Machine Learning

CSE 142

Xin (Eric) Wang

Wednesday, November 17, 2021

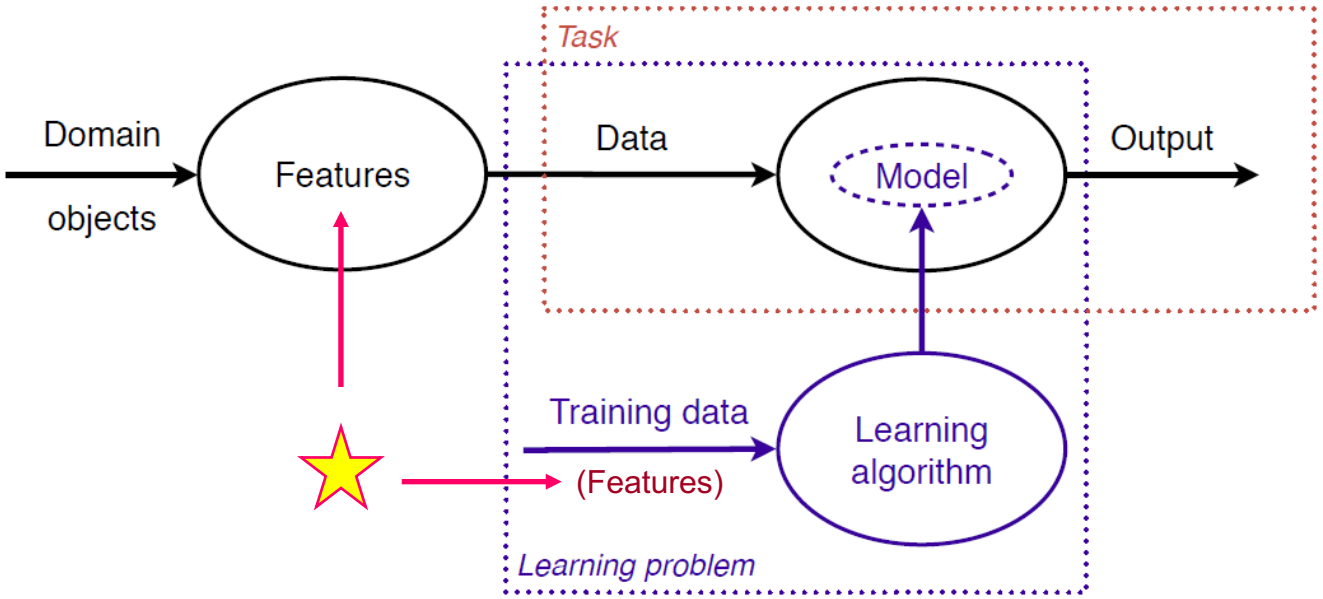
Today

- Features (Ch. 10)

Notes

- Discussion session next week: Naïve Bayes classifier
- HW#3 – extended by two days, due Friday midnight
  - Finetune your model locally on the validation set
  - Only submit your FINAL code to CodaLab (you are not supposed to exploit the test set for validation, which is cheating)
- HW#4 out Friday, due on 12/3 (Friday)

A machine learning system



A **task** requires an appropriate mapping – a **model** – from data described by **features** to outputs. Obtaining such a model from training data is what constitutes a **learning problem**.

Tasks are addressed by **models**.

Learning problems are solved by **learning algorithms** that produce **models**.

Features

- Let's think about features (aka attributes) a bit more than we have….
- What's a feature? A mapping from the instance space **X** to the feature domain **F**

$$f_i : \mathbf{X} \rightarrow \mathbf{F}_i$$

- The model can be thought of as just a new feature – a particular combination of the input features, constructed to solve the task at hand
  - E.g., the $\boldsymbol{w^T x}$ value in a linear classifier or SVM
- There are different categories of features and of permissible operations on features

Main feature types

- Quantitative features
  - Measured on a meaningful numeric scale
  - Domain is often real values
  - E.g.: weight, height, age, angle, match to template, …
- Ordinal features
  - The relevant information is the ordering, not the scale
  - Domain is an ordered set
  - E.g., rank, street addresses, preference, ratings, …
- Categorical features
  - No scale or order information
  - Domain is an unordered set
  - E.g., colors, names, parts of speech, binary attributes, …

Main feature types

Kinds of features, their properties, and allowable statistics:

Kind	*Order*	*Scale*	*Tendency*	*Dispersion*	*Shape*
Categorical	×	×	mode	n/a	n/a
Ordinal	√	×	median	quantiles	n/a
Quantitative	√	√	mean	range, interquartile range, variance, standard deviation	skewness, kurtosis

Statistics – calculations on the features

Mode – the value that occurs most frequently
Median – the middle value in an ordered list
Mean (expected value) – the arithmetic average of the values

{ 0, 0, 0, 1, 2, 9, 1000 } ⇨ Mode = 0
Median = 1
Mean = 144.57

Main feature types

Kinds of features, their properties, and allowable statistics:

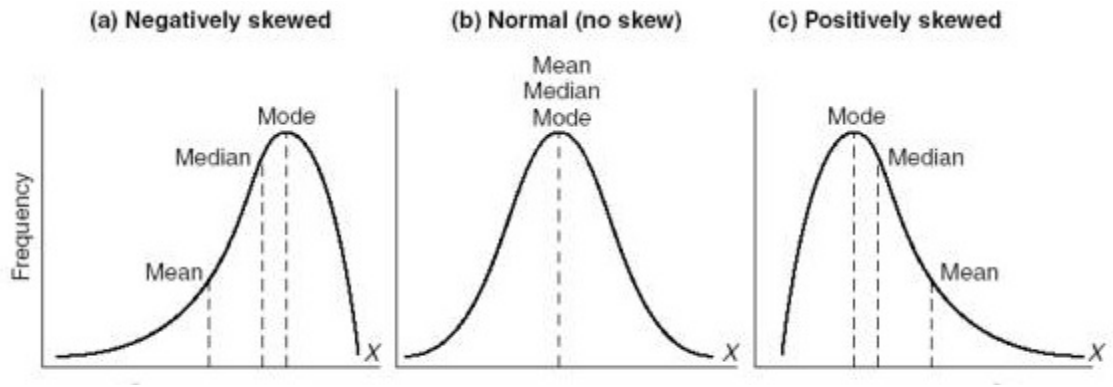
Kind	*Order*	*Scale*	*Tendency*	*Dispersion*	*Shape*
Categorical	×	×	mode	n/a	n/a
Ordinal	√	×	median	quantiles	n/a
Quantitative	√	√	mean	range, interquartile range, variance, standard deviation	skewness, kurtosis

Statistics – calculations on the features

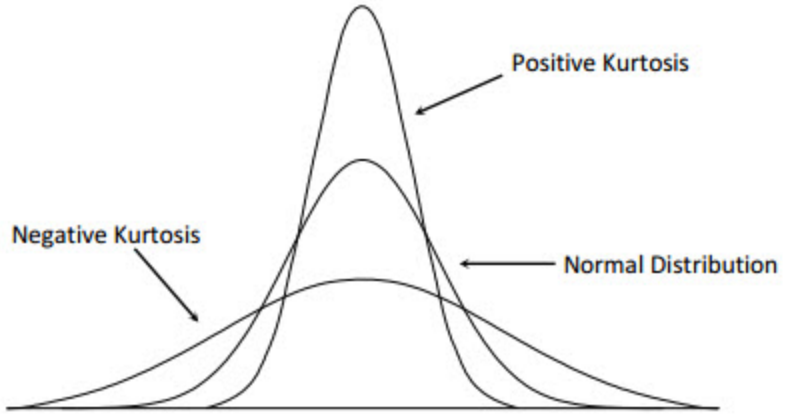
Range = (max. value – min. value)
Standard deviation (σ) = Square root of the variance (σ^2)
Skewness $\propto$ 3rd moment (lack of symmetry: right or left skewed)
Kurtosis $\propto$ 4th moment (tailedness of the probability distribution)

Skewness and kurtosis

Skewness:



Kurtosis:



Main feature types

Kinds of features, their properties, and allowable statistics:

Kind	*Order*	*Scale*	*Tendency*	*Dispersion*	*Shape*
Categorical	×	×	mode	n/a	n/a
Ordinal	√	×	median	quantiles	n/a
Quantitative	√	√	mean	range, interquartile range, variance, standard deviation	skewness, kurtosis

Statistics – calculations on the features

Percentiles: p^{th} percentile = the value such that p percent of the instances fall below it

Deciles – multiples of 10 percentiles (10^{th}, 20^{th}, etc.)

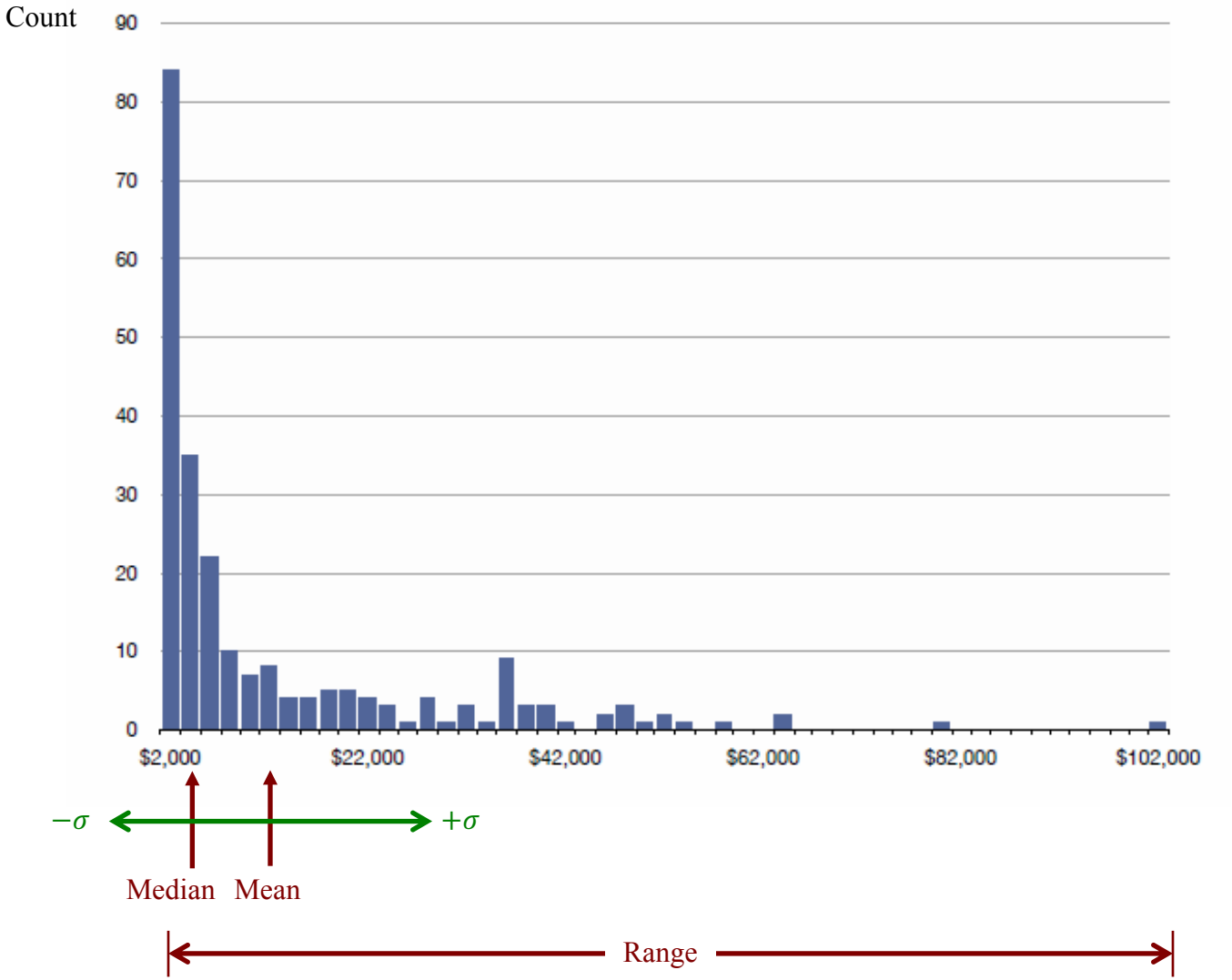
Quartiles – multiples of 25 percentiles (25^{th}, 50^{th}, etc.)

Interquartile range – the difference between the first and third quartiles (75^{th} and 25^{th} percentiles)

Histogram

Shows data as a frequency plot, with data collected in *bins*

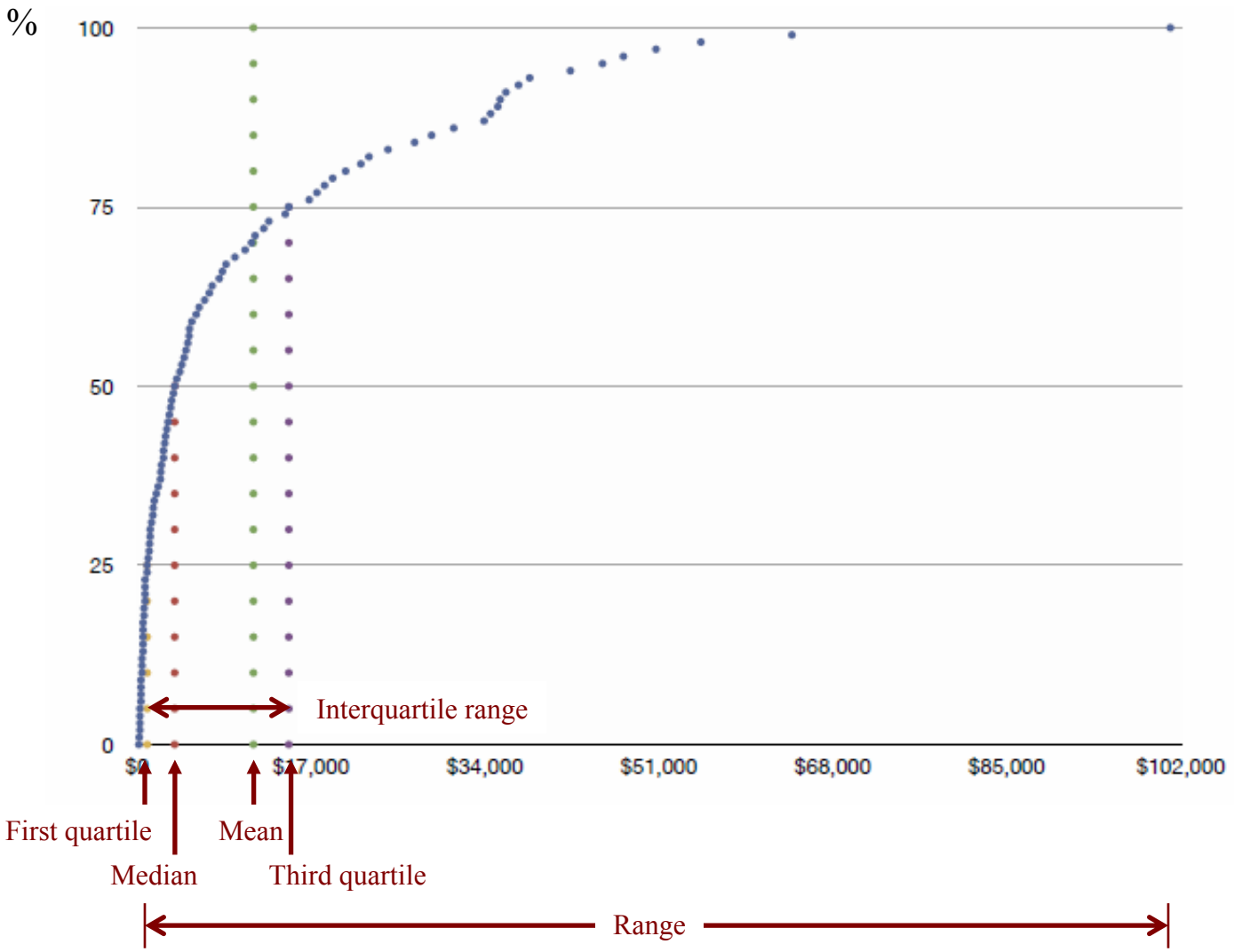
A histogram of GDP per capita for 231 countries (bin size = $2000):



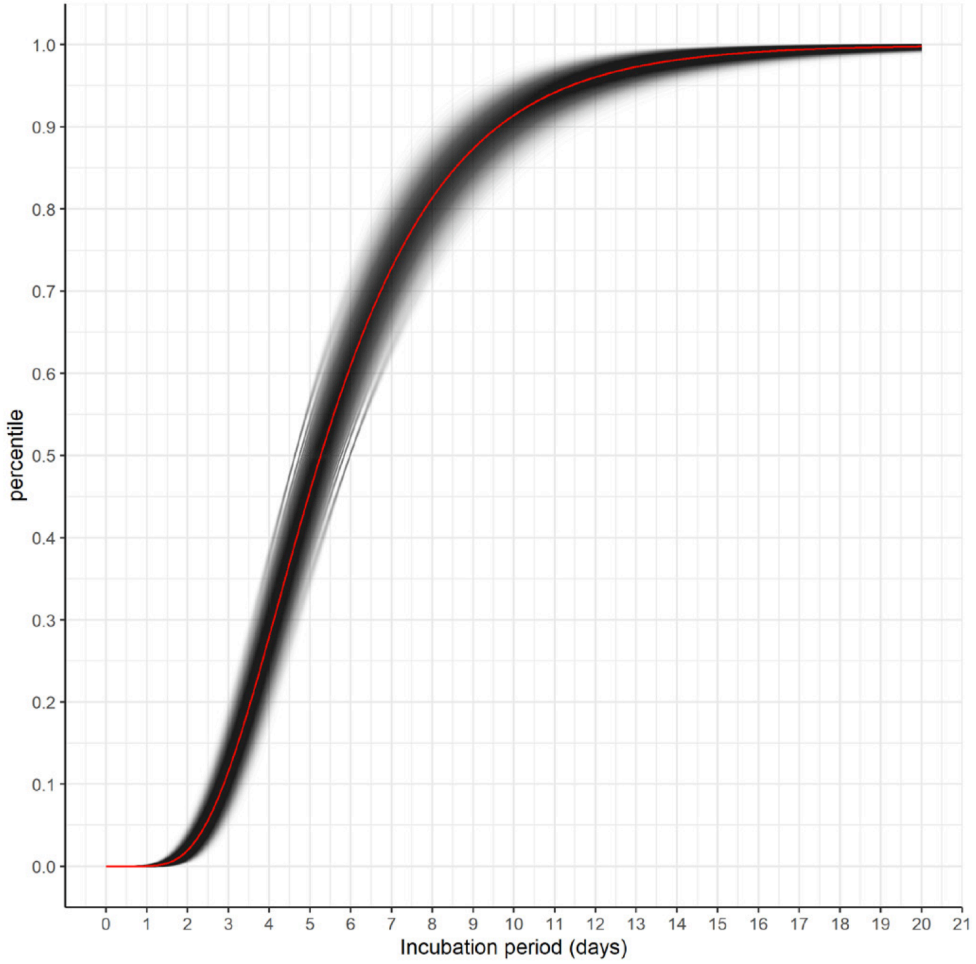
Percentile plot

Shows cumulative fraction of data – what % fall below a given value

A percentile plot of GDP per capita for 231 countries:



Percentile plot: incubation period of COVID-19



https://bmjopen.bmj.com/content/bmjopen/10/8/e039652.full.pdf

Feature construction and selection

- New features can be constructed from the given feature set
  - E.g., kernel methods do this, sometimes for great benefit
- Features may be combined in various ways, e.g.:
  - N-grams – groups of N consecutive words (tokens)
    - E.g., trigrams: "my first teacher," "my first job," "my first the"
  - Cartesian products of categorical features
    - E.g., Shapes × Colors = { blue circle, red square, green line, … }
  - $\boldsymbol{w}$ vector for a perceptron (combining training data point features)
  - Kernel function κ (e.g., 2D to 3D transformation)
- Typical approach: Generate new features, then select a suitable subset prior to learning
  - How to determine/evaluate suitability? How many to keep?
    - Many different approaches to this problem….

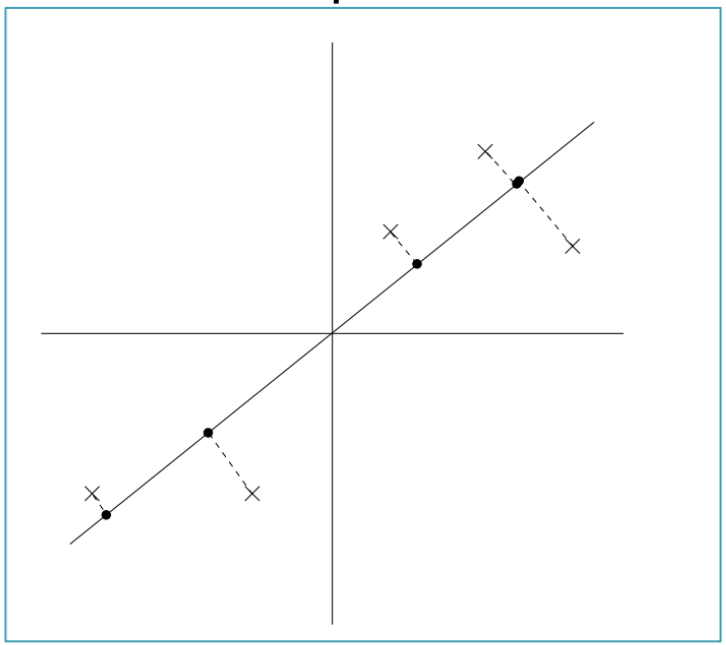
Principle Component Analysis (PCA)

- One of the most widely used feature construction/selection techniques is Principal Component Analysis (PCA)
  - PCA constructs new features that are linear combinations of given features
- Computed eigenvectors and eigenvalues hold useful information
- Often used for dimensionality reduction, finding the intrinsic linear structure in the data

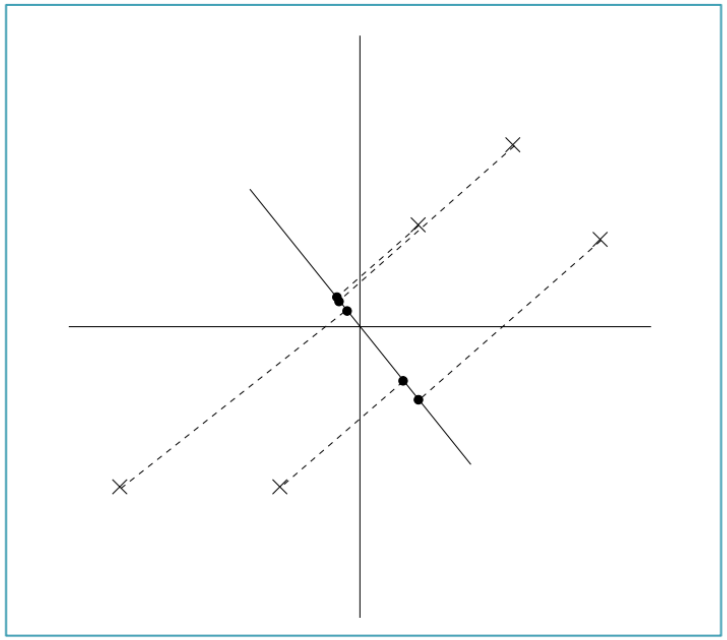
Quiz: maximizing the variance

- Consider the two projections below, which maximizes the variance?

Option A

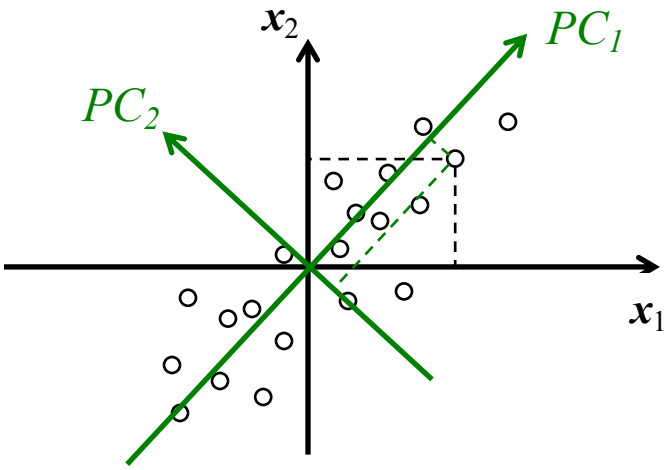


Option B



Principle Component Analysis (PCA)

- One of the most widely used feature construction/selection techniques is Principal Component Analysis (PCA)
  - PCA constructs new features that are linear combinations of given features
- Computed eigenvectors and eigenvalues hold useful information
- Often used for dimensionality reduction, finding the intrinsic linear structure in the data



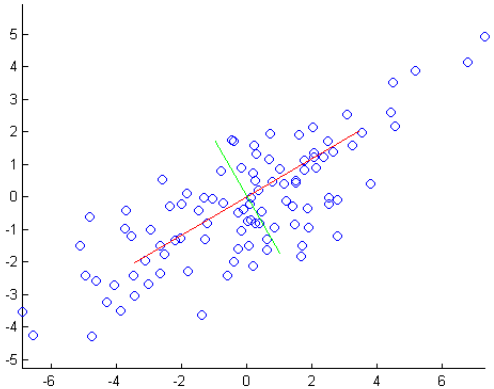
Given features 1 and 2 (x_1, x_2)
Computed features 1 and 2 (green axes)

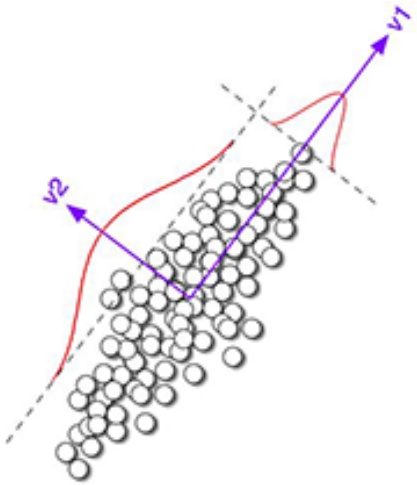
$$\mathrm{PC}_1 = \underset{y}{\operatorname{argmax}}(y^T X)(X^T y)$$

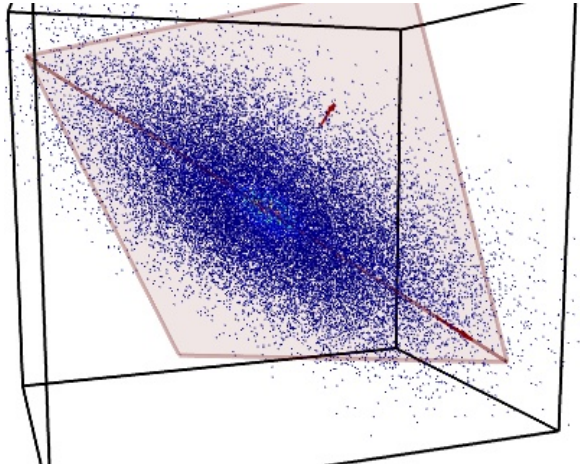
(maximize variance of points projected onto unit vector y)

PCA

- The first principal component is the direction of maximum (1D) variance in the data
- The second principal component is the direction of maximum variance orthogonal to the first PC
  - Etc. for additional PCs
- For N-dimensional data, there can be N principal components
  - But perhaps only *k* of them are useful ($k < N$) – dimensionality reduction!
- PCA typically assumes zero-mean data
  - First subtract the mean from each data point; centroid is thus (0, 0)







PCA and eigendecomposition

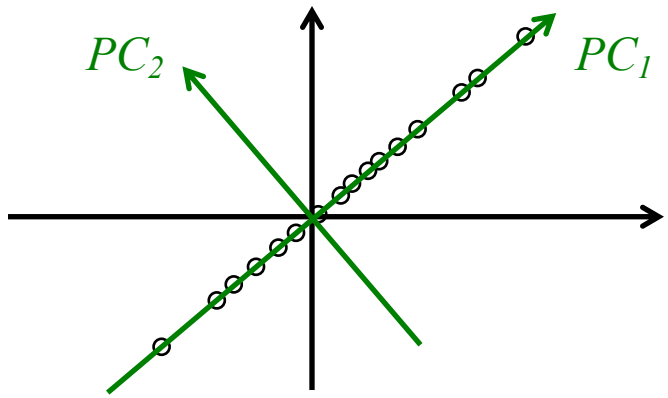
- For *n* points of dimension *d*, let $X = [x_1\ x_2\ \ldots\ x_n]$ $(d \times n)$
- The scatter matrix, $\boldsymbol{S} = \boldsymbol{X_z X_z}^T$ $(d \times d)$, measures variance
- The eigenvectors of S are the principal components of the data, ordered by decreasing eigenvalues

$$Su_i = \lambda_i u_i \rightarrow SU = U\Lambda \equiv S = U\Lambda U^T \text{ (eigendecomposition)}$$

where U is the matrix of eigenvectors (columns of U) and Λ is a diagonal matrix of eigenvalues $\text{diag}(\lambda_i, \lambda_2, \ldots, \lambda_d)$

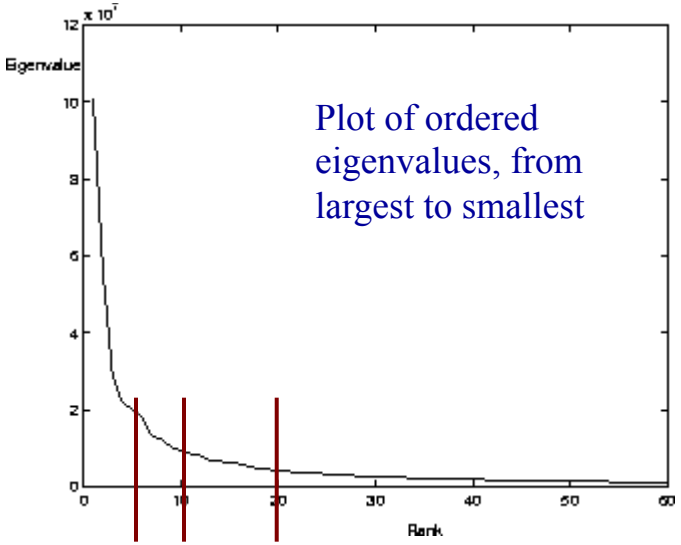
  - The eigenvector u_i associated with the largest eigenvalue λ_i is the first principal component

- If rank(S) < d, then some eigenvalues will be zero



PCA for dimensionality reduction

- So the eigenvalues can give clues to the intrinsic dimensionality of the data, or at least provide a way to more efficiently approximate high-dimensional data with lower-dimensional feature vectors
- For example:



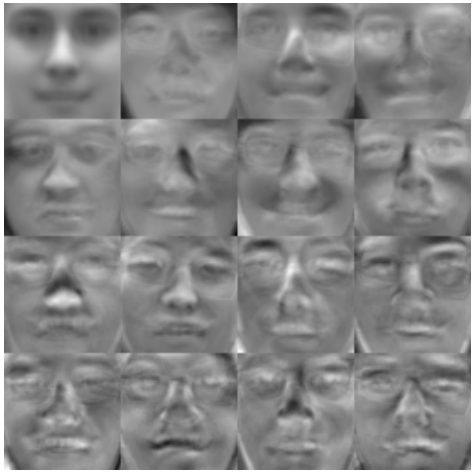
60-dimensional data (60 eigenvectors and eigenvalues)

Many of the eigenvalues are small, meaning that their associated eigenvectors don't contribute much to the representation of the data

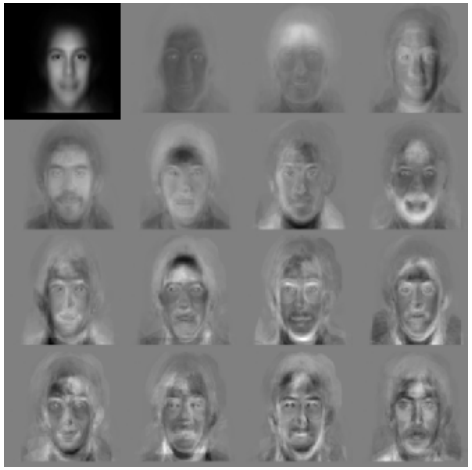
We can choose a cutoff – say, only use the first 20 eigenvectors (or 10, or 5)

Face recognition via "Eigenfaces"

- A well-known technique for face recognition based on computing eigenvectors of a training set of face images, i.e., Eigenfaces



Eigenfaces 1

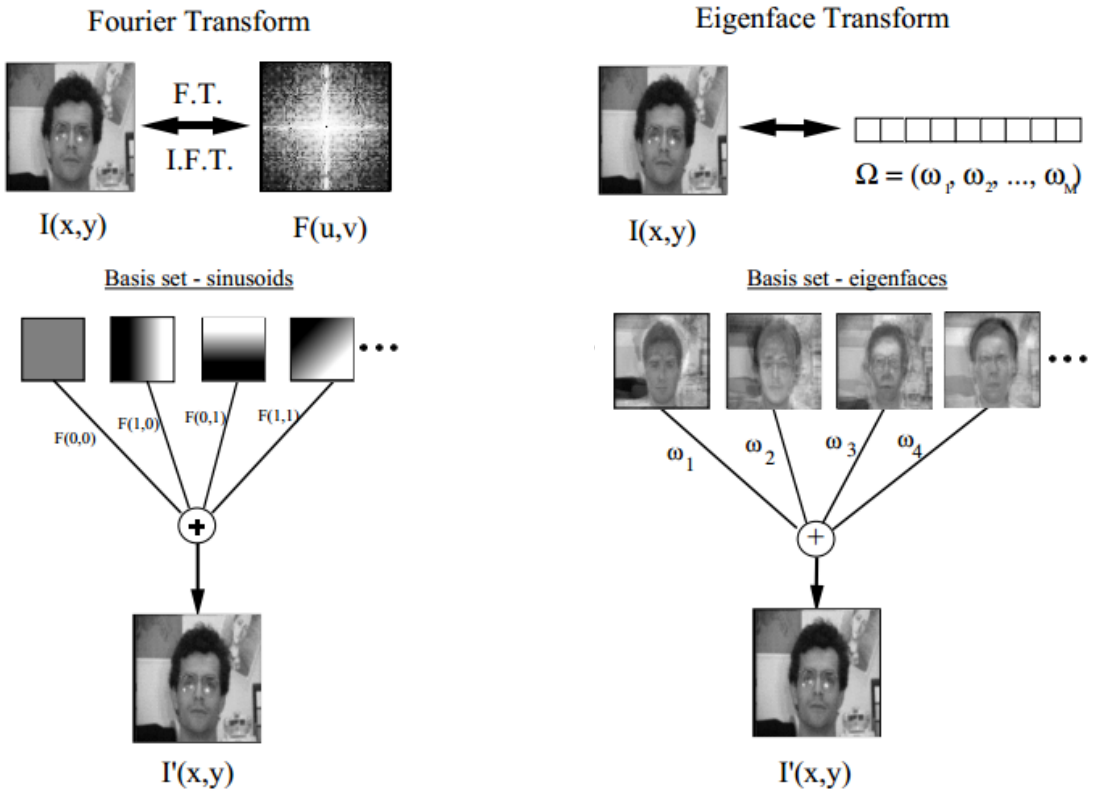


Eigenfaces 2

Keep in mind: an image is just an N-dimensional point or vector (where N = rows × cols)

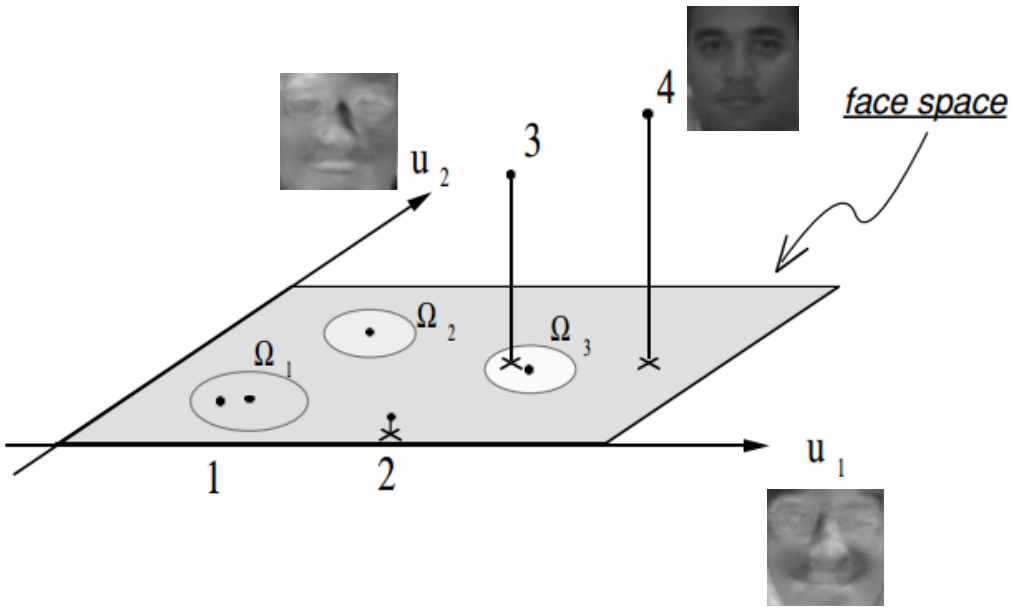
Face recognition via "Eigenfaces"

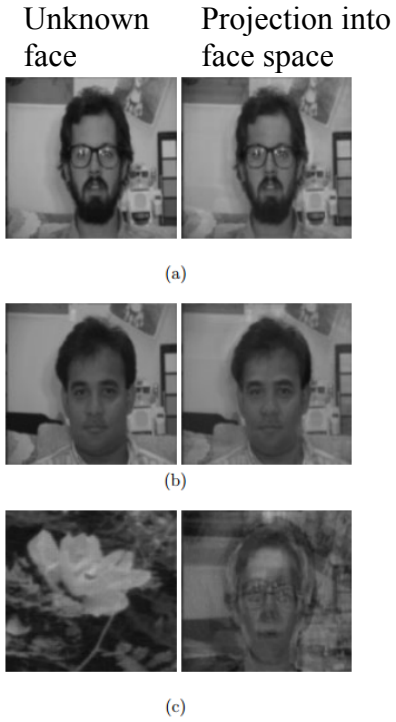
- Eigenvectors (eigenfaces) can be thought of as *basis vectors* for reconstructing data (face images)



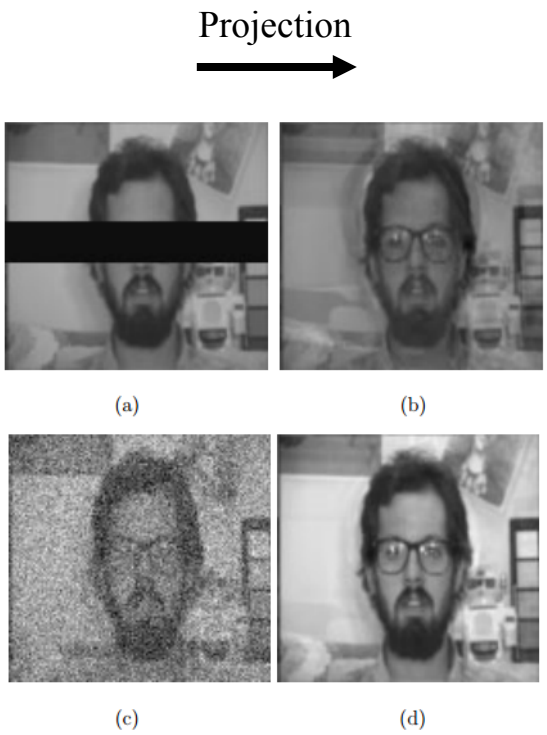
Face recognition via "Eigenfaces"

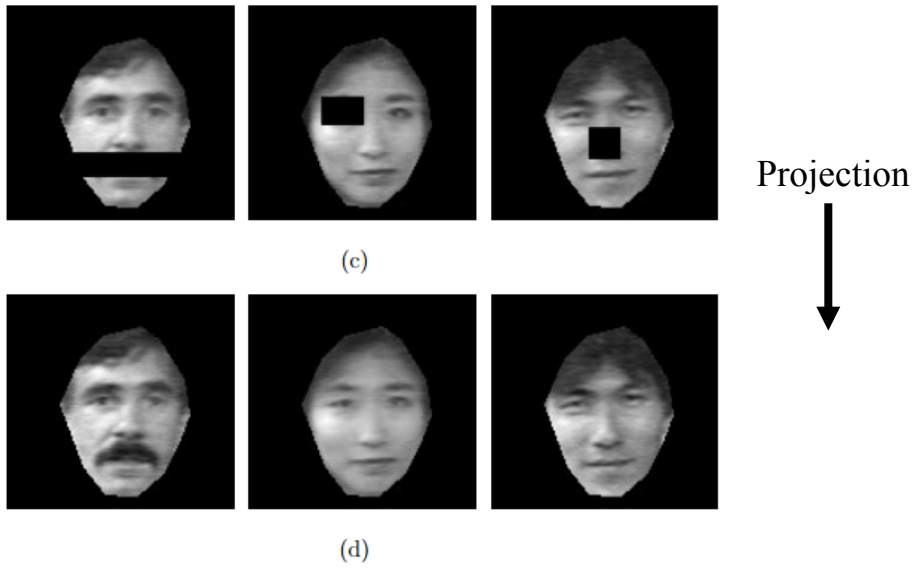
- The Eigenfaces span a (relatively) low-dimensional *face space*, representing all possible face images
- A new (unknown) face image is projected into the face space (reconstructed by the Eigenfaces)
  - The distance between the face image and its reconstruction is the *distance from face space*





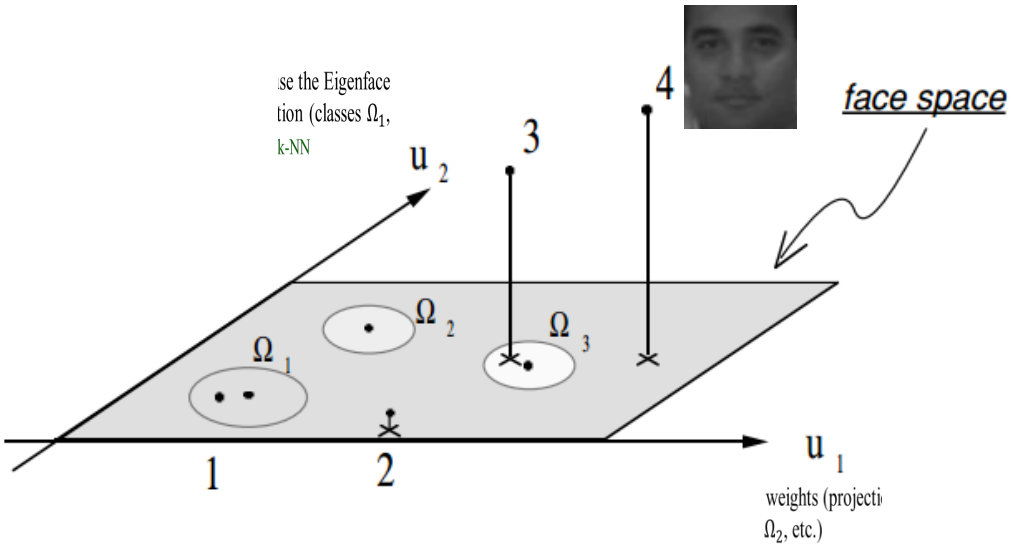
Face recognition via “Eigenfaces”

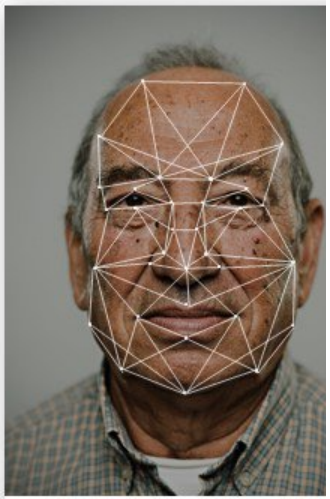


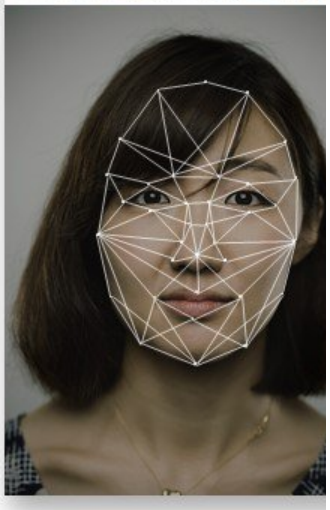


Face recognition via "Eigenfaces"

- The *distance from face space* measure could be used for face detection: Does this image (or part of an image) look like a face?
- If yes, then use the Eigenface weights (projections) as features for classification (classes Ω_1, Ω_2, etc.)
  - E.g., using k-NN







San Francisco Bans Facial Recognition Technology

Attendees interacting with a facial recognition demonstration at this year's CES in Las Vegas. Joe Buglewicz for The New York Times

By Kate Conger, Richard Fausset and Serge F. Kovaleski

May 14, 2019

Model ensembles

Chapter 11 in the textbook

Model ensembles

- We've seen how combining features can be beneficial
- We can also combine models to increase performance
  - Combinations of models are known as model ensembles
  - Potential for better performance at the cost of increased complexity
- General approach to model ensembles:
  - Construct multiple different models from adapted versions of the training data (e.g., reweighted or resampled)
  - Combine the predictions of these models in some way (averaging, voting, weighted voting, etc.)
- Two of the best-known ensemble methods are bagging and boosting
- These are "meta" methods – i.e., they are independent of the particular learning method (linear classifier, SVM, etc.)

Bagging

- Bagging = "bootstrap aggregation"
  - Create T models on different random samples of the training data set
  - Each sample is a set of training data called a bootstrap sample
    - Could be any size – often $|D|$ is used, the size of the training set
- Bootstrap samples: Sample the data set with replacement (i.e., a data point can be chosen more than once in a bootstrap sample)
  - For $j = 1$ to $|D|$, choose with uniform probability from training data points $D = \{ d_1, d_2, \ldots, d_{|D|} \}$
  - Gather these into the bootstrap sample D_i
- Use $\{ D_1, D_2, \ldots, D_T \}$ to train models $\{ M_1, M_2, \ldots, M_T \}$, then combine the predictions of the T models
- Differences between the T bootstrap samples create diversity among the models in the ensemble

Bagging

Algorithm Bagging($D, T, \mathscr{A}$) – train an ensemble of models from bootstrap samples.

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.

Output : ensemble of models whose predictions are to be combined by voting or averaging.

for $t = 1$ to T **do**

  build a bootstrap sample D_t from D by sampling $|D|$ data points with replacement;

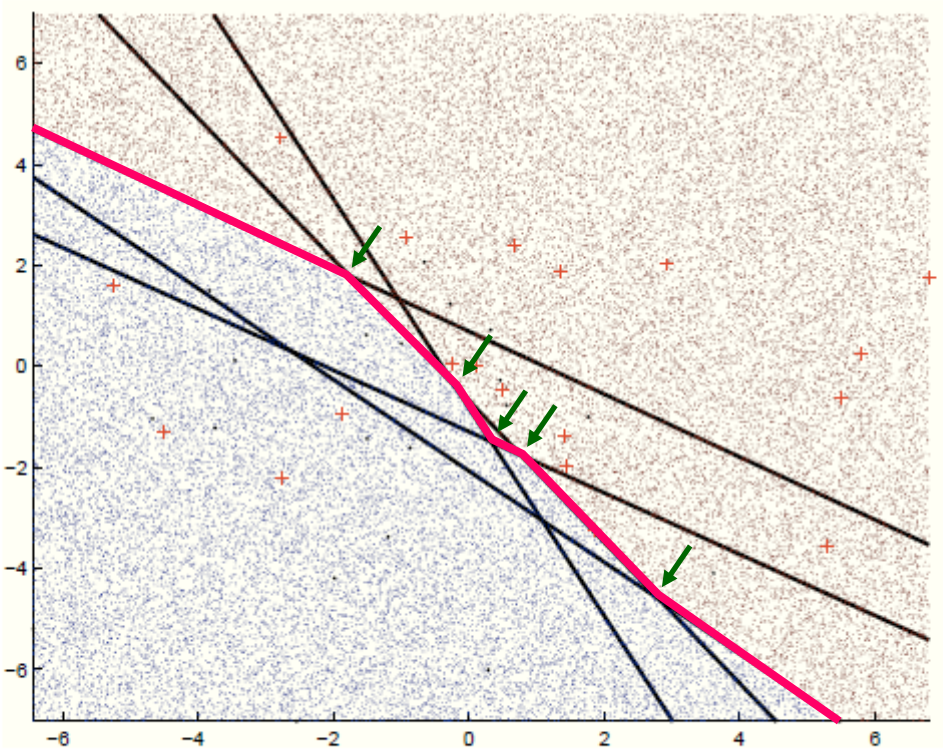
  run $\mathscr{A}$ on D_t to produce a model M_t;

end

return $\{M_t | 1 \leq t \leq T\}$

Bagging example 1

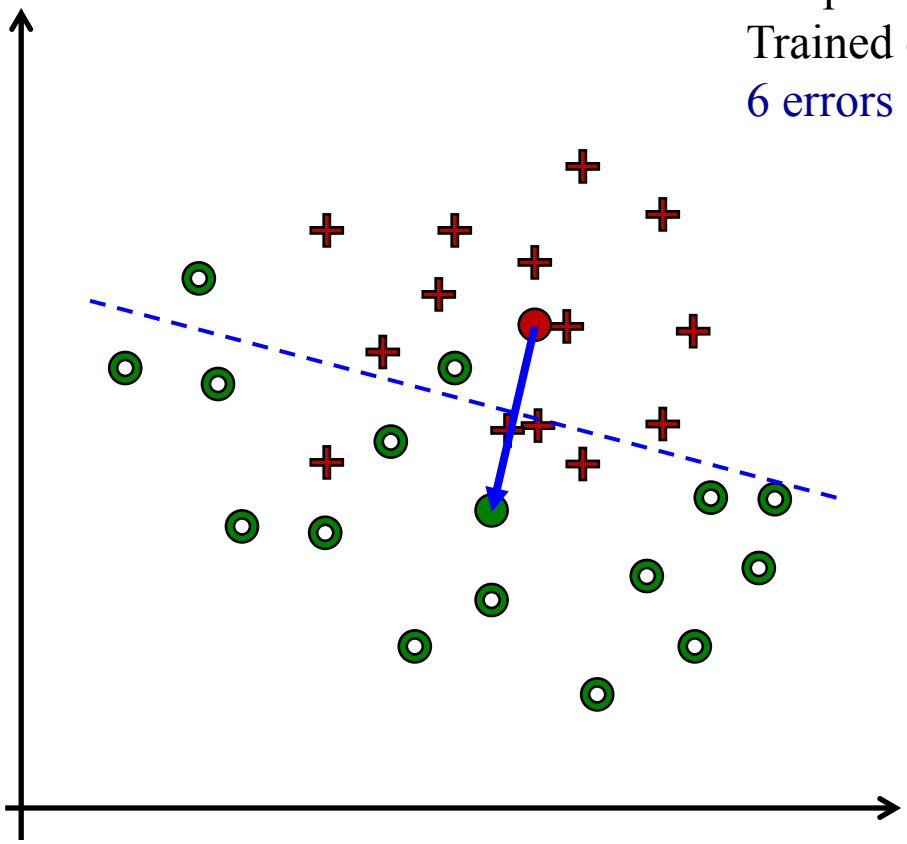
- Train 5 binary linear classifiers by bagging
- Use majority vote (for example) to determine classification output for a new $\boldsymbol{x}$
- The decision boundary is piecewise linear



Bagging example 2

Simple linear classifier
Trained on all data
6 errors

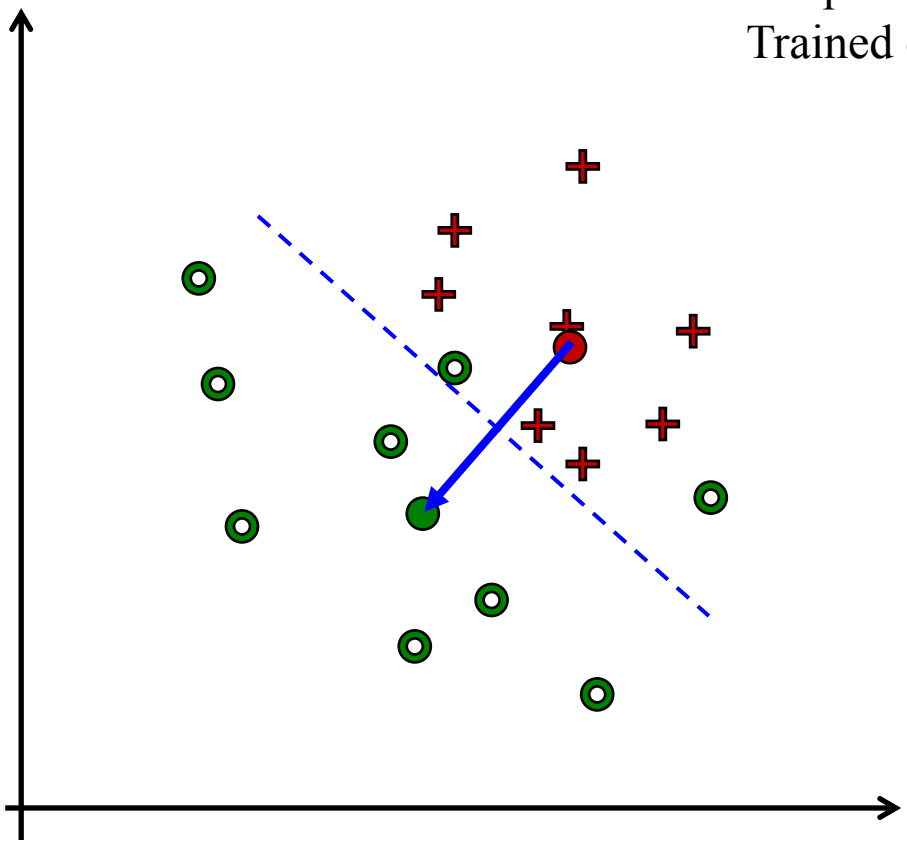
|D| = 29



Bagging example 2

Simple linear classifier
Trained on bootstrap sample set 1

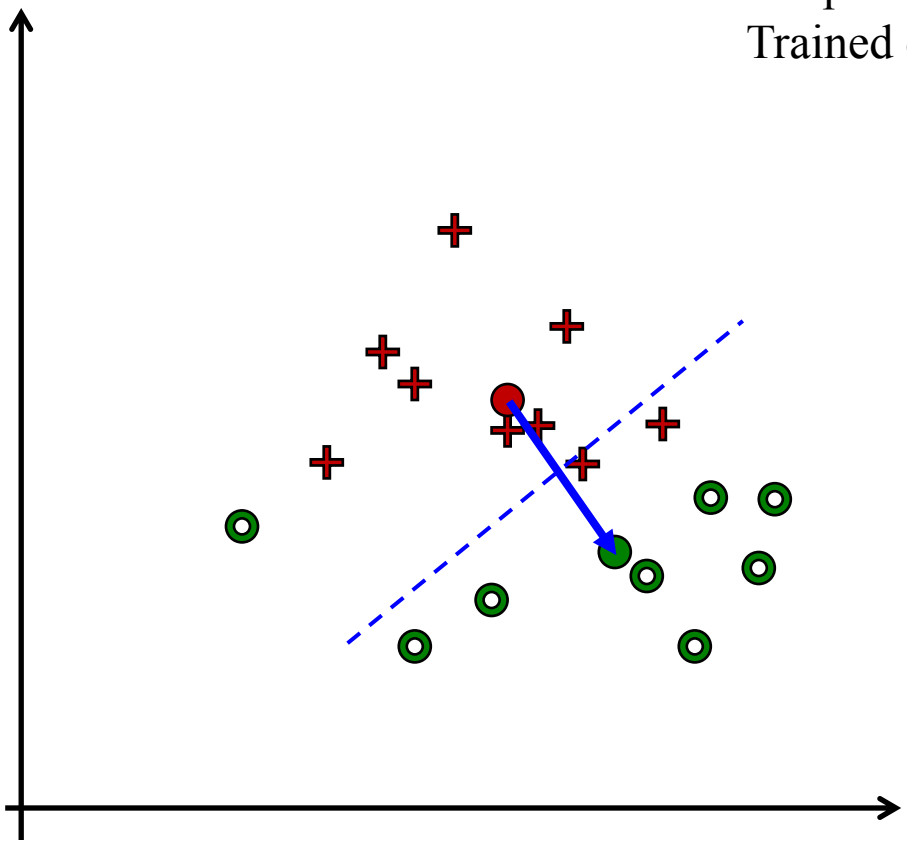
$|D_1| = 17$



Bagging example 2

Simple linear classifier
Trained on bootstrap sample set 2

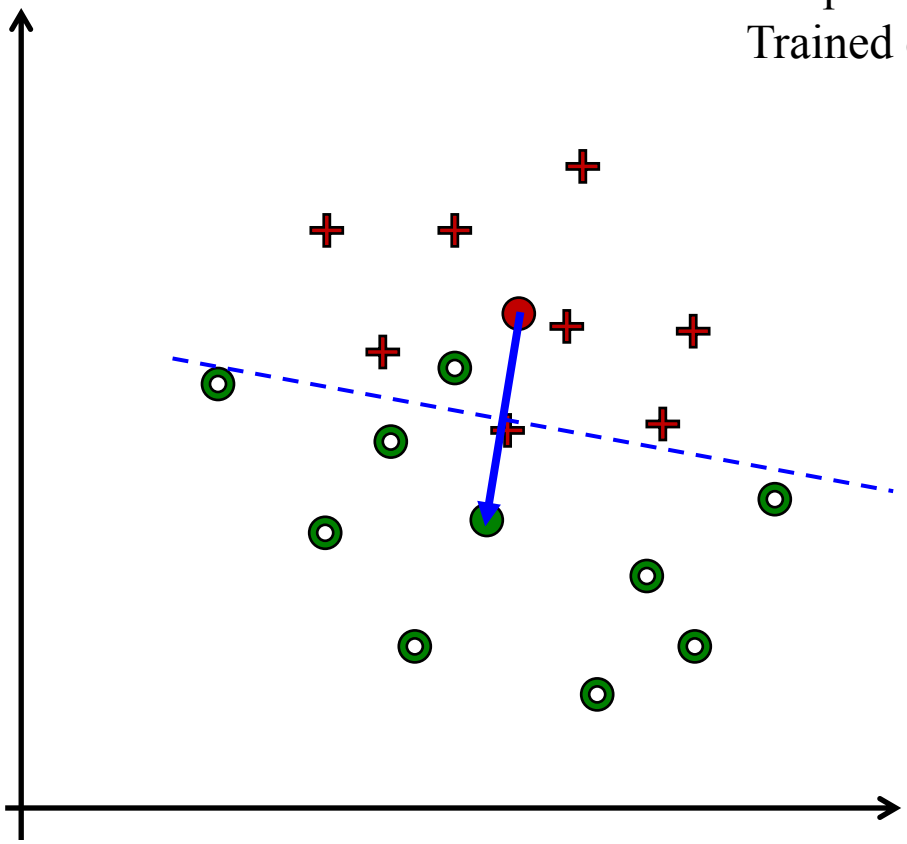
$|D_2| = 17$



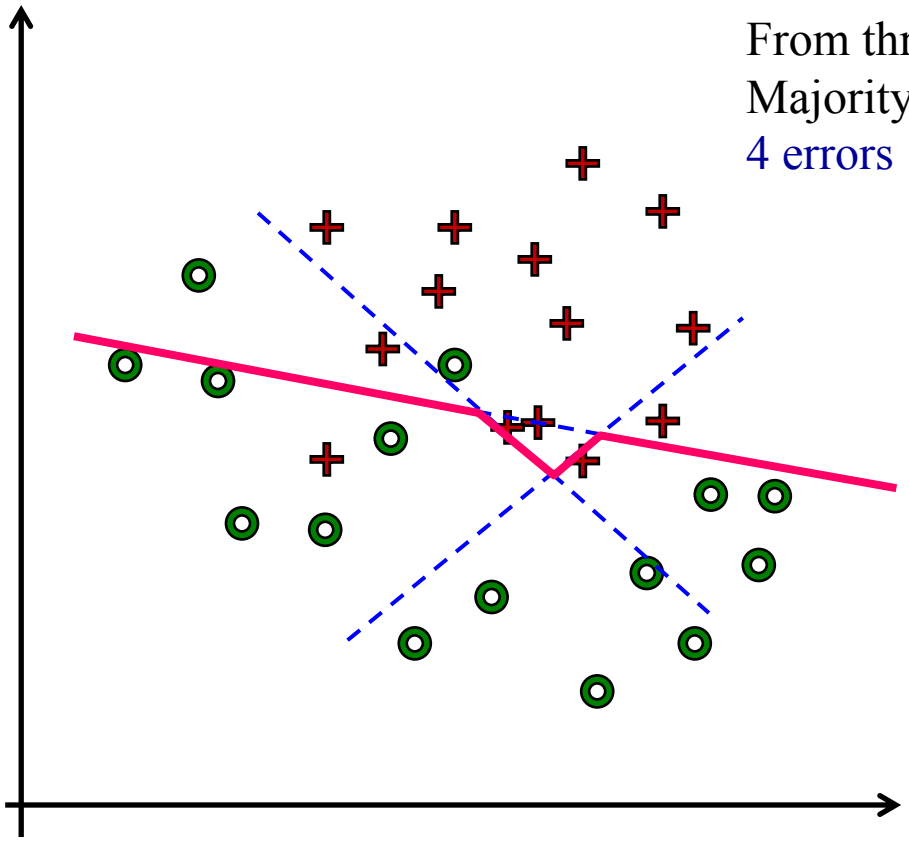
Bagging example 2

Simple linear classifier
Trained on bootstrap sample set 3

$|D_3| = 17$



Bagging example 2



Ensemble linear classifier
From three bootstrap data sets
Majority vote
4 errors

Bagging with tree models

- Bagging is particularly useful for tree models (e.g., decision trees), which can be very sensitive to variations in the training data
  - E.g., build *T* decision trees, produce *T* output labels for an input, then vote to determine the classifier output

- Subspace sampling: Build each decision tree from a different random subset of the **features**

A bunch of trees

Bagging + subspace sampling = *random forests* method

Random forest

Algorithm RandomForest(D, T, d) – train an ensemble of tree models from bootstrap samples and random subspaces.

Input : data set D; ensemble size T; subspace dimension d.

Output : ensemble of tree models whose predictions are to be combined by voting or averaging.

for $t = 1$ to T **do**

  build a bootstrap sample D_t from D by sampling $|D|$ data points with replacement;

  select d features at random and reduce dimensionality of D_t accordingly;

  train a tree model M_t on D_t without pruning;

end

return $\{M_t | 1 \leq t \leq T\}$

Quiz: model ensemble

- See Canvas.

Next

- Boosting (3/2)
- Machine learning experiments (3/2)
- Neural network & deep learning (3/4 - end)