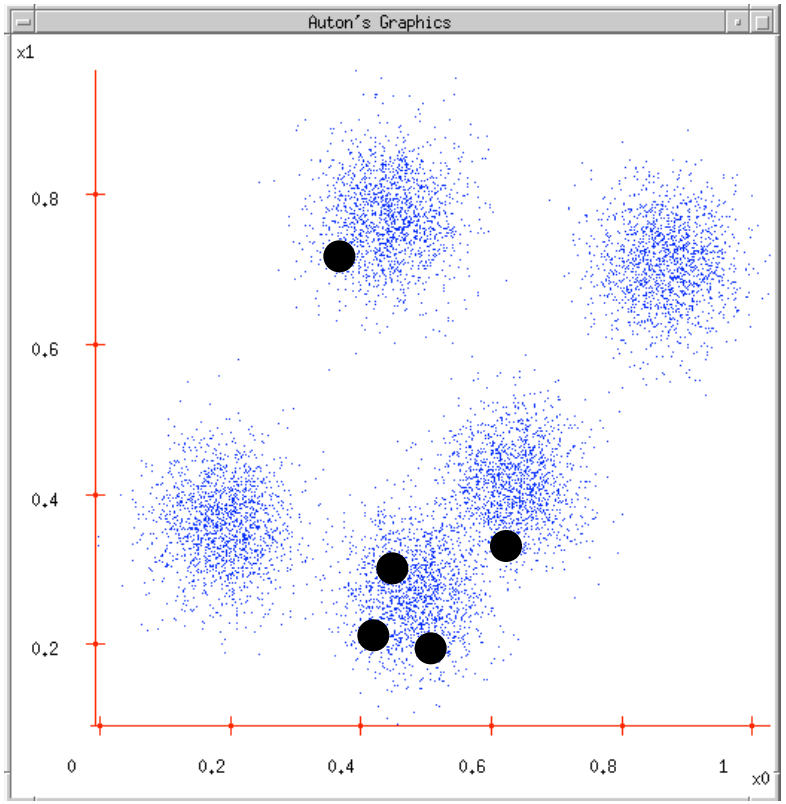
Clustering Data



K-Means Clustering

K-Means (k , X)

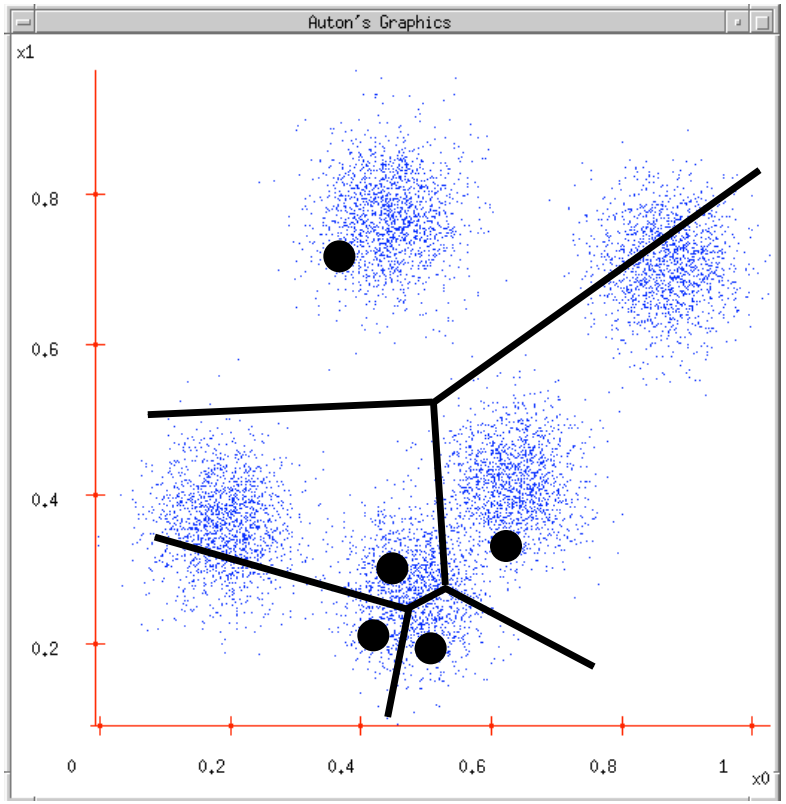
- Randomly choose k cluster center locations (centroids)
- Loop until convergence
  - Assign each point to the cluster of the closest centroid
  - Re-estimate the cluster centroids based on the data assigned to each cluster



K-Means Clustering

K-Means (k , X)

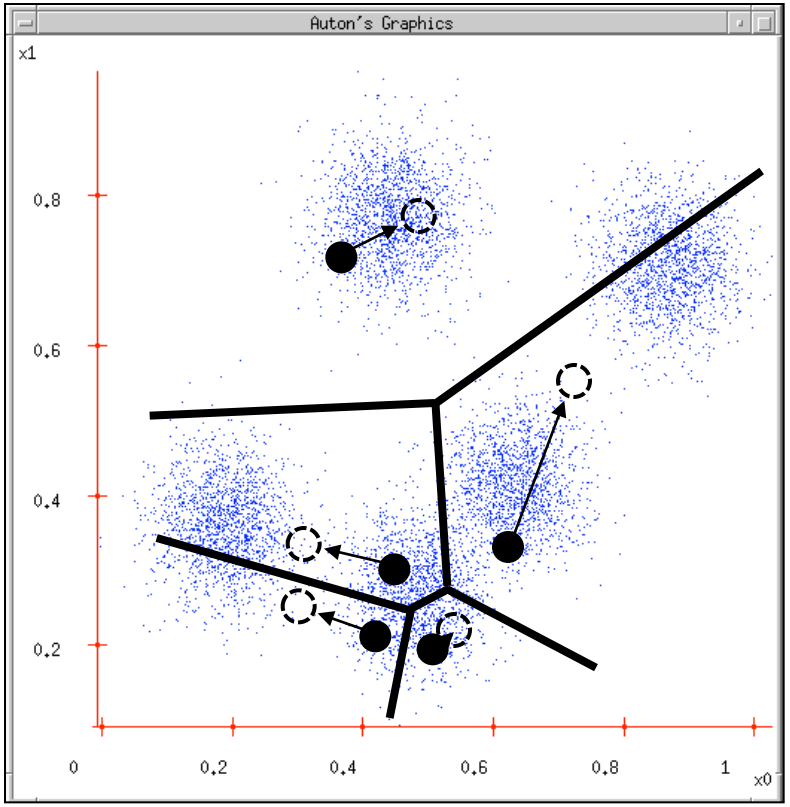
- Randomly choose k cluster center locations (centroids)
- Loop until convergence
  - Assign each point to the cluster of the closest centroid
  - Re-estimate the cluster centroids based on the data assigned to each cluster



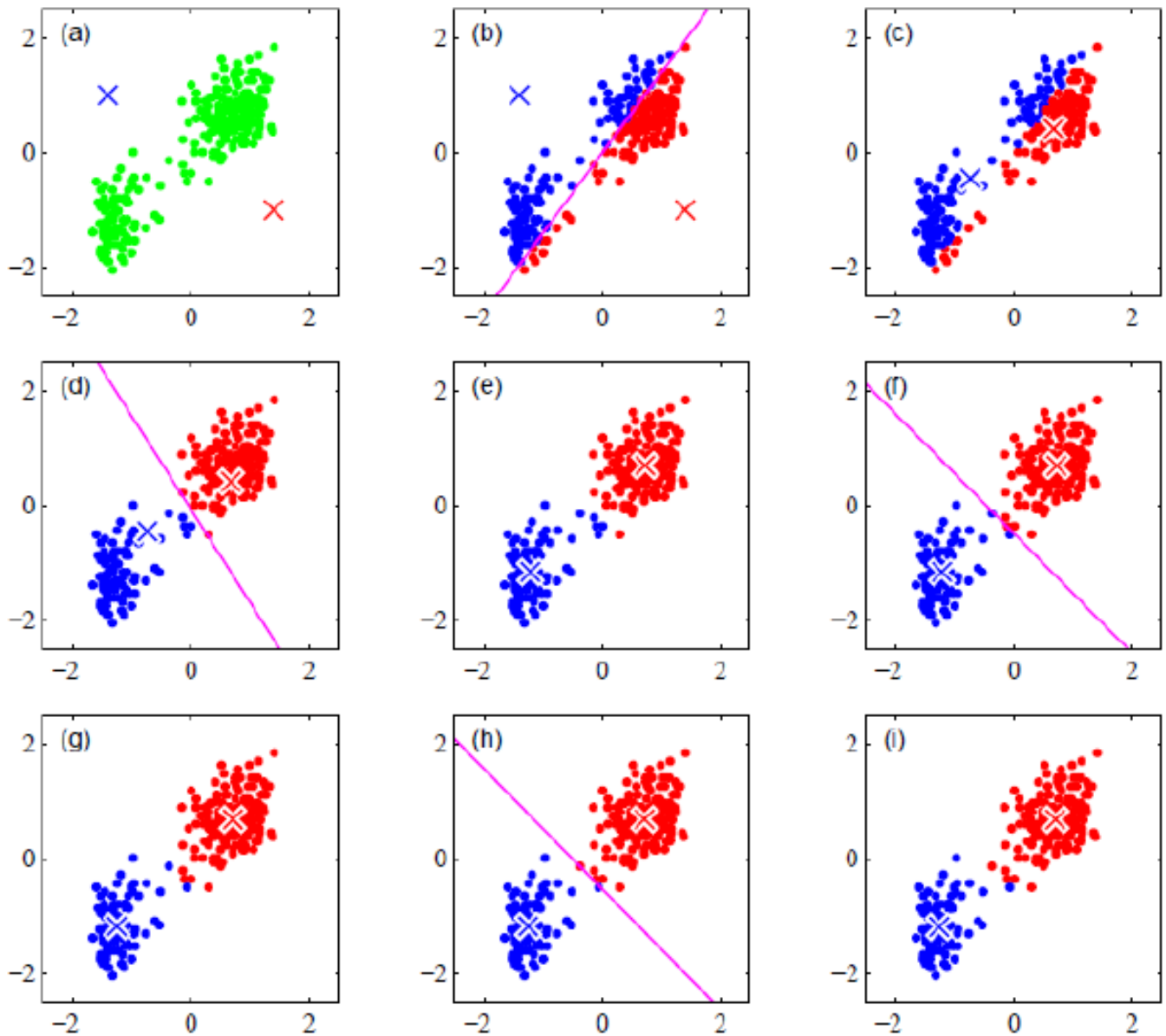
K-Means Clustering

K-Means (k , X)

- Randomly choose k cluster center locations (centroids)
- Loop until convergence
  - Assign each point to the cluster of the closest centroid
  - Re-estimate the cluster centroids based on the data assigned to each cluster



K-means Example (Several Iterations)



K-Means Objective Function

- K-means finds a local optimum of the following objective function:

$$\arg\min_{\boldsymbol{\mathcal{S}}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{S}_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|_2^2$$

where $\boldsymbol{\mathcal{S}} = \{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$ is a partitioning over $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ s.t. $X = \bigcup_{i=1}^{k} \mathcal{S}_i$ and $\boldsymbol{\mu}_i = \mathrm{mean}(\mathcal{S}_i)$

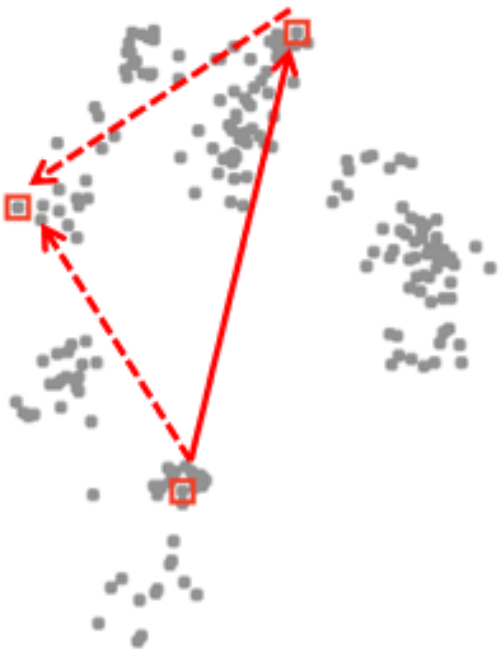
Initialization methods

Random

- Usually, choose random data index
- Ensures centers are near some data
- Issue: may choose nearby points

Distance-based

- Start with one random data point
- Find the point farthest from the clusters chosen so far
- Issue: may choose outliers

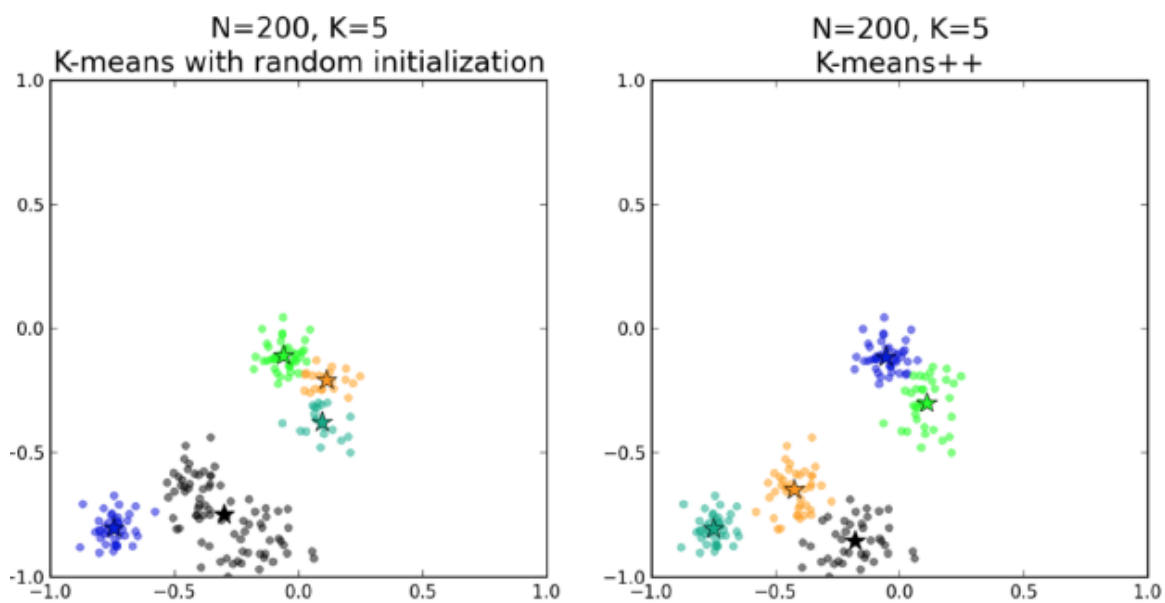


K-means++

Key idea: Run K-means, but with a better initialization

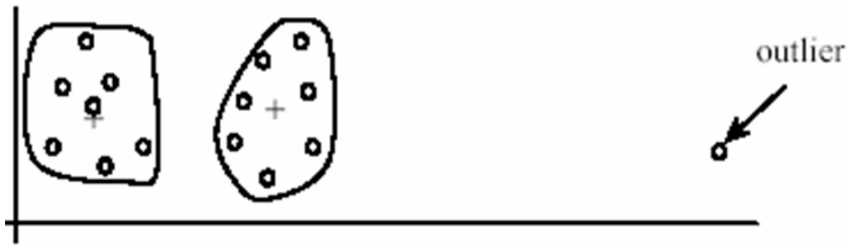
- Choose center μ_1 at random
- For $j = 2, \ldots, k$
  - Choose μ_j among $x_1, \ldots, x_n$ with high probability:

Meaning that if x_i is close to one of the already chosen cluster means $\mu_1, \ldots \mu_{j-1}$, then we assign a lower probability of selecting it as the next cluster mean.

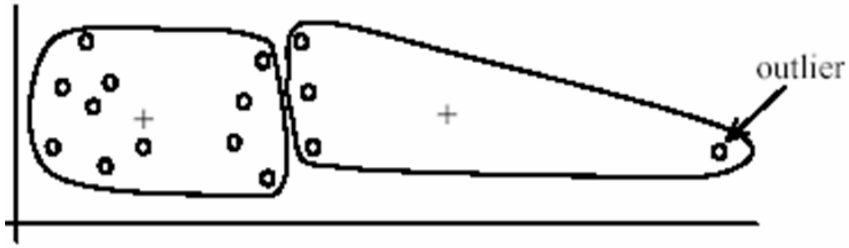


K-Means pros and cons

- Pros
  - Finds cluster centers that minimize conditional variance (good representation of data)
  - Easy to implement
- Cons
  - Need to choose K
  - Sensitive to outliers
  - Prone to local minima
  - All clusters have the same parameters (e.g., distance measure is non-adaptive)



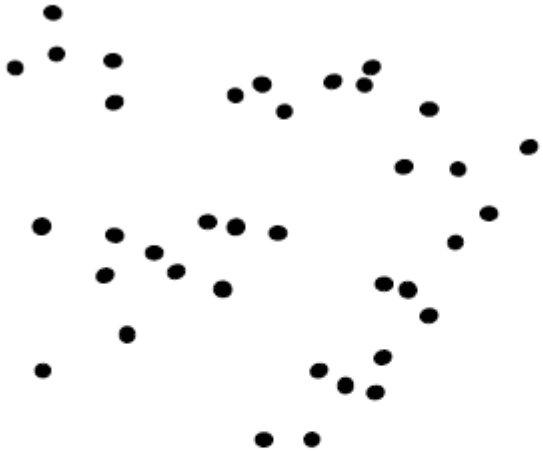
(B): Ideal clusters



How might we cluster?

- K-means
  - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters

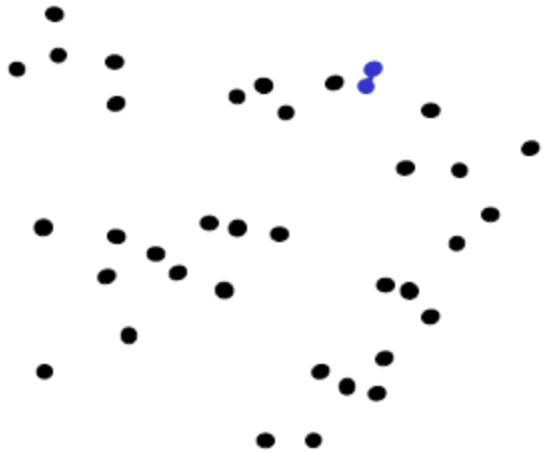
Agglomerative clustering



1. Say "Every point is its own cluster"

Copyright © 2001, 2004, Andrew W. Moore

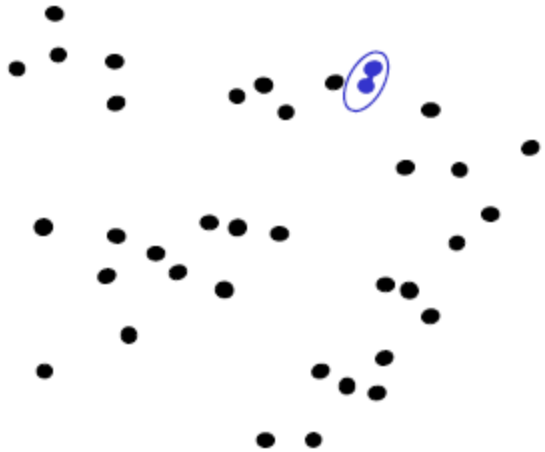
Agglomerative clustering



1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters

Copyright © 2001, 2004, Andrew W. Moore

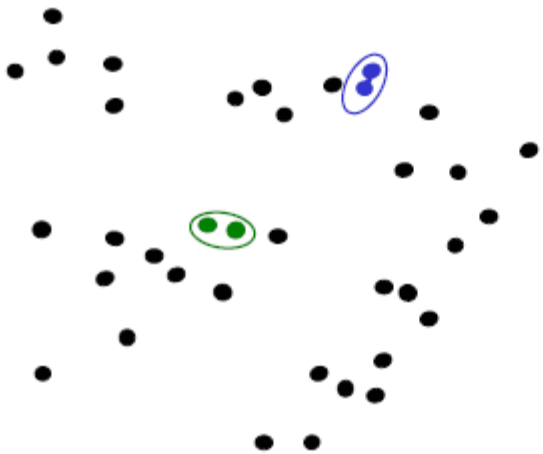
Agglomerative clustering



1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster

Copyright © 2001, 2004, Andrew W. Moore

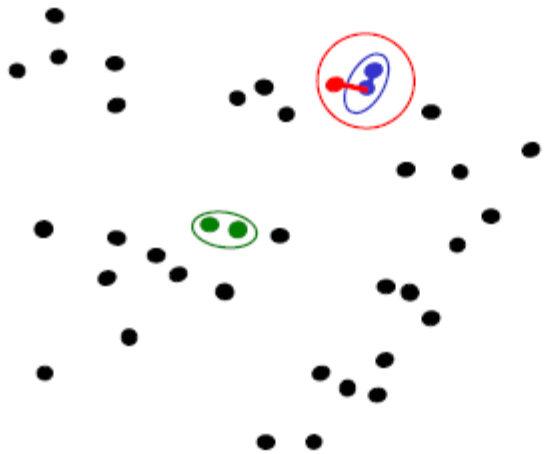
Agglomerative clustering



1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Copyright © 2001, 2004, Andrew W. Moore

Agglomerative clustering

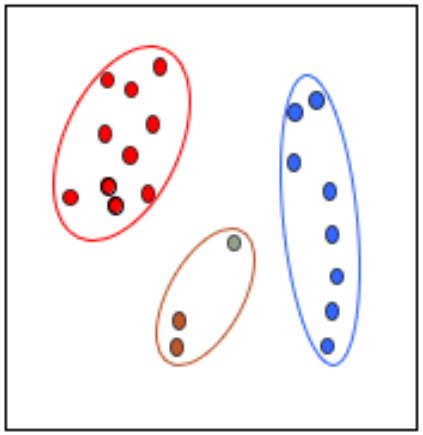


1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

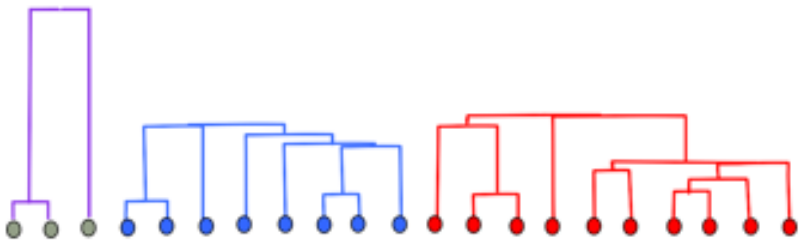
Copyright © 2001, 2004, Andrew W. Moore

Agglomerative clustering

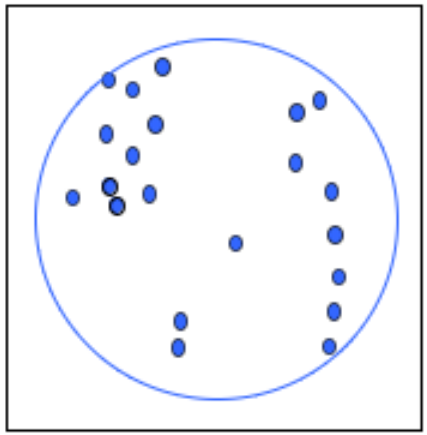
Data:



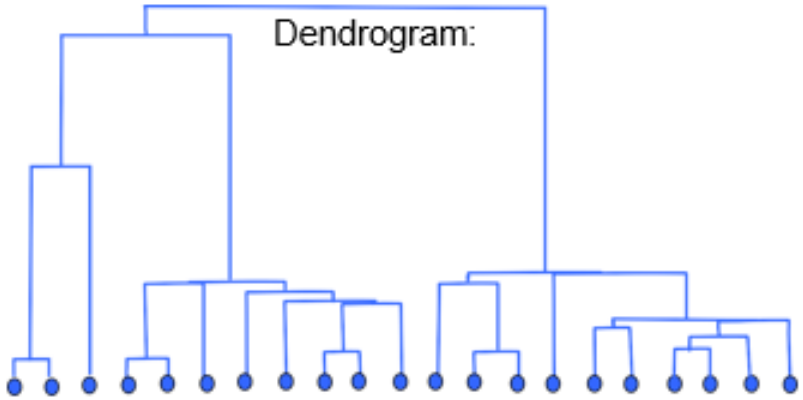
Dendrogram:



Data:



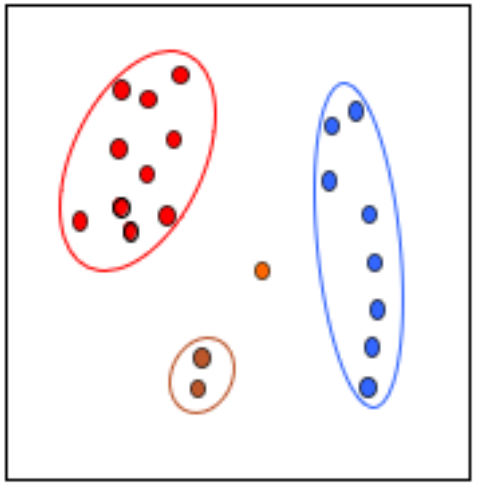
Dendrogram:



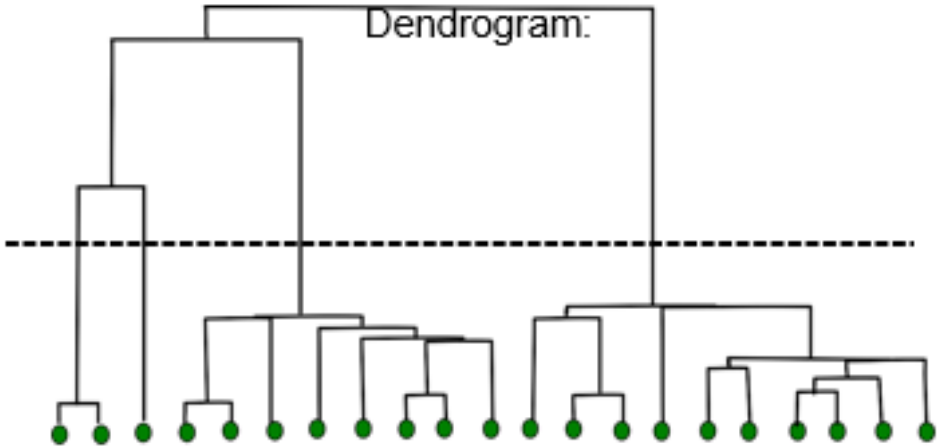
Agglomerative clustering

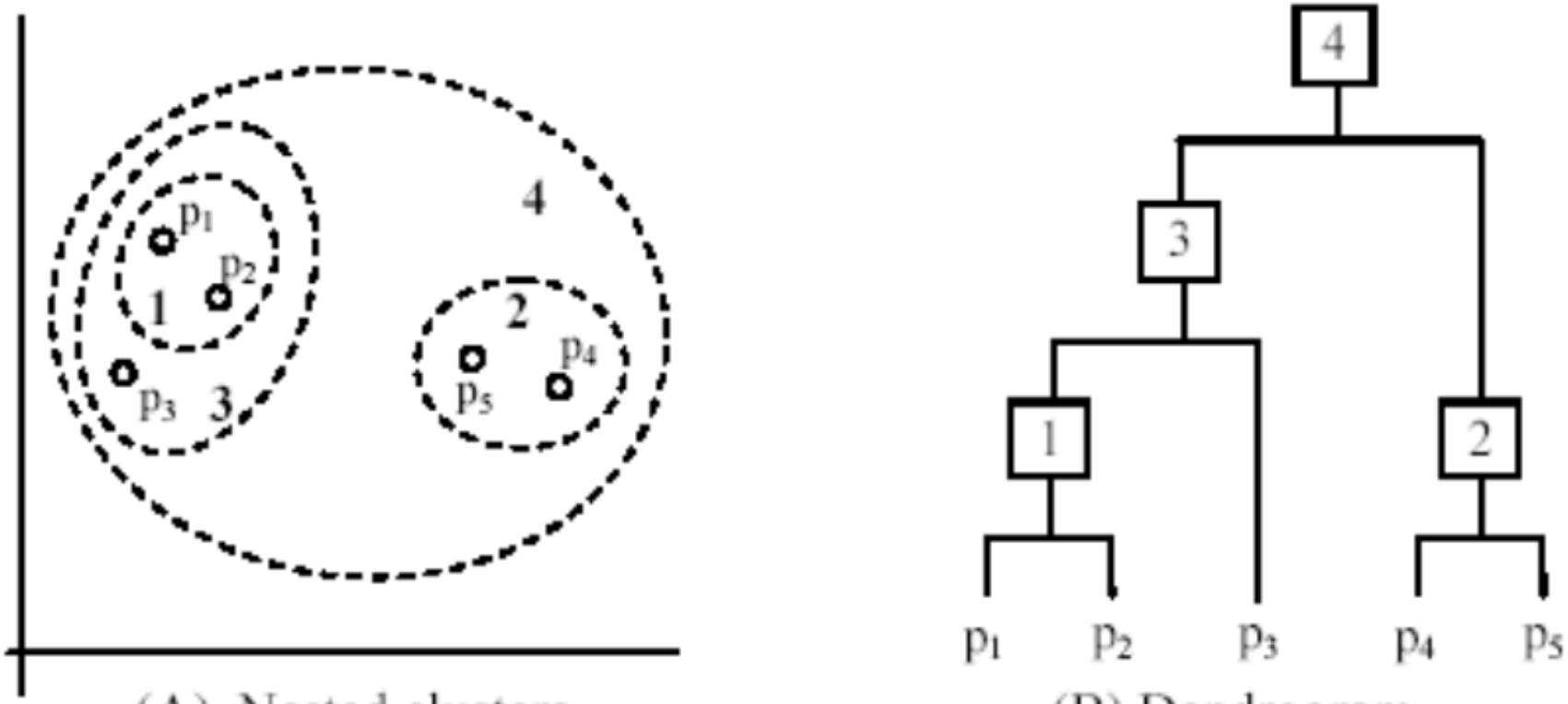
Given the sequence, can select a number of clusters or a dissimilarity threshold:

Data:



Dendrogram:



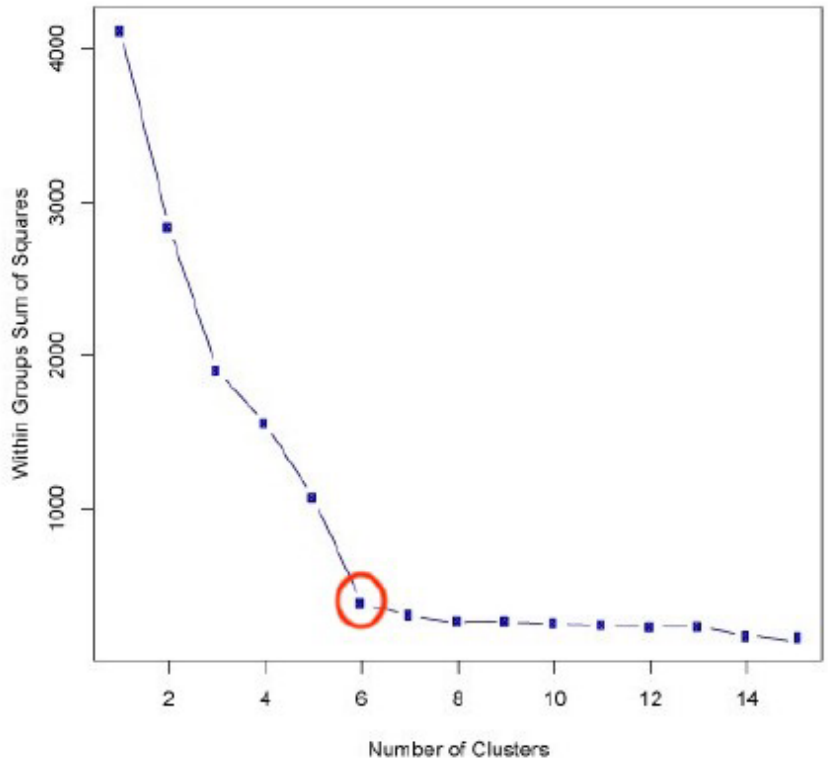


(A). Nested clusters (B) Dendrogram

Elbow Method

- How to select k?
  - Prior knowledge
  - Heuristics (e.g., elbow method)

Select a small value of k that adding a new cluster doesn't reduce the within-cluster distances much

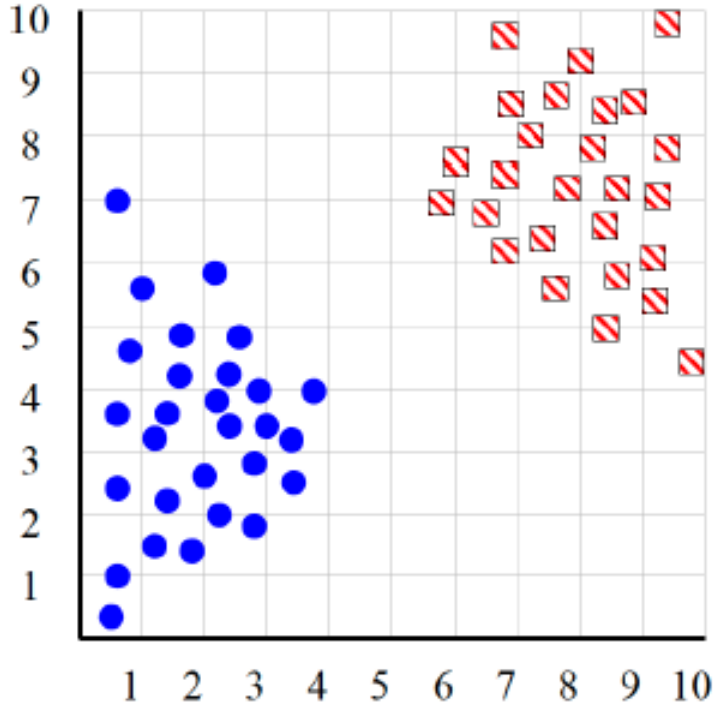


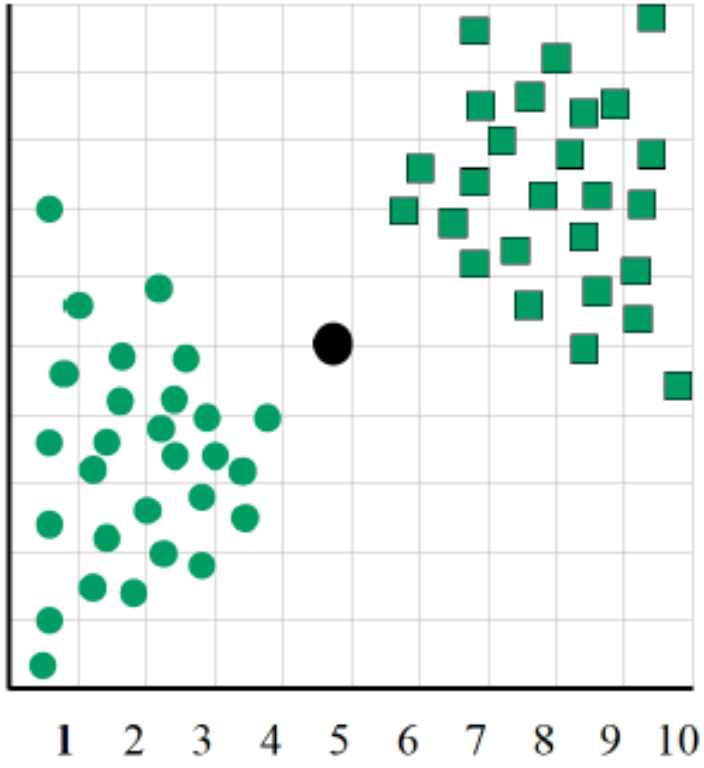
- For each cluster, measure distance of every point to its centroid
- Sum these distances across all clusters
- This total is the within-cluster (intra-cluster) distance

Elbow Method

How can we find the right number of clusters? Track the objective function as we increase k!

When $k = 1$, objective value is 873.

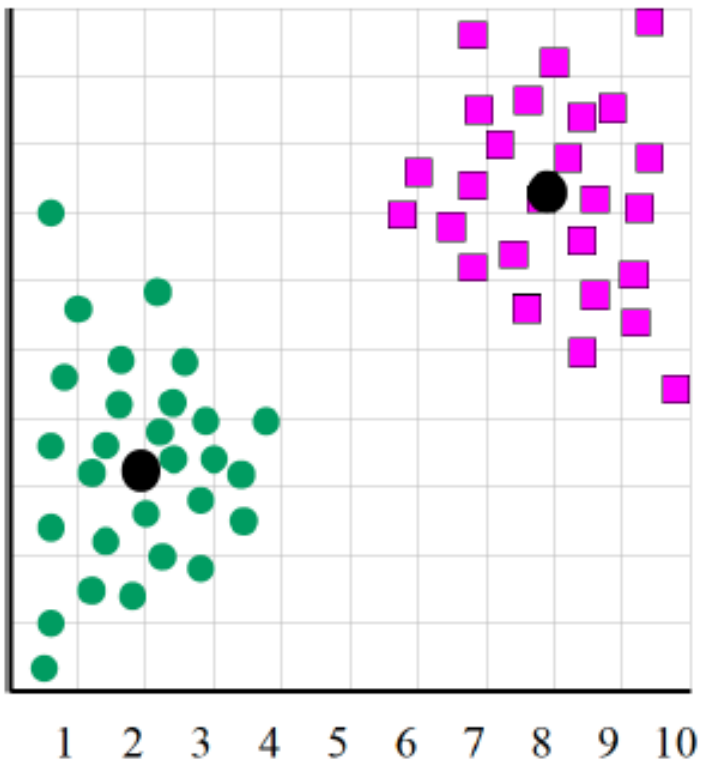




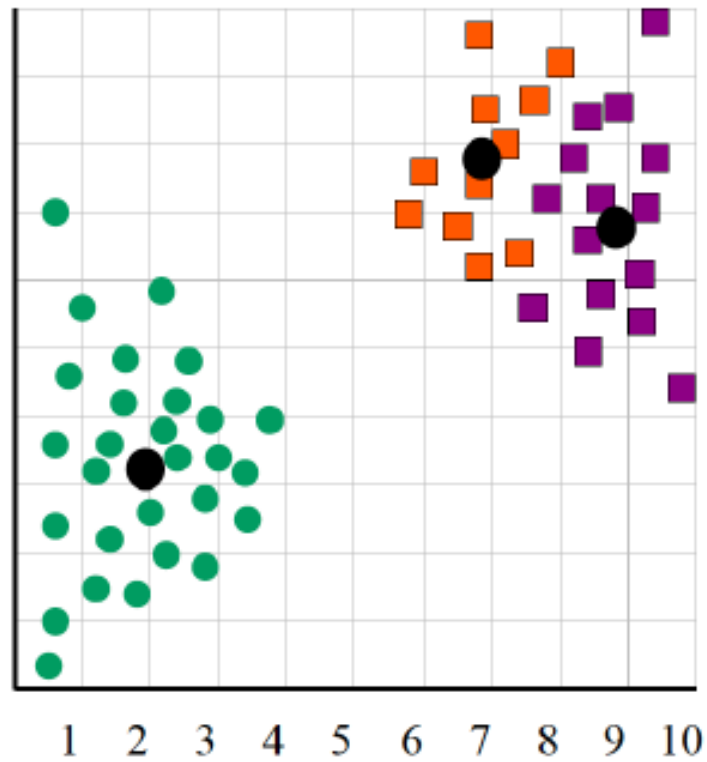
Elbow Method

How can we find the right number of clusters? Track the objective function as we increase k!

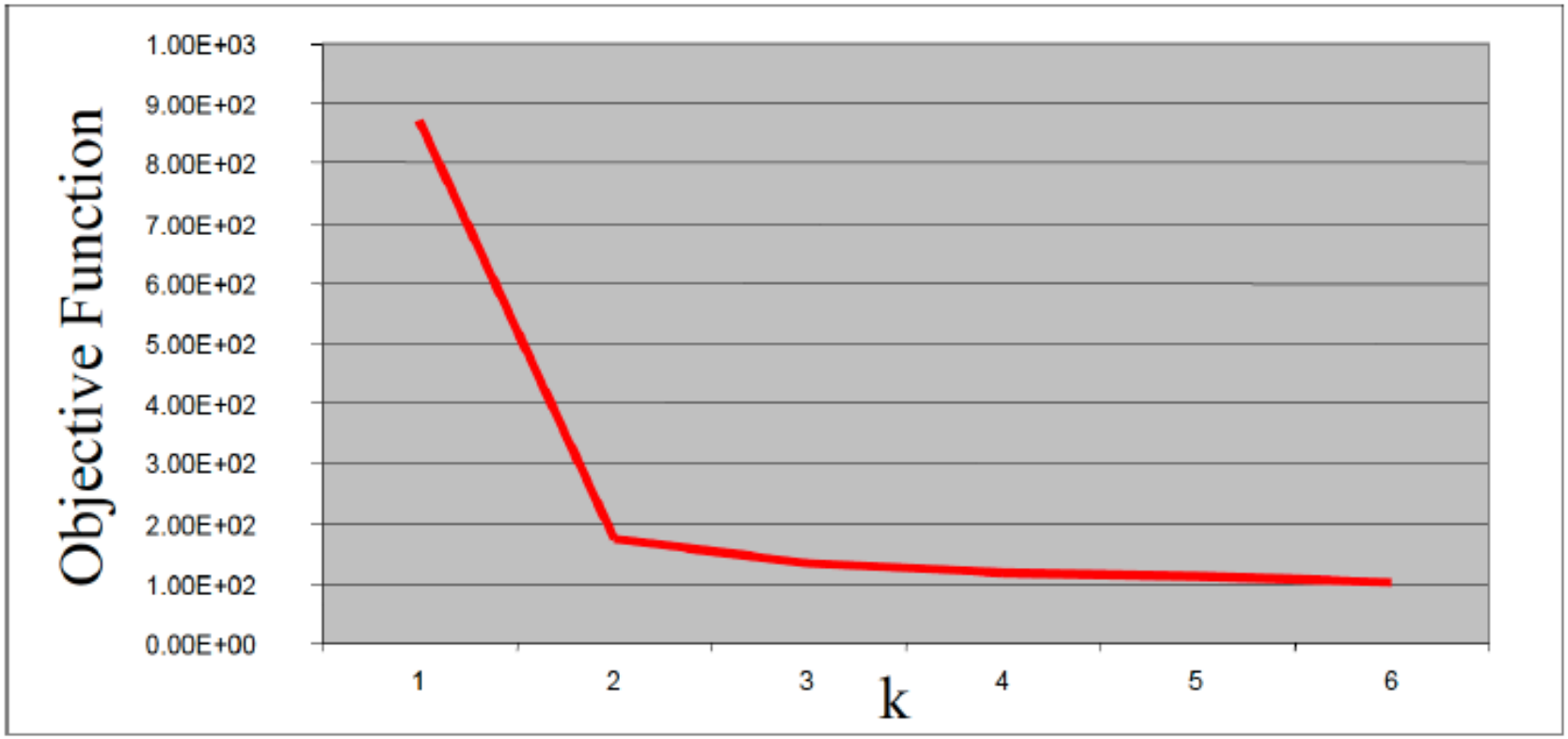
When $k = 2$, objective value is 173.1.



When $k = 3$, objective value is 133.6.



In this case, there is an abrupt change at $k = 2$ that suggests there are two "natural" clusters in the data.



Performance metrics

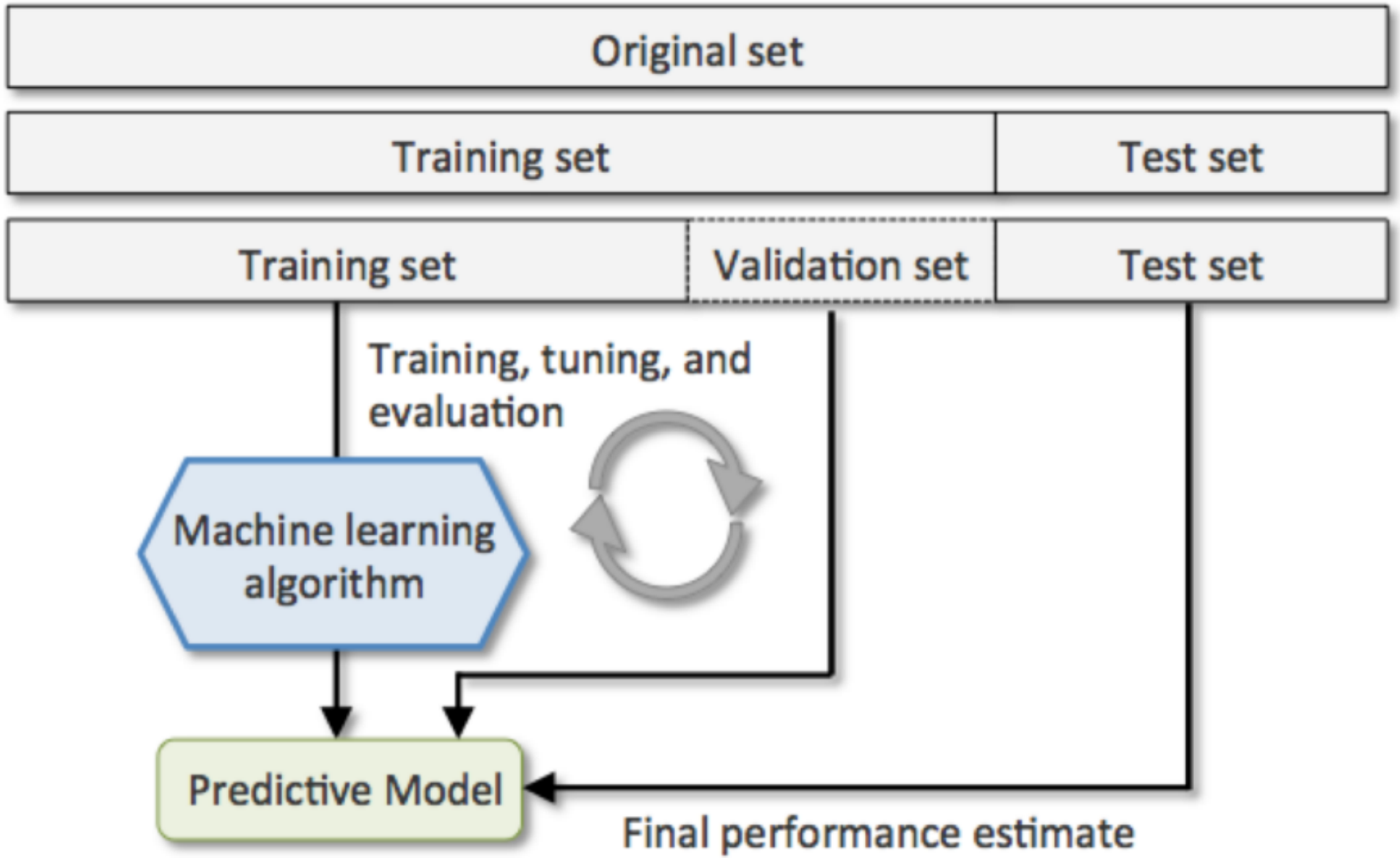
1. Metrics

2. Precision and recall

3. Receiver Operating Characteristic (ROC) curves

4. Worked example

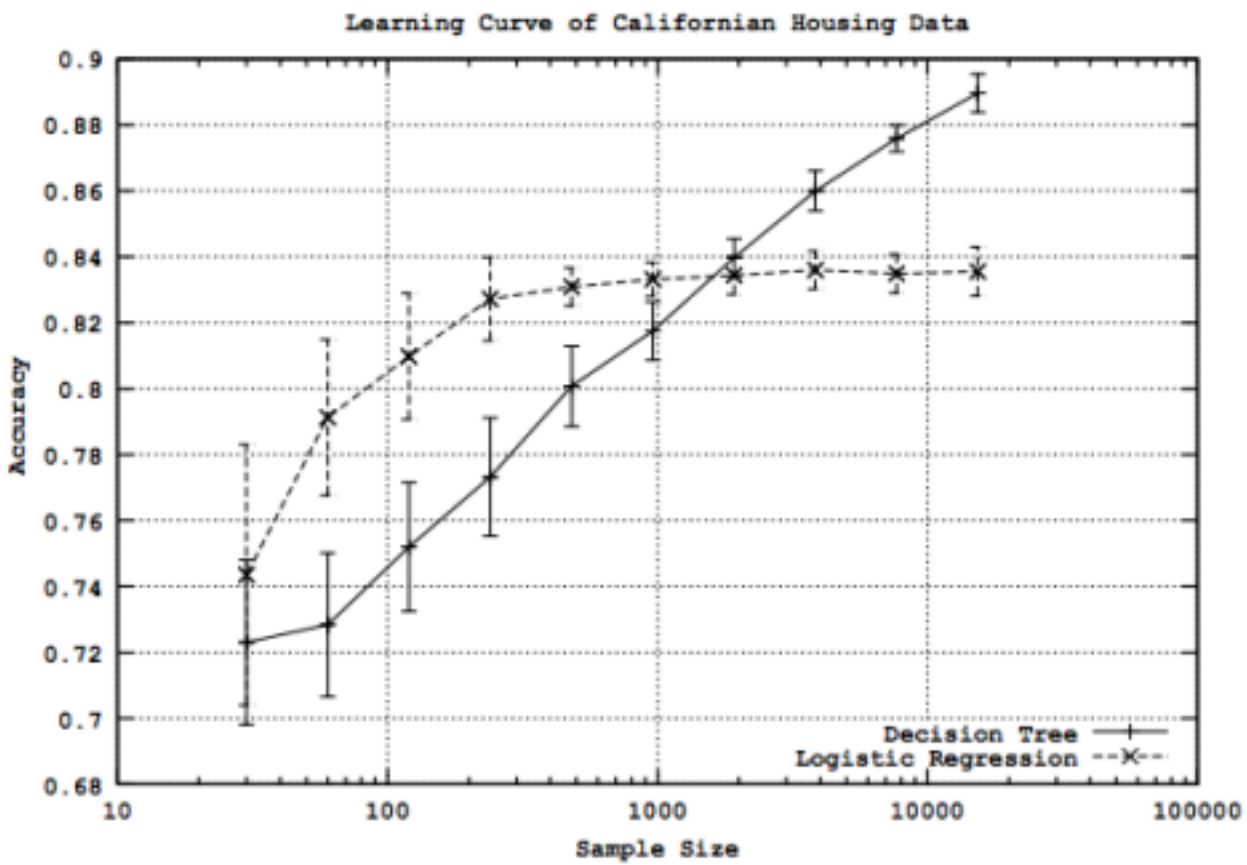
Evaluation


Original set
Training set
Test set
Training set
Validation set
Test set
Training, tuning, and evaluation
Machine learning algorithm
Predictive Model
Final performance estimate

Learning curves

How does the accuracy of a learning method change as a function of the training-set size?

this can be assessed by plotting *learning curves*



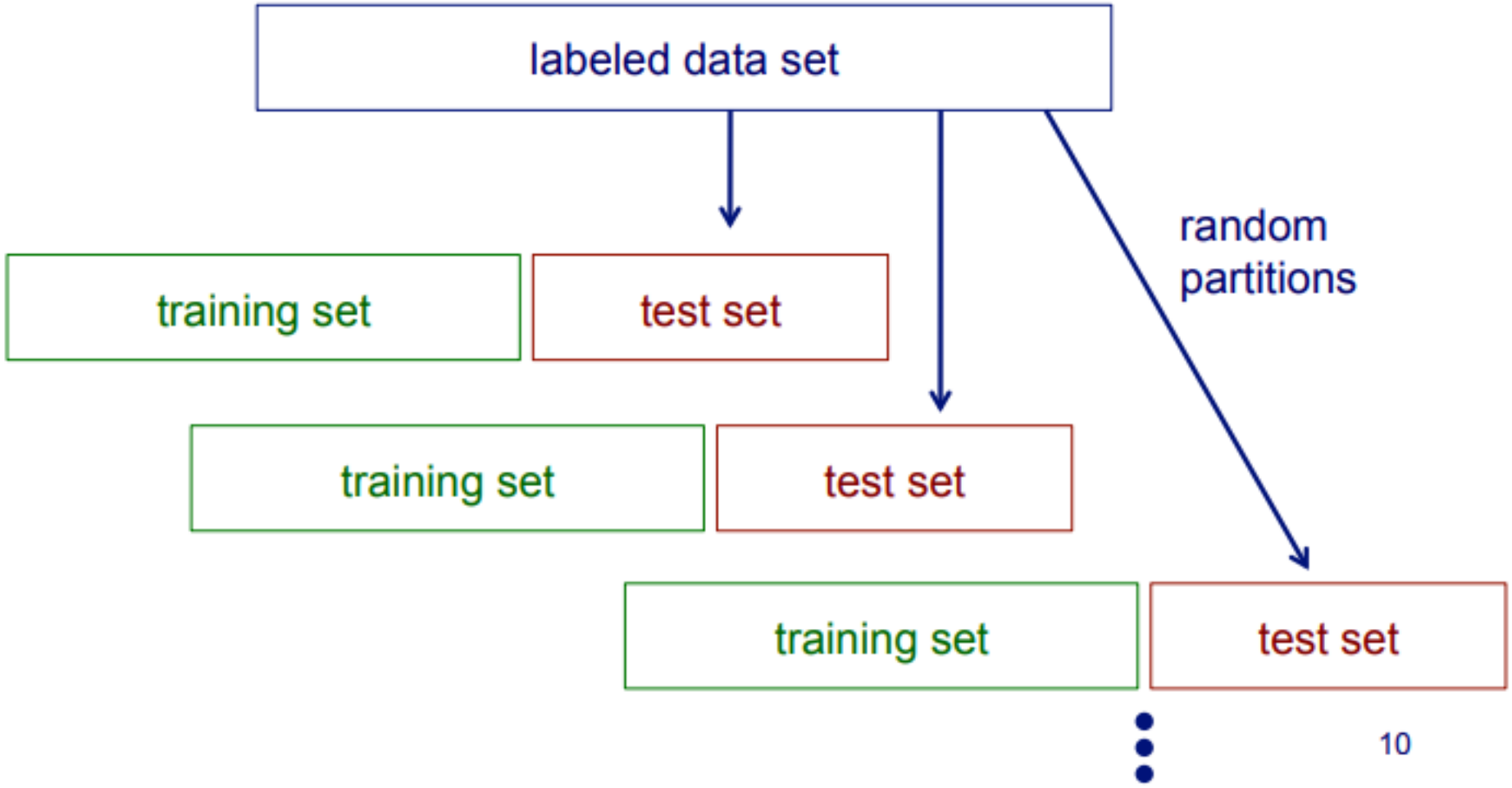
Limitations of using a single training/test partition

- we may not have enough data to make sufficiently large training and test sets
  - a larger test set gives us more reliable estimate of accuracy (i.e. a lower variance estimate)
  - but… a larger training set will be more representative of how much data we actually have for learning process

- a single training set doesn't tell us how sensitive accuracy is to a particular training sample

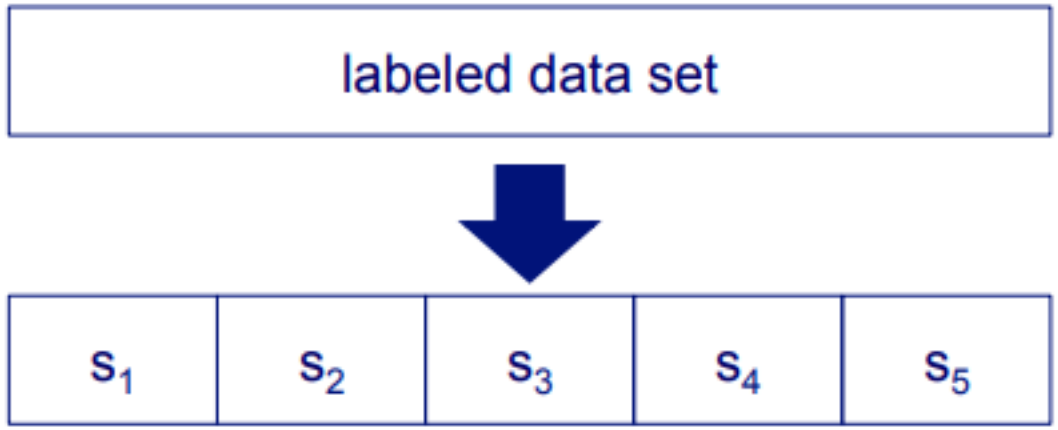
Random resampling

We can address the second issue by repeatedly randomly partitioning the available data into training and set sets.



Cross validation

partition data
into *n* subsamples



iteratively leave one subsample out for the test set, train on the rest

iteration	train on	test on
1	s_2 s_3 s_4 s_5	s_1
2	s_1 s_3 s_4 s_5	s_2
3	s_1 s_2 s_4 s_5	s_3
4	s_1 s_2 s_3 s_5	s_4
5	s_1 s_2 s_3 s_4	s_5

Cross validation example

Suppose we have 100 instances, and we want to estimate accuracy with cross validation

iteration	train on	test on	correct
1	s_2 s_3 s_4 s_5	s_1	11 / 20
2	s_1 s_3 s_4 s_5	s_2	17 / 20
3	s_1 s_2 s_4 s_5	s_3	16 / 20
4	s_1 s_2 s_3 s_5	s_4	13 / 20
5	s_1 s_2 s_3 s_4	s_5	16 / 20

accuracy = 73/100 = 73%

Metrics

It is extremely important to use **quantitative metrics** for evaluating a machine learning model

- Until now, we relied on the **cost function value** for regression and classification

- Other metrics can be used to **better evaluate** and understand the model

- **For classification**
  - ✓ Accuracy/Precision/Recall/F1-score, ROC curves,…
- **For regression**
  - ✓ Normalized RMSE, Normalized Mean Absolute Error (NMAE),…

2x2 Confusion Matrix

		Predicted class	
		P	N
Actual class	P	True positives (TP)	False negatives (FN)
	N	False positives (FP)	True negatives (TN)

$$ERR = \frac{FP + FN}{FP + FN + TP + TN} = 1 - ACC$$

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} = 1 - ERR$$

Accuracy

Accuracy is a measure of **how close** model prediction are to their true value.

$$\text{Accuracy} = \frac{\#\ \text{Correct classifications}}{\#\ \text{All classifications}}$$

- If a classifier make 10 predictions and 9 of them are correct, the accuracy is 90%.

- Accuracy is a measure of **how well** a binary classifier correctly identifies or excludes a condition.
- It's the **proportion of correct predictions among the total number of cases examined.**

- Consider a fraud detection model for credit card transactions. Suppose we have a dataset with 1000 transactions, where 990 are legitimate (majority class) and 10 are fraudulent (minority class).
- Positive class = Fraudulent transactions
- Negative class = Legitimate transactions

Model Performance:

- The model correctly identifies 980 legitimate transactions (true negatives) and 8 fraudulent transactions (true positives).
- However, it misclassifies 2 fraudulent transactions as legitimate (false negatives) and 10 legitimate transactions as fraudulent (false positives).

Accuracy Calculation:

Total correct predictions: 980 (true negatives) + 8 (true positives) = 988

Accuracy: 988/1000 = 0.988 or 98.8%

The problem:

- The accuracy metric suggests the model is performing extremely well (98.8%), but it's actually performing poorly on the minority class (fraudulent transactions).

- The model only detected 8 out of 10 fraudulent transactions (80% recall) and misclassified 2 fraudulent transactions as legitimate (false negatives).

Precision and recall

Suppose that $y = 1$ in presence of a **rare class** that we want to detect

Precision *(How much we are precise in the detection)*

Of all patients where we classified $y = 1$, what fraction actually has the disease?

$$\frac{\text{True Positive}}{\text{\# Estimated Positive}} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

Recall *(How much we are good at detecting)*

Of all patients that actually have the disease, what fraction did we correctly detect as having the disease?

$$\frac{\text{True Positive}}{\text{\# Actual Positive}} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

Confusion matrix

Estiamted class \ Actual class	1 (p)	0 (n)
1 (Y)	True positive (TP)	False positive (FP)
0 (N)	False negative (FN)	True negative (TN)

Trading off precision and recall

Logistic regression: $0 \leq s(\boldsymbol{\varphi}^\top \boldsymbol{\theta}) \leq 1$

- Classify 1 if $s(\boldsymbol{\varphi}^\top \boldsymbol{\theta}) > 0.5$
- Classify 0 if $s(\boldsymbol{\varphi}^\top \boldsymbol{\theta}) < 0.5$

These thresholds can be different from 0.5!

At different thresholds, correspond different confusion matrices!

Suppose we want to classify $y = 1$ (disease) only if very confident

- Increase threshold → Higher precision, lower recall

Suppose we want to avoid missing too many cases of disease (avoid false negatives)

- Decrease threshold → Higher recall, lower precision

Need for Unified Metric Harmonic Mean

	Precision(P)	Recall (R)	Average
Algorithm 1	0.5	0.4	0.45
Algorithm 2	0.7	0.1	0.4
Algorithm 3	0.02	1.0	0.51

- The harmonic mean gives more weight to lower values.

- If either precision or recall is low, the F1-score remains low, highlighting the imbalance.

- This ensures that a model performs well on both metrics, not just one.

F1-score

It is usually better to compare models by means of one number only. The $\mathbf{F1-score}$ can be used to **combine precision and recall**

	Precision(P)	Recall (R)	Average	F_1 Score
Algorithm 1	0.5	0.4	0.45	0.444
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1.0	0.51	0.0392

The best is Algorithm 1

Algorithm 3 classifies always 1

Average says not correctly that Algorithm 3 is the best

$$\text{Average} = \frac{P + R}{2} \qquad F_1\text{score} = 2\frac{P \cdot R}{P + R}$$

- $P = 0 \text{ or } R = 0 \Rightarrow F_1\text{score} = 0$
- $P = 1 \text{ and } R = 1 \Rightarrow F_1\text{score} = 1$

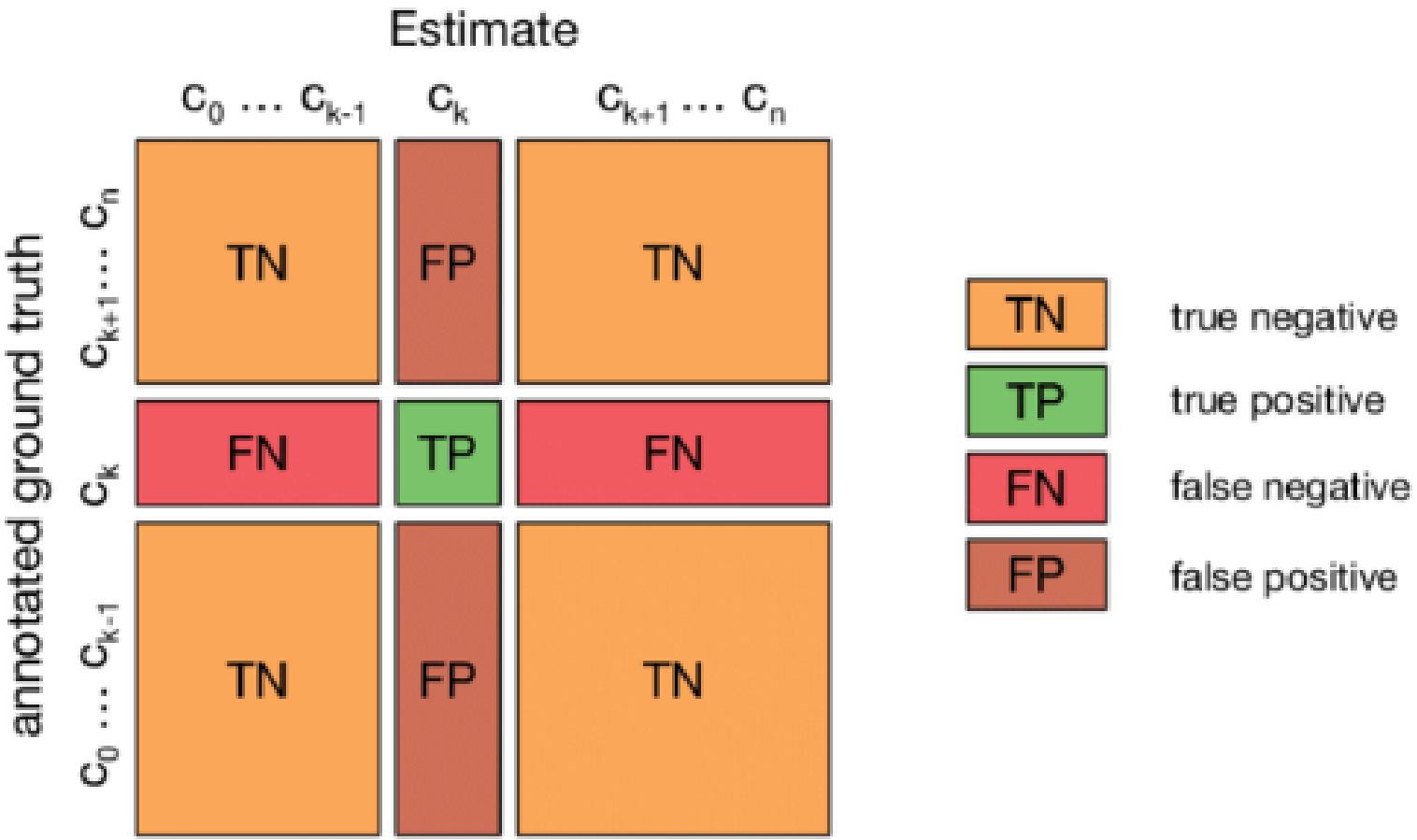
		urgent	normal	spam
system output	urgent	8	10	1
	normal	5	60	50
	spam	3	30	200

$$\textbf{precision}_u = \frac{8}{8+10+1}$$

$$\textbf{precision}_n = \frac{60}{5+60+50}$$

$$\textbf{precision}_s = \frac{200}{3+30+200}$$

$$\textbf{recall}_u = \frac{8}{8+5+3} \qquad \textbf{recall}_n = \frac{60}{10+60+30} \qquad \textbf{recall}_s = \frac{200}{1+50+200}$$


Estimate
$c_0 \dots c_{k-1}$
c_k
$c_{k+1} \dots c_n$
annotated ground truth
$c_{k+1} \dots c_n$
c_k
$c_0 \dots c_{k-1}$
TN
FP
TN
FN
TP
FN
TN
FP
TN
TN
true negative
TP
true positive
FN
false negative
FP
false positive

Micro- vs. Macro-Averaging

If we have more than one class, how do we combine multiple performance measures into one quantity?

Macro-averaging: Compute performance for each class, then average.

Micro-averaging: Collect decisions for all classes, compute contingency table, evaluate

Evaluation

system output \ gold labels	urgent	normal	spam	
urgent	8	10	1	$\text{precision}_u = \frac{8}{8+10+1}$
normal	5	60	50	$\text{precision}_n = \frac{60}{5+60+50}$
spam	3	30	200	$\text{precision}_s = \frac{200}{3+30+200}$
	$\text{recall}_u = \frac{8}{8+5+3}$	$\text{recall}_n = \frac{60}{10+60+30}$	$\text{recall}_s = \frac{200}{1+50+200}$	

Figure 6.5 Confusion matrix for a three-class categorization task, showing for each pair of classes (c_1, c_2), how many documents from c_1 were (in)correctly assigned to c_2

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

Summaries of the confusion matrix

Different metrics can be computed from the confusion matrix, depending on the class of interest *(https://en.wikipedia.org/wiki/Precision_and_recall)*

		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$	F_1 score = $\frac{1}{\frac{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}{2}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		

Ranking instead of classifying

Instance description	True class	Score
……	1	0,99
……	1	0,98
……	0	0,96
……	0	0,90
……	1	0,88
……	1	0,87
……	0	0,85
……	1	0,80
……	0	0,70

	p	n
Y	0	0
N	100	100

	p	n
Y	1	0
N	99	100

	p	n
Y	2	0
N	98	100

	p	n
Y	2	1
N	98	99

	p	n
Y	6	4
N	94	96

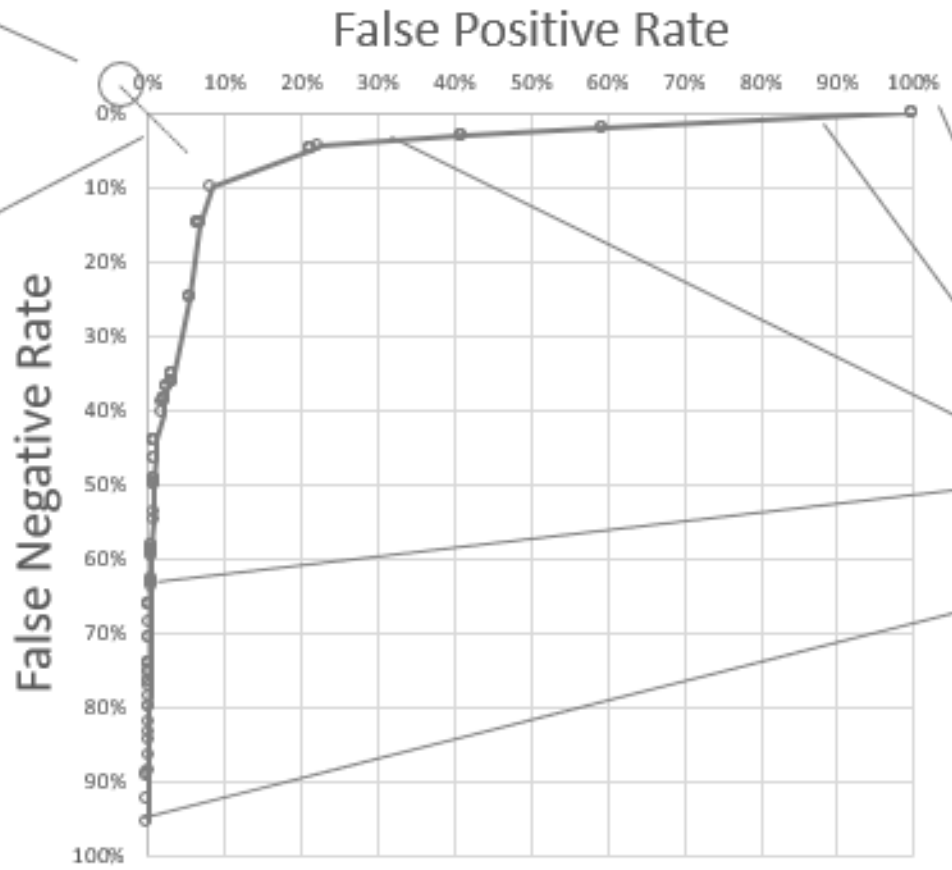
Different confusion matrices by changing the **threshold**

ROC Curve (Receiver Operating Characteristic)

Percent of 0s classified as 1

Perfect score:

- 0% of 1s called 0
- 0% of 0s called 1



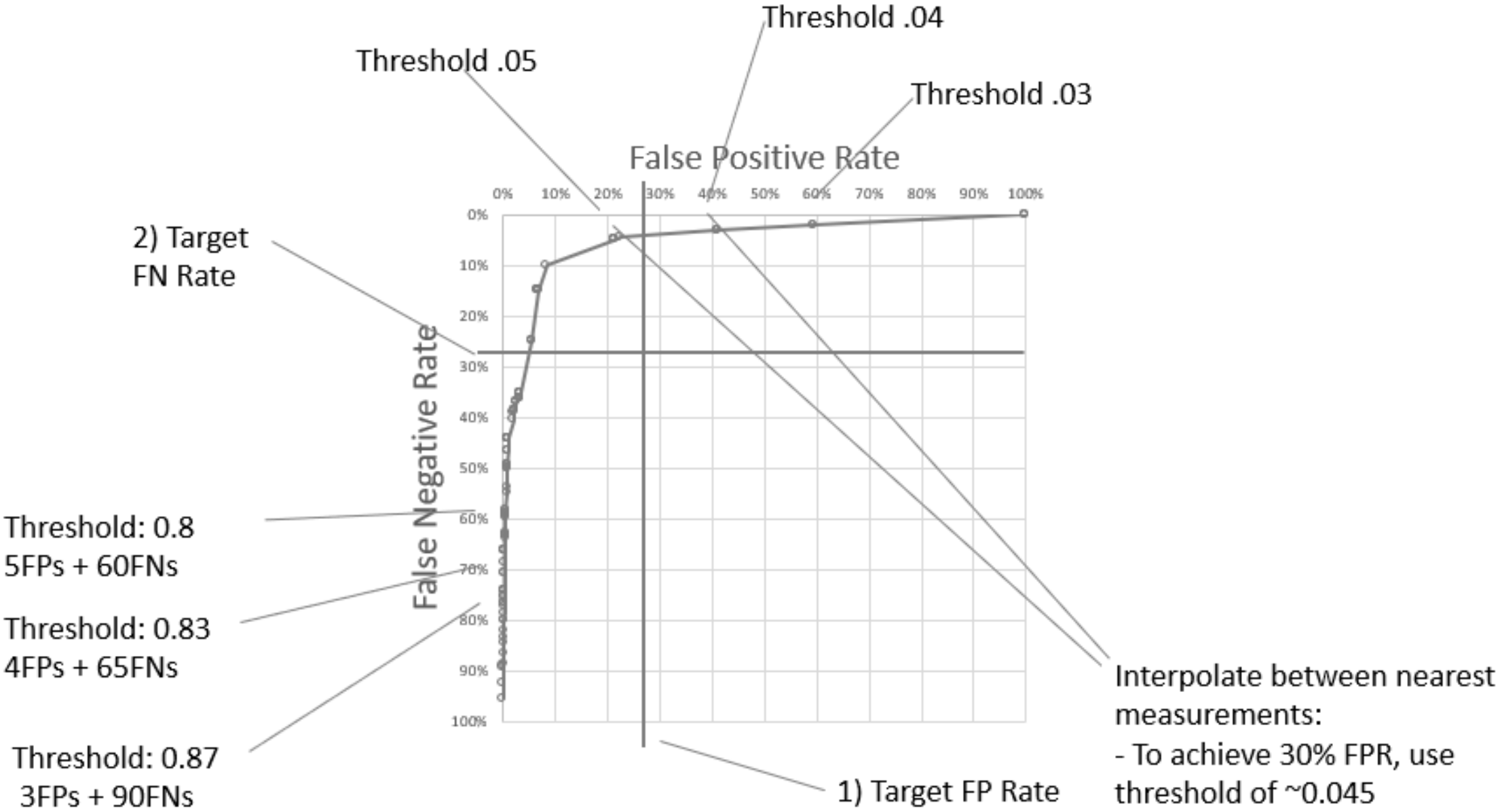
This model's distance from perfect

Sweep threshold from 0 to 1

- Threshold 0: 'all' classified as 1
- Threshold 1: 'all' classified as 0

Percent of 1s classified as 0

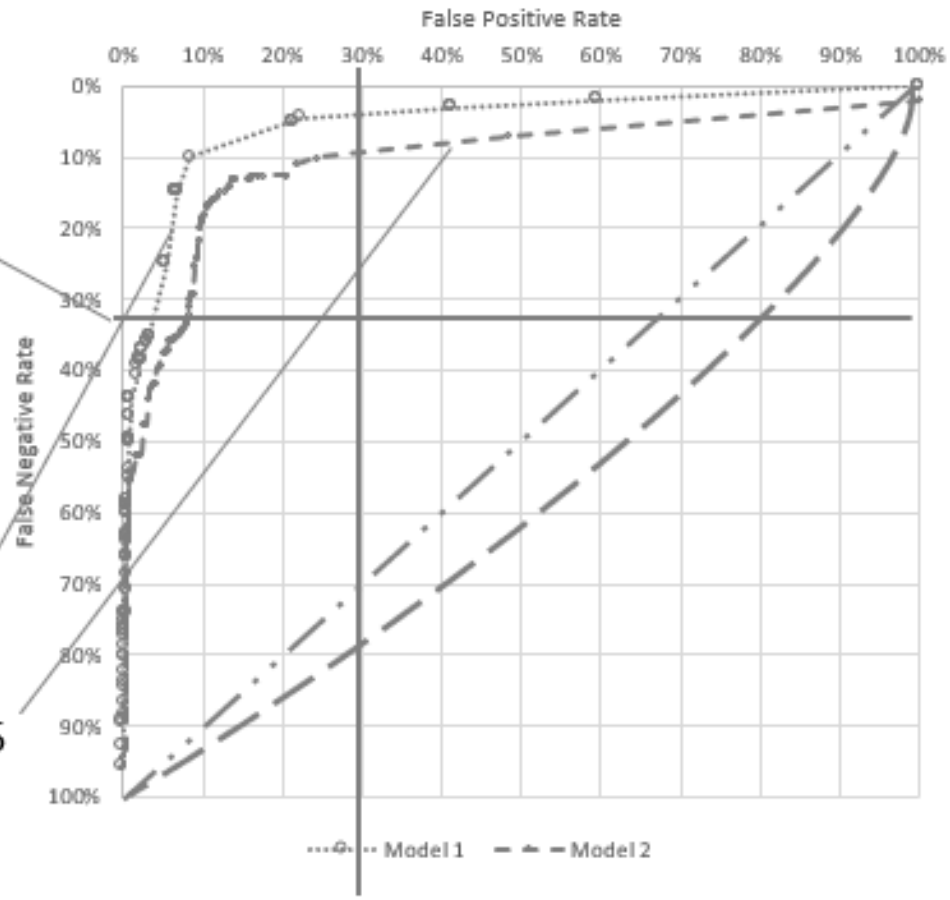
Operating Points



Comparing Models with ROC Curves

Model 1 better than Model 2 at every FPR or FNR target

Model 1 is better than Model 2 at this FNR



Model 1: AUC ~97
Model 2: AUC ~89.5

Model 1 better than Model 2 at this FPR

Area Under Curve (AUC)

- Integrate Area under the curve
- Perfect score is 1
- Higher scores allow for generally better tradeoffs
- AUC of 0.5 indicates model is essentially randomly guessing
- AUC of < 0.5 indicates you're doing something wrong…

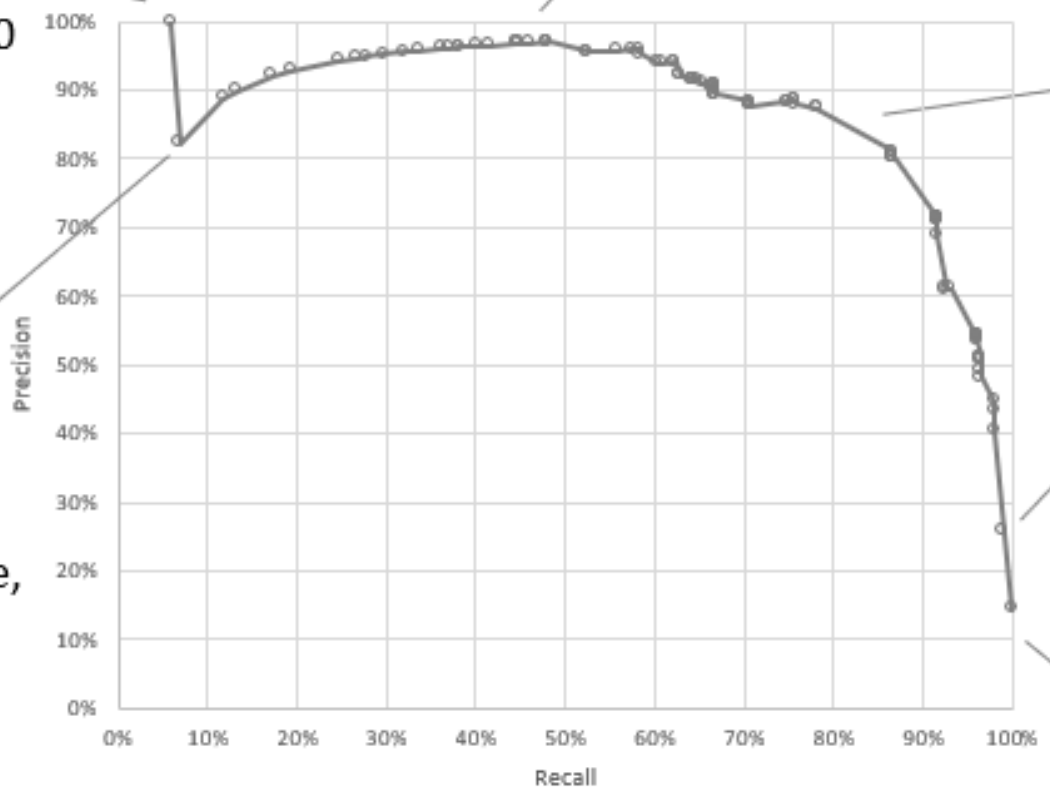
Precision Recall Curves – PR Curves

High Precision

Threshold near 1:
- Everything classified as 1 is a 1
- But many 1s are classified as 0

Low Recall

Incremental Classifications More Accurate:
- Threshold continuing lower
- 95%+ of new 1s are actually 1s



Incremental Classifications Less Accurate:
- Most new 1s are actually 0s

Near perfect recall:
- All 1s classified as 1s
- Precision around 25%

Sliding threshold lower:
- Classify a bunch of 0s as 1s
- Structure indicative of a sparse, strong feature

Threshold set to 0:
- Everything Classified as 1

You are shown a set of 21 coins: 10 gold and 11 copper. Your task to accept all gold coins and reject all copper ones

You accept 7 coins as being gold (these are your positives)

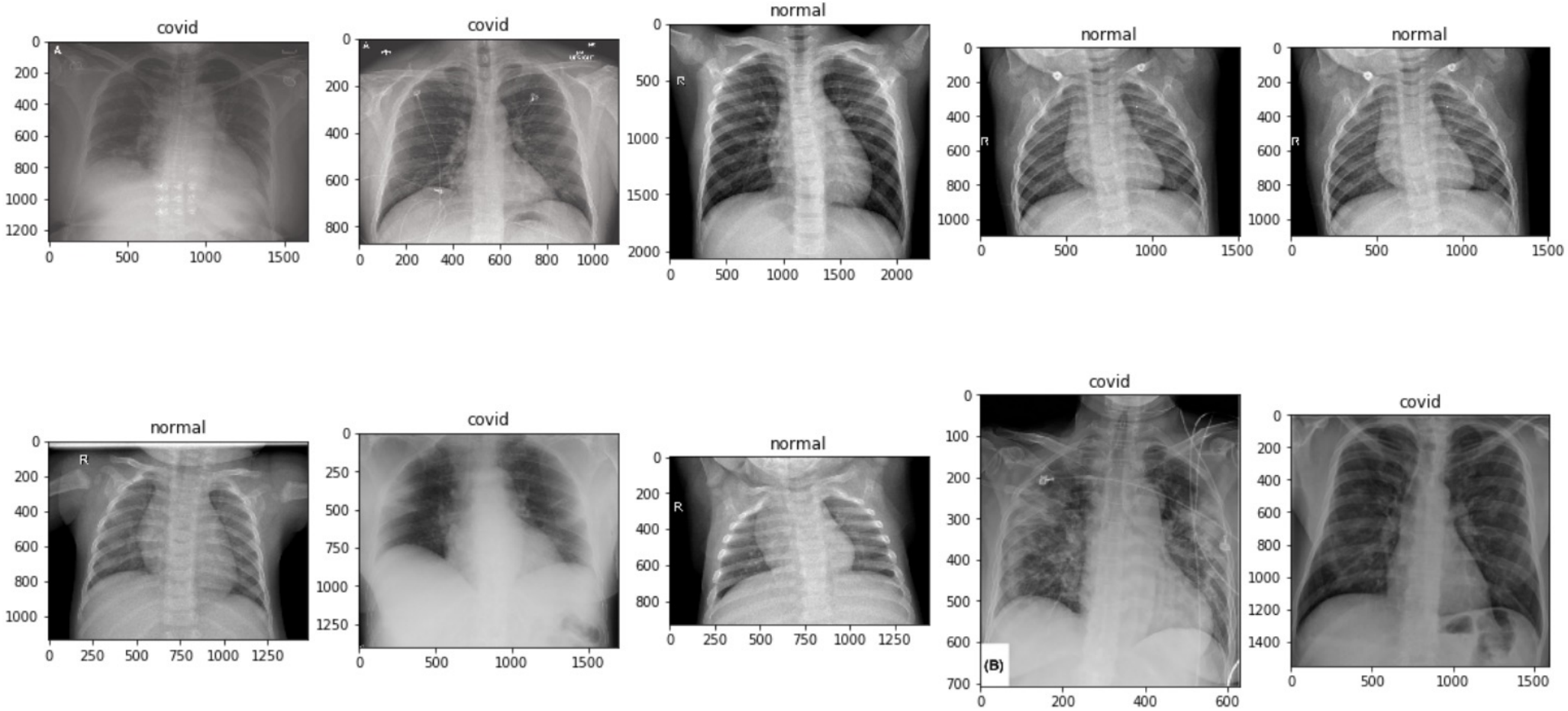
- 5 of these are actually gold (these are your true positives, tp)
- 2 of these are copper (these are your false positives, fp)
- You falsely rejected 5 gold ones (false negatives, fn)
- You correctly rejected 9 copper ones (true negatives, tn)

	Actual Gold	**Actual Copper**
Predicted Gold	5	2
Predicted Copper	5	9

- Your **precision** is $\frac{tp}{all\ of\ your\ positives} = \frac{tp}{tp+fp} = \frac{5}{7}$
- Your **recall** is $\frac{tp}{number\ of\ actual\ gold\ coins} = \frac{tp}{tp+fn} = \frac{5}{10}$
- Your **specificity** is $\frac{tn}{number\ of\ copper\ coins} = \frac{tn}{fn+tn} = \frac{9}{11}$
- Your **accuracy** is $\frac{correct\ answers}{all\ attempts} = \frac{tp+tn}{tp+tn+fp+fn} = \frac{5+9}{5+9+2+5} = \frac{14}{21}$

Pneumonia detection

Suppose to have at disposal X-ray images of lungs: **Healthy** people - **Covid-19 disease** patients



Acknowledgments

We want to use a classifier to perform classification:

- **Healthy** patients: class 0
- Patients with a **disease**: class 1

The input data are directly the X-ray **images**

For these computer vision tasks, the state of the art algorithm are the **Convolutional Neural Networks:**

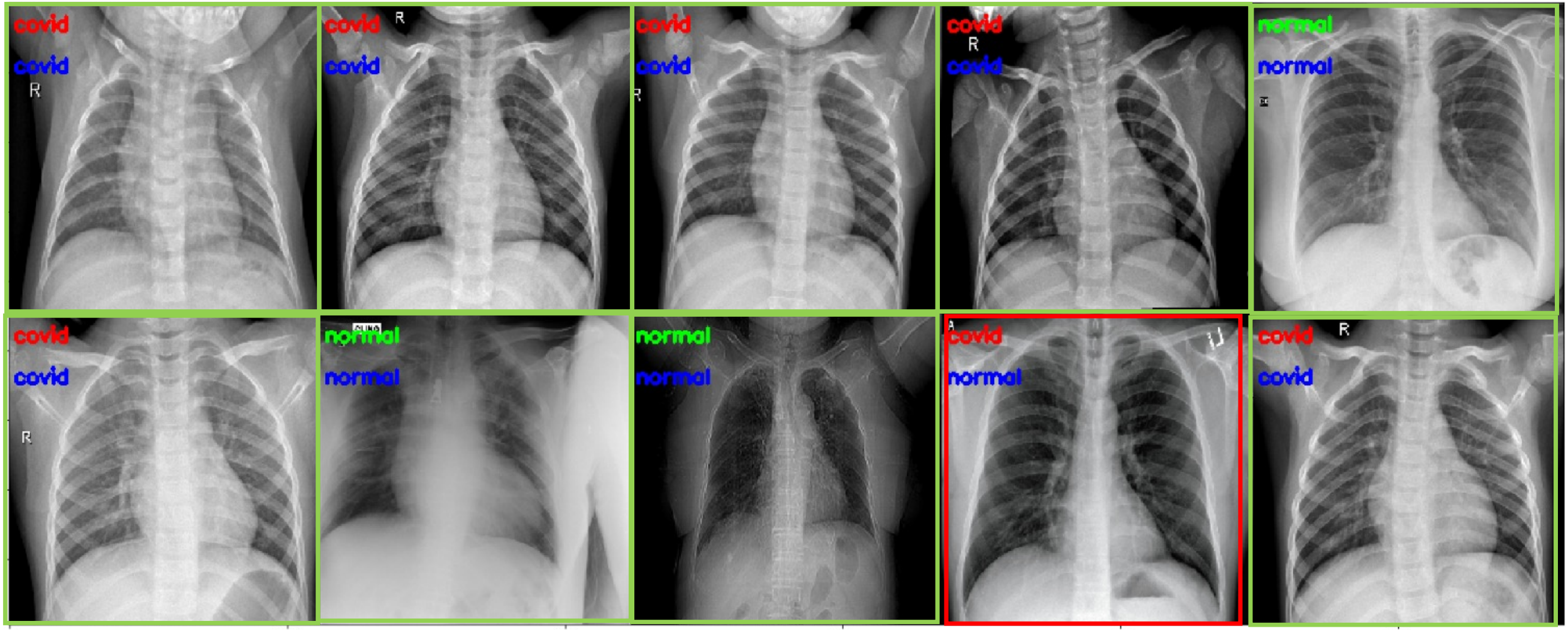
- we can use them to classify the images into **healthy** and **disease**

Pneumonia detection

True label

Estimated covid label

Estimated healthy label



Pneumonia detection

Classification results on test set

Sensitivity (recall, true positive rate)

$$\frac{\text{True Positive}}{\text{\# Actual Positive}} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} = 0.92$$

Specificity (true negative rate)

$$\frac{\text{True Negative}}{\text{\# Actual Negative}} = \frac{\text{True Negative}}{\text{False Positive} + \text{True Negative}} = 1$$

Actual class

Estimated class	1 (p)	0 (n)
1 (Y)	**True positive** **11**	**False positive** **0**
0 (N)	**False negative** **1**	**True negative** **11**

- **Accuracy**: $\approx 96\%$

Pneumonia detection

Classification results on test set

Sensitivity (recall, true positive rate)

$$\frac{\text{True Positive}}{\text{\# Actual Positive}} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} = 0.92$$

Specificity (true negative rate)

$$\frac{\text{True Negative}}{\text{\# Actual Negative}} = \frac{\text{True Negative}}{\text{False Positive} + \text{True Negative}} = 1$$

- **Sensitivity:** of patients that ***do have*** COVID-19 (i.e., *true positives*), we could accurately identify them as "COVID-19 positive" 92% of the time using our model

- **Specificity:** of patients that ***do not have*** COVID-19 (i.e., true negatives), we could accurately identify them as "COVID-19 negative" 100% of the time using our model.

Pneumonia detection

Classification results on test set

Sensitivity (recall, true positive rate)

$$\frac{\text{True Positive}}{\text{\# Actual Positive}} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} = 0.92$$

Specificity (true negative rate)

$$\frac{\text{True Negative}}{\text{\# Actual Negative}} = \frac{\text{True Negative}}{\text{False Positive} + \text{True Negative}} = 1$$

- Being able to **accurately detect healthy patients** with 100% accuracy is great. We do not want to quarantine someone for nothing

- …but **we don't want to classify someone as «healthy» when they are «COVID-19 positive»**, since it could infect other people without knowing

Summary

Balancing sensitivity and specificity is incredibly challenging when it comes to medical applications

The results should **always be validated** with another pool of people

Furthermore, we need to be **concerned of what the model is actually learning:**

- Does the results align with the medical knowledge?
- Was the dataset well representative of the population or there was selection bias?