SUPERVISED LEARNING

SLIDES ADAPTED (AND EXTENDED) FROM ETHEM ALPAYDIN © THE MIT PRESS, 2004

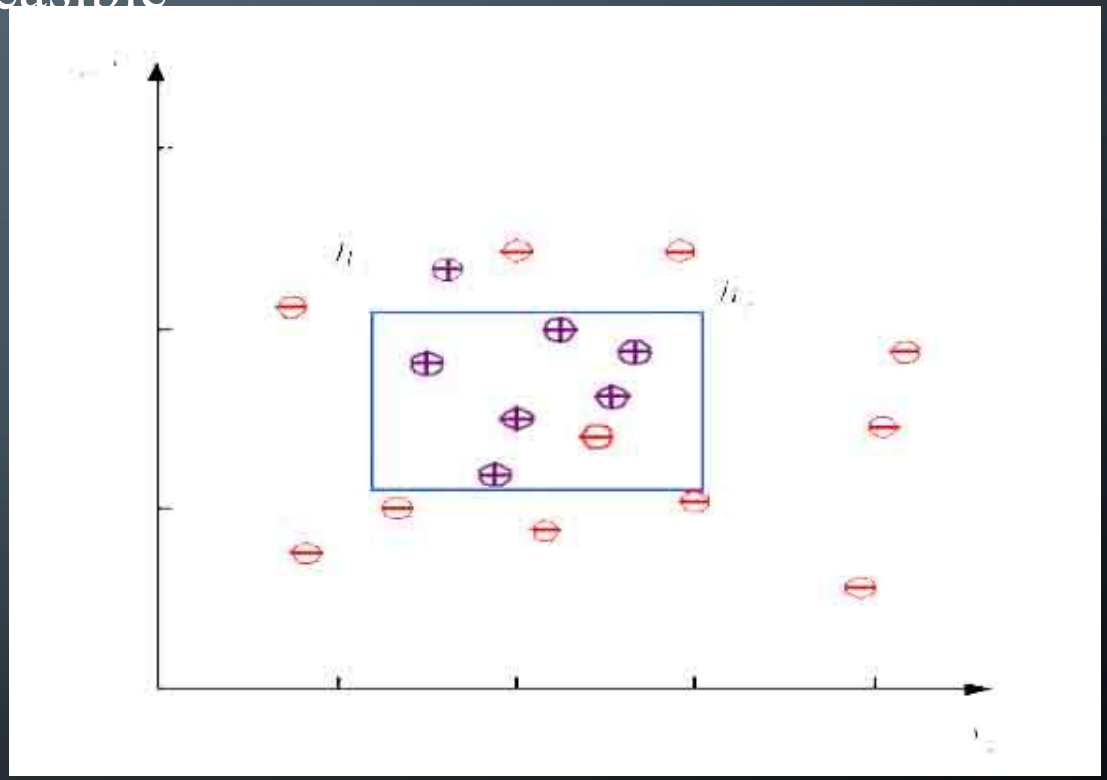
SYLLABUS

Probably Approximately Correct Learning (PAC), Noise, Learning Multiple classes, Model Selection and Generalization, Dimensionality reduction- Subset selection, Principle Component Analysis

NOISE

- Any unwanted anomaly
- Difficulty to learn the class, zero error infeasible with simple hypothesis class
- Noise may be
  - Imprecision in recording
  - Errors in labeling data points- also called teacher's noise
  - Additional/Hidden attributes

  which was not considered
- Consequences
  - No h on *H* may be consistent !



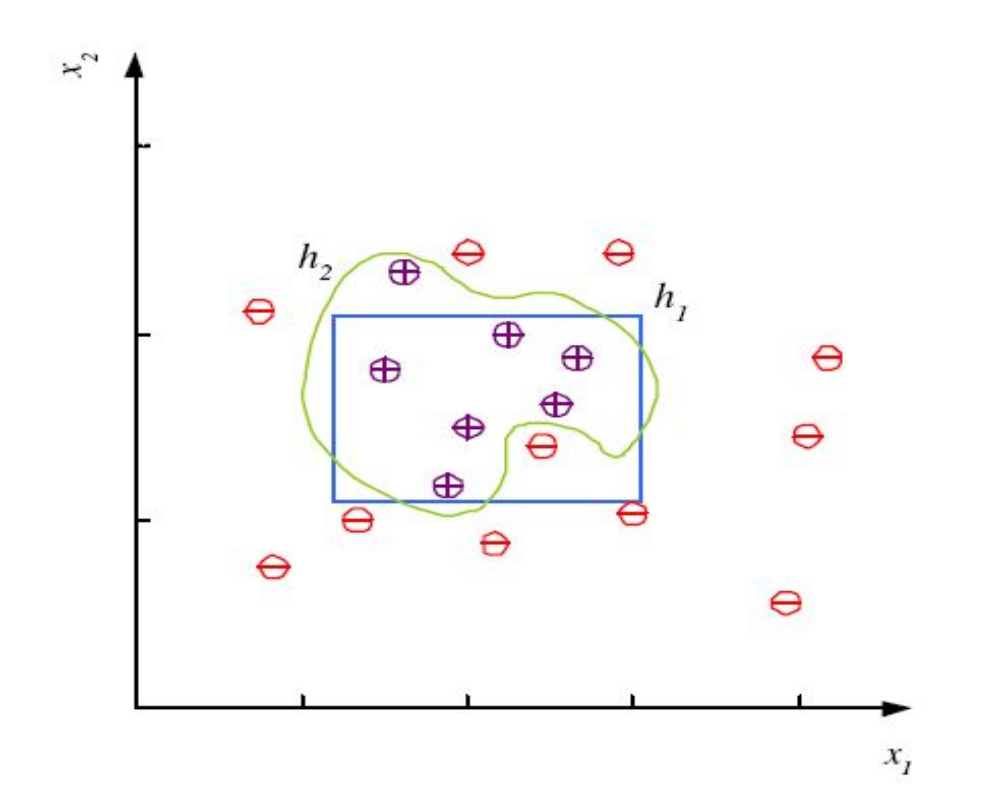
- When there is noise, a simple boundary could not separate + and – ve instances
- To separate them, a complicated hypothesis that corresponds to a hypothesis class with larger capacity is required
- Rectangle- defined by 4 numbers
- Complicated shape- larger number of parameters required
- 2 possibilities
  - Complex model- which give perfect fit to data and attain zero error
  - Simple model- with some error

NOISE AND MODEL COMPLEXITY

Use the simpler one because

- Simpler to use (lower computational complexity)
- Easier to train (lower space complexity)
- Easier to explain (more interpretable)
- Generalizes better (lower variance) –

Occam's razor-*Simpler explanations are more plausible and unnecessary complexity should be shaved off*



PROBABLY APPROXIMATELY CORRECT (PAC) LEARNING

- PAC is a framework for mathematical analysis of machine learning algorithms.
  - It was proposed in 1984 by Leslie Valiant.
- the learner (that is, the algorithm)
  - receives samples and
  - must select a hypothesis from a certain class of hypotheses (Hypothesis Class)
- We want
  - our hypothesis to be correct always, if not *approximately correct*
  - to be *confident* in our hypothesis ie hypothesis will be correct most of the time- probability is associated with it

PROBABLY APPROXIMATELY CORRECT (PAC) LEARNING

The goal is that:

- with high probability (the "probably" part), the selected hypothesis will have low generalization error (the "approximately correct" part)
- Any (efficient) algorithm that returns hypothesis that is Probably Approximately Correct is called a PAC-learning algorithm

PROBABLY APPROXIMATELY CORRECT (PAC) LEARNING

Notations

- X-Domain of possible examples/ instance space
- C-Class of possible concepts to label X

  family of functions c:X -> {0,1}

  - concept c is a member of C
  - can also be thought of as a subset of X
- If C is a subset of X, it defines a unique function μ_c : X -> {0 , 1} as follows:

  μ_c (x) = 1 if x is a subset of C; 0 otherwise
- H-Set of hypothesis considered by the learner
- h is a hypothesis – fn h: X -> {0 , 1}
- δ,ε –correctness bounds

INFORMAL DEFINITION

- Let X be an instance space, C a concept class for X, h a hypothesis in C and F an arbitrary, but fixed, probability distribution.
- The concept class C is said to be PAC-learnable if there is an

  algorithm A which, for samples drawn with **any probability distribution F** and **any concept c ϵ C,** will **with high probability** produce a **hypothesis h ϵ C** whose **error is small.**

NOTATIONS

1. True Error

- **true error** of a hypothesis h with respect to a target concept c

$$error_F(h) = P_{x \in F}(h(x) \neq c(x))$$

Probability of x drawn from X with probablity distribution F

- Probability that h will misclassify an instance x
- not directly observable to the learner
- learner can only see the **training error** of each hypothesis

2. Length or dimension of an instance

- length or dimension or size of an instance in the instance space X
  - instance space X is the **n-dimensional Euclidean space**, then each example is **specified by n real numbers**, so the length of the examples is taken as **n**
  - if X is the space of the conjunctions of **n Boolean literals**, then the length may be taken as **n**

3. Size of a concept

- For any concept c, size(c) is the size of the smallest representation of c using some finite alphabet

FORMAL DEFINITION

Consider a concept class C defined over a **set of instances X** of length ***n*** ***and a learner L using hypothesis space H.***

C is PAC-learnable by L using H if for all c **E C, distributions *F over X,***

ε such that 0 < ε < 1/2, and δ ***such that*** 0 < δ < 1/2,

learner L will with probability at least (1 - δ) output a hypothesis h ***ϵ H such that error$_F$(h) <= ε, in time that is polynomial in 1/*** ε , ***1/*** δ, ***n, and*** size(c).

Figuring out if algorithm is PAC- learnable:

- As long as our learning algorithm requires
  - no more than polynomial computation per training example, and
  - no more than a polynomial number of training examples, then
  - the total computation required will be polynomial as well

APPLICATION

- The application is the derivation of a mathematical expression to estimate the size of samples (m) that would produce a hypothesis with a given high probability and which has a generalization error of given low probability.

- We use the following assumptions and notations:
  - assume that the hypothesis space H is finite. Let |H| denote the number of elements in H
  - assume that the concept class C be equal to H
  - Let m be the number of elements in the set of samples
  - Let ε and δ be such that $0 < \varepsilon, \delta < 1$
  - The algorithm can be any consistent algorithm, that is, any algorithm which correctly classifies the training examples

APPLICATION

How many training examples *m* should we have, such that with *probability at least* $1 - \delta$, *h* has error at most ε ?

It can be shown that, if m is chosen such that

$$m \geq \frac{1}{\epsilon}(\ln|H| + \ln(1/\delta))$$

then any consistent algorithm will successfully produce any concept in H with probability $(1 - \delta)$ and with an error having a maximum probability of ε.

EXAMPLE 1

Consider the class C of target concepts described by conjunctions of Boolean literals.

- A Boolean literal is any Boolean variable (e.g., Old), or its negation (e.g., -Old)
- conjunctions of Boolean literals include target concepts such as ‘Old Λ –Tall’

Is C PAC-learnable?

- show that any consistent learner will require only a polynomial number of training examples to learn any c in C
- Solution is a specific algorithm that uses polynomial time per training example

- Compute the number m of random training examples sufficient to ensure L will
  - with probability $(1 - \delta)$
  - output a hypothesis with maximum error ε

Consider the hypothesis space H defined by conjunctions of literals based on n Boolean variables:

$$x1, x2, \ldots .xn$$

- Each conjunction defines a subset of X eg. x1, ~x2 ,x4
- The concept class C:
  - all subsets of X defined by conjunctions of Boolean literals over x1,……, xn
- H=C (hypothesis space H identical to C)

- To accomplish this, we need only determine the size IHI of the hypothesis space.
- The size |H|of this hypothesis space is 3^n.
  - There are only three possibilities for each variable in any given hypothesis:
    - Include the variable as a literal in the hypothesis (old/x1)
    - include its negation as a literal (~old/-x1)
    - ignore it
  - Given n such variables, there are **3^n distinct** hypotheses.

- Substituting $|H| = 3^n$

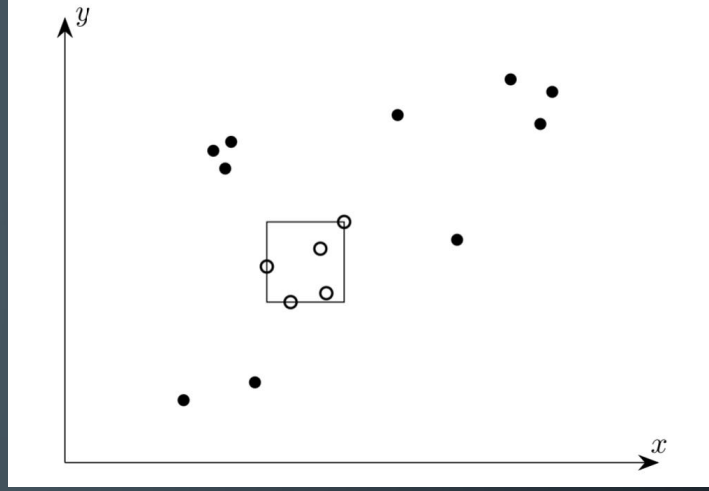
$$m \geq \frac{1}{\epsilon}(n \ln 3 + \ln(1/\delta))$$

- *m grows linearly in the number of literals n*
- *linearly in 1/* ε
- logarithmically in 1/ δ
- Hence polynomial computation

eg. FIND- S algorithm

- if a consistent learner attempts to learn a target concept described by conjunctions of up to 10 Boolean literals (n=10)
- we desire a 95% probability (1-δ) that it will learn a hypothesis with error less than 0.1 (ε)
- then it suffices to present *m* randomly drawn training examples,

EXAMPLE 2

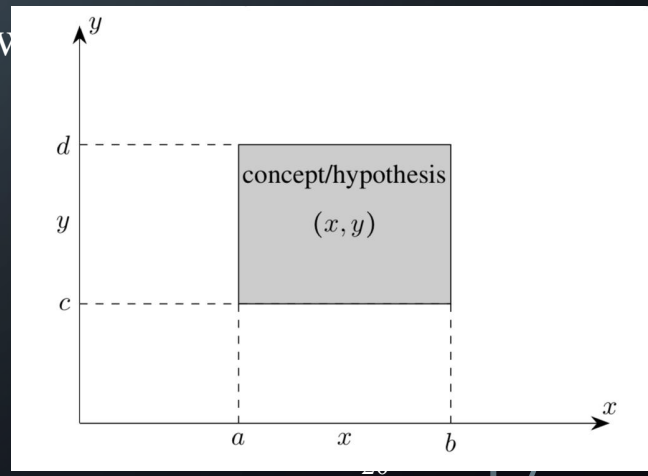


- Let the instance space be the set X of all points in the Euclidean plane.
- Each point is represented by its coordinates (x, y). Dimension or length of the instances is 2
- Let the concept class C be the set of all “axis-aligned rectangles” in the plane;
  - that is, the set of all rectangles whose sides are parallel to the coordinate axes in the plane
- Since an axis-aligned rectangle can be defined by a set of inequalities of the follov
  four parameters

  a <= x <= b

  c <= y <= d

  the size of a concept is 4



- We take the set H of all hyp et C of concepts, H = C

$$m \geq \frac{1}{\epsilon}(\ln|H| + \ln(1/\delta))$$

LEARNING MULTIPLE CLASSES

- Binary Classification - Two class problem
- K class problem - there may be more than two classes

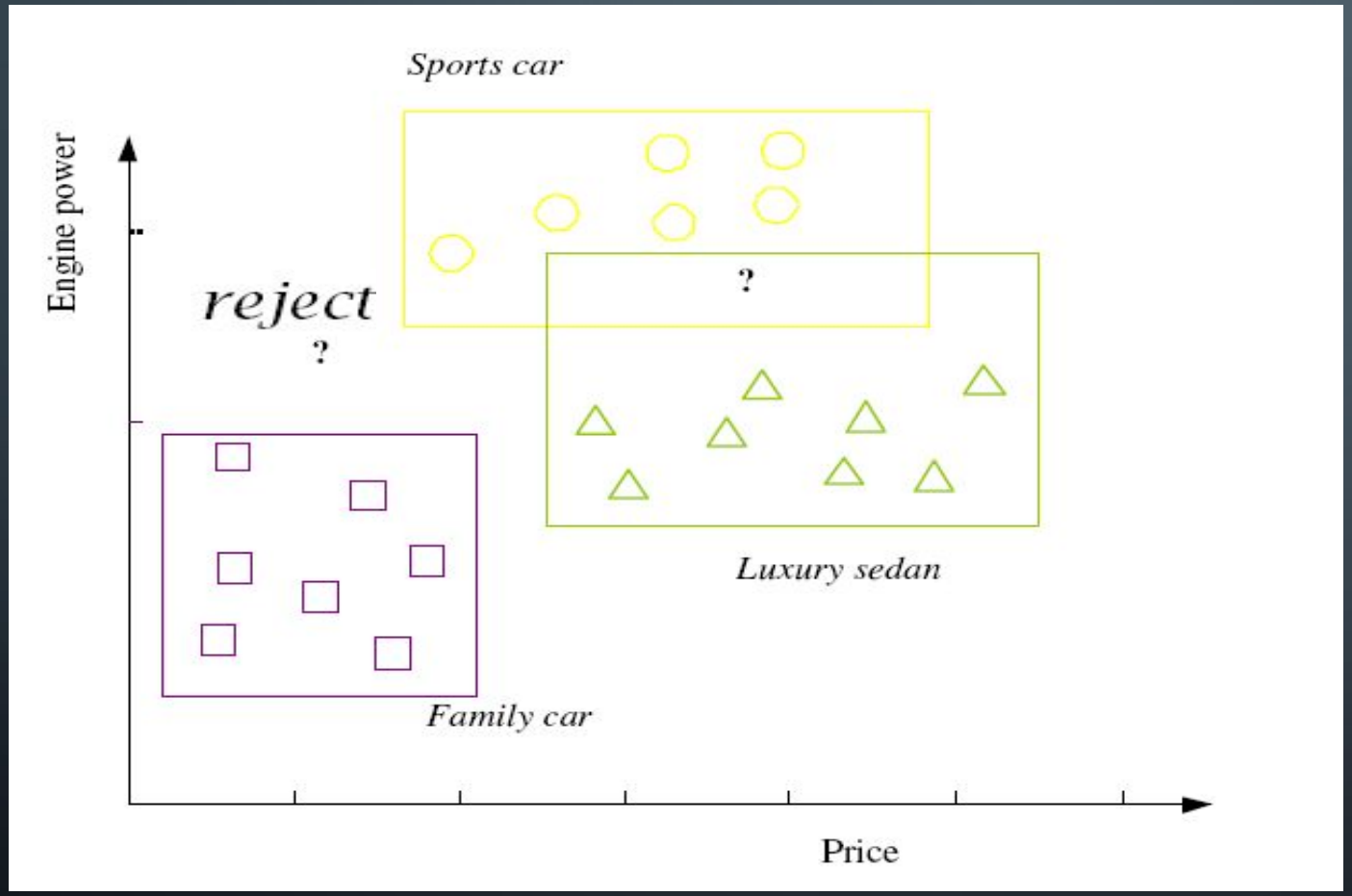
$$X = \{\mathbf{x}^t, r^t\}_{t=1}^{N}$$

where

$$r_i^t = \begin{cases} 1 \text{ if } \mathbf{x}^t \in C_i \\ 0 \text{ if } \mathbf{x}^t \in C_j, j \neq i \end{cases}$$

- Two methods are generally used to handle such cases:
  - “one-against-all"
  - “one-against-one”

LEARNING MULTIPLE CLASSES, C_I I=1,...,K ONE AGAINST MANY APPROACH



$$X = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

$$r_i^t = \begin{cases} 1 \text{ if } \mathbf{x}^t \in C_i \\ 0 \text{ if } \mathbf{x}^t \in C_j, j \neq i \end{cases}$$

Train hypotheses $h_i(\mathbf{x})$, $i = 1,...,K$:

$$h_i\left(\mathbf{x}^t\right) = \begin{cases} 1 \text{ if } \mathbf{x}^t \in C_i \\ 0 \text{ if } \mathbf{x}^t \in C_j, j \neq i \end{cases}$$

- 3 hypothesis induced, each one covering instances of one class and leaving instances of other two classes
- ML- learn boundary separating instances of one class from the instances of all other classes

$$h_i(\mathbf{x}^t) = \begin{cases} 1 \text{ if } \mathbf{x}^t \in C_i \\ 0 \text{ if } \mathbf{x}^t \in C_j, j \neq i \end{cases}$$

- Total empirical error = sum of predictions (erroneous) for all the classes over all instances

$$E(\{h_i\}_{i=1}^K | \mathcal{X}) = \sum_{t=1}^{N} \sum_{i=1}^{K} 1(h_i(x^t) \neq r_i^t)$$

- For a given instance, only one of the hypothesis should be 1(the suitable class is chosen)

- If “no” or “two or more” hypothesis =1: case of doubt and classifier rejects such cases

- Can build hypothesis for both positive and negative instances

  Eg. For the problem separating family car(h1) from sports car(h2),

  - if input given luxury Sedan

LEARNING MULTIPLE CLASSES, C_I I=1,...,K ONE AGAINST ONE APPROACH

- a classifier is constructed for each pair of classes.
- If there are K different class labels then K(K − 1)/2 classifiers are constructed
- An unknown instance is classified with the class getting the most votes
- Ties are broken arbitrarily

For example,

- let there be three classes, A, B and C
- In the OVO method we construct $3(3-1)/2 = 3$ binary classifiers
  - A vs B (h1)
  - A vs C (h2)
  - B vs C (h3)
- To classify any x : apply each of the three classifiers (h1,h2,h3) to x
- Let the three classifiers assign the classes A, B, B respectively to x.
- Since a label to x is assigned by the majority voting, we assign the class label of B to x

MODEL SELECTION & GENERALIZATION

- Boolean functions : all inputs and outputs are binary
  - d inputs -> 2^d examples
  - 2^d examples -> $2^{2^{\wedge}d}$ possible Boolean functions

    As each example can be labeled 0 or 1

x_1	x_2	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}	h_{15}	h_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

MODEL SELECTION & GENERALIZATION

- Each distinct training example removes half the hypothesis (wrong guesses)

  Eg.

  - *If x1=0,x2=1 output should be 0*
  - *This eliminates h5,h6,h7,h8,h13,h14,h15,h16*

- *Interpretation of learning-*

  Start with all possible hypothesis -> with more training samples -> remove inconsistent hypothesis

INDUCTIVE BIAS

- Learning is an ill—posed problem
  - If the training set contains only a small subset of all possible instances solution is not unique
  - Training data itself is never sufficient to find a unique solution
  - There are always infinitely many consistent hypothesis

- Inductive **bias : assumptions made to about $\mathcal{H}$ so that learning is possible**

  Eg. Hypothesis class H - axis aligned rectangles

- The model selection is about choosing the right inductive bias

SIMPLE MODEL

- Even though a complex model may not be making any errors in prediction, there are certain advantages in using a simple model:
  1. A simple model is easy to use
  2. A simple model is easy to train. It is likely to have fewer parameters
  (easier to find the corner values of a rectangle than the control points of an arbitrary shape)
  3. A simple model is easy to explain
  4. A simple model would generalize better than a complex model.
  (This principle is known as Occam's razor)

A MODEL SHOULD NOT BE TOO SIMPLE

- Expectation: the simpler model to changes less than a complex model
- A simple model is thus said to have less **variance**

A too simple model assumes more, is more rigid

- May fail if indeed the underlying class is not that simple
- A simpler model has more **bias**

- Finding the optimal model corresponds to minimizing both the bias and the variance

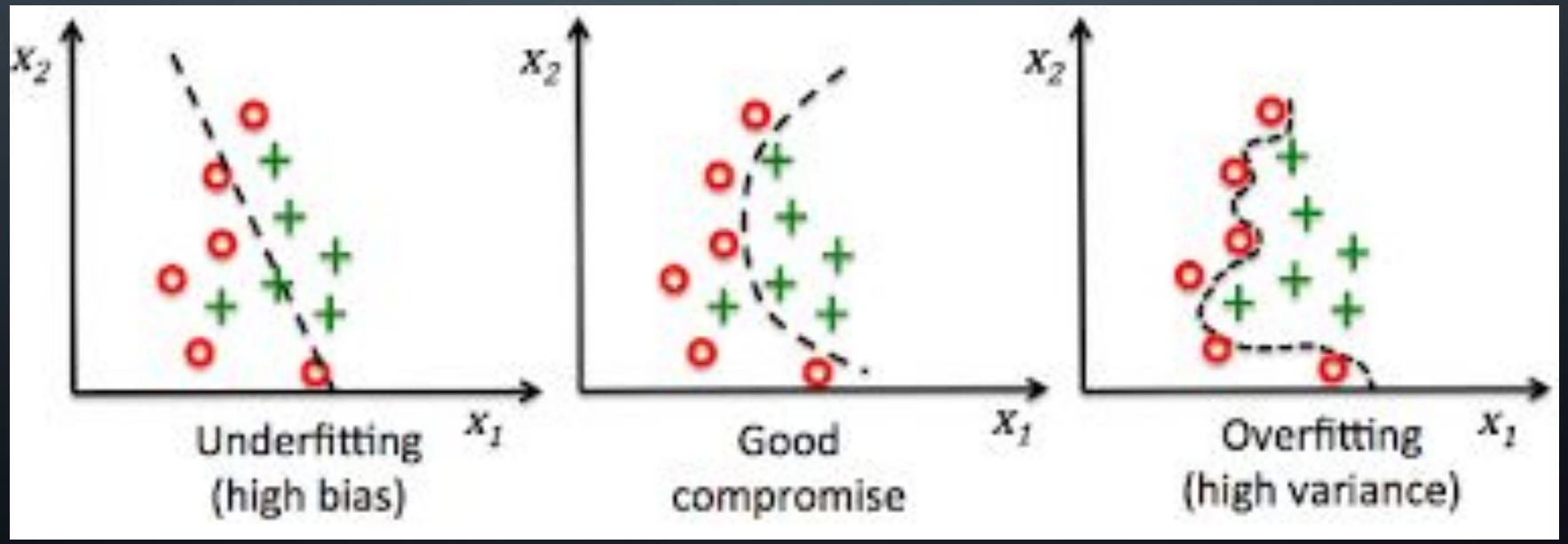
BIAS AND VARIANCE

- Bias :Error from erroneous assumption in the learning algorithm

  Eg. Assuming linearity (high bias) results in under fitting

- Variance: error from sensitivity to small fluctuations in the training set

  Eg. Using higher order polynomial than necessary results in over- fitting



MODEL SELECTION AND GENERALIZATION

GENERALIZATION

- How well a Model predict the right output for new instances
- Concepts learned by a machine learning model apply to specific examples (not seen by the model when it was learning)
- Generalize well from the training data to any data from the problem domain
- Make predictions in the future on data the model has never seen
- Overfitting and underfitting are the two biggest causes for poor performance

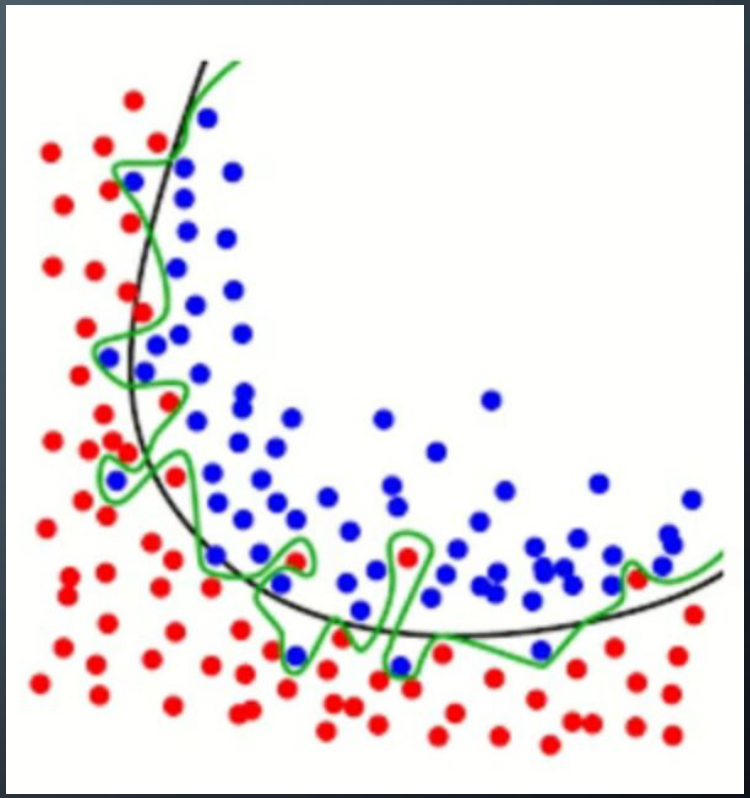
MODEL SELECTION AND GENERALIZATION

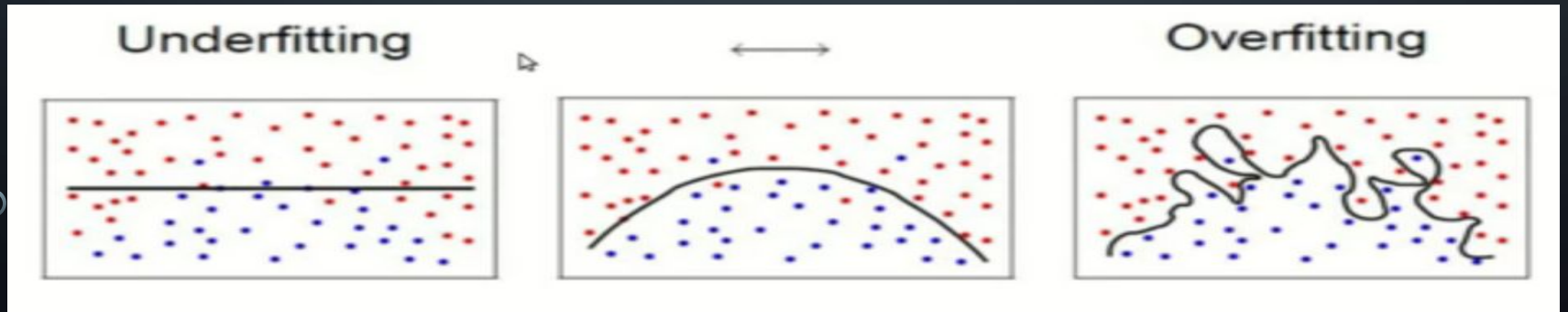
- Generalization:

  How well a model performs on a new data

- Overfitting-H more complex than C

- Underfitting-H less complex than C





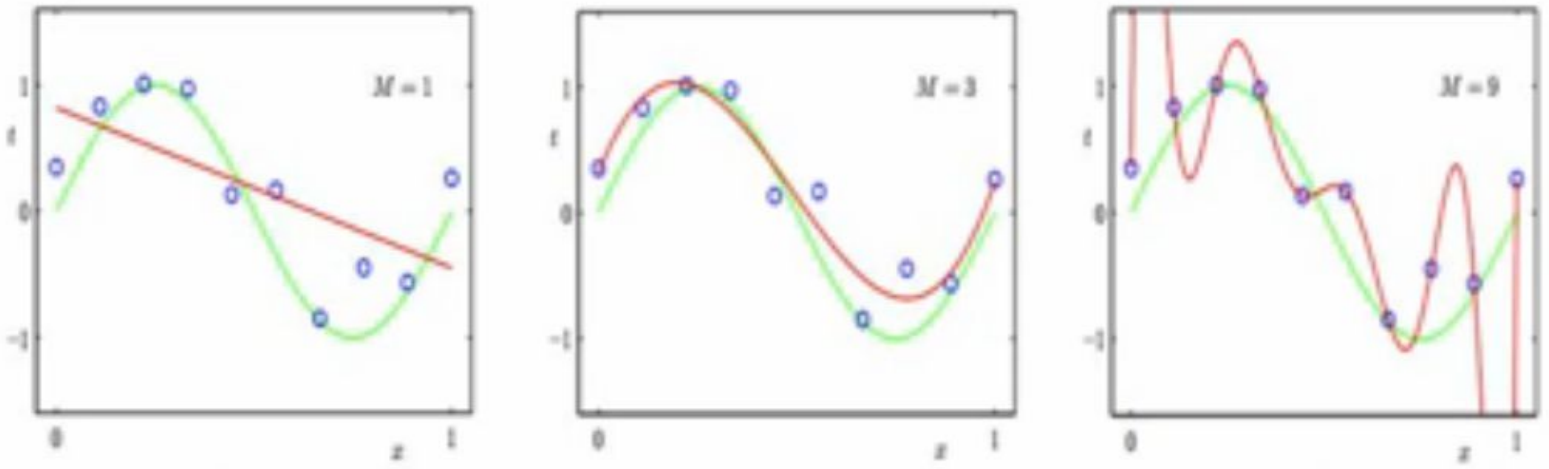
UNDER FITTING AND OVER FITTING

- two biggest causes for poor performance of machine learning algorithm

- Over fitting
  - model that models the training data too well
  - a model learns the detail & noise in the training data too well
  - negatively influences the performance of the model on new data

- Under fitting
  - model can neither model the training data nor generate new data
  - not suitable and will be obvious as it will have poor performance in the training data

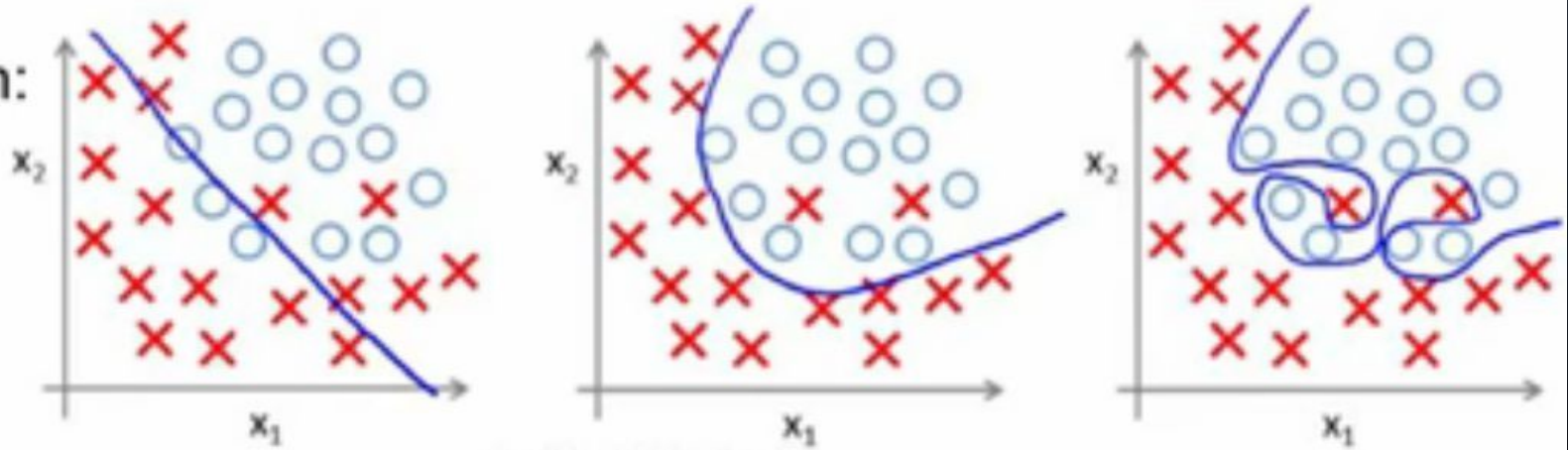
Regression:



predictor too inflexible:
cannot capture pattern

predictor too flexible:
fits noise in the data

Classification:



TRIPLE TRADE-OFF

- There is a trade-off between three factors (Dietterich, 2003):
  1. Complexity of H, $c(H)$,
  2. Training set size, N
  3. Generalization error, E, on new data
- Dependencies
  - As $N\uparrow$, $E\downarrow$
  - *For over-complex* H, $\alpha\sigma$ $N\uparrow$; first $E\downarrow$ and then $E\uparrow$

GENERALIZATION ERROR

Error on new examples!

- To test the generalization ability of hypothesis, dataset divided into two parts
  - Training Set (fit the hypothesis)
  - Validation set(test generalization)
- Two errors: training error, and testing error usually *called generalization error* *(http://en.wikipedia.org/wiki/Generalization_error)*
- Measuring the generalization error is a major challenge in data mining and machine learning (http://www.faqs.org/faqs/ai-faq/neural-nets/part3/section-11.html)

VALIDATION

- Hypothesis that is most accurate on the validation set is the best one-
- Cross validation : Process of finding the most accurate hypothesis using validation set
- To report error of the model, do not use validation error .

  Validation set- used to choose best model

- Test set/ publication set- contain examples not used in training or validation

  Eg: problem in class- training set

  problem in exam- validation set

  problem in later professional life- test set

CROSS-VALIDATION

- To estimate generalization error, we need data unseen during training. We split the data as
  - Training set (50%)
  - Validation set (25%)
  - Test (publication) set (25%)
- Resampling when there is few data

DIMENSIONS

- The complexity of any classifier or regressor depends on the number of inputs
  - time and space complexity
  - the necessary number of training examples
- the datasets have large number of variables for some problems
- the number of variables is more than the number of observations
- difficulty when dealing with such high-dimensional data
- Normally the number of input variables is reduced

DIMENSIONALITY REDUCTION

DIMENSIONALITY REDUCTION

In statistical and machine learning, dimensionality reduction or dimension reduction is the process of reducing the number of variables under consideration by obtaining a smaller set of principal variables.

WHY REDUCE DIMENSIONALITY?

- Reduces time complexity-d, N: Less computation
- Reduces space complexity: Less parameters thus memory -Saves the cost of extracting unnecessary feature
- Simpler models are more robust on small datasets and has got less variance(noise, outliers etc)
- Data can be explained with fewer features-More interpretable; simpler explanation
- Data visualization (structure, groups, outliers, etc)good, if plotted in fewer dimensions

DIMENSIONALITY REDUCTION

Two main methods for reducing dimensionality

1. ***Feature Selection***:
   - Finding k of the d dimensions that give us the most information
   - discard other (d-k) dimensions

   (Choosing $k<d$ important features, ignoring the remaining $d – k$)

   E.g. Subset selection algorithms

2. ***Feature Extraction:***
   - Finding new set of k dimensions that are combinations of original d dimensions

   (Project the original x_i , i =1,...,d dimensions to new $k<d$ dimensions, z_j , j =1,...,k)

   E.g. Supervised (LDA) or unsupervised(PCA)

FEATURE SELECTION

- Feature selection techniques are used for four reasons:
  - It reduces the complexity of a model and makes it easier to interpret
  - It enables the machine learning algorithm to train faster
  - It improves the accuracy of a model if the right subset is chosen
  - It reduces overfitting
- Feature selection technique is used when
  - the data contains many features that are either redundant or irrelevant
  - can thus be removed without incurring much loss of information

SUBSET SELECTION

- There are 2^d subsets of d features
- Finding best subset of set of features (variables, predictors) for use in model construction
- Contains least no of dimensions with most accuracy
- Two approaches
  - Forward selection
  - Backward selection

Forward Selection

- Start with no variables
- Add one by one, which decreases the error most
- Stop till further addition of features does not decrease the error(or slight decrease)

Backward Selection

- Start with all variables
- Remove them one by one, whose effect decreases the error the most
- Stop until any further removal increases the error

- We use the following notations:
  - n : number of input variables
  - x1; : : : ; xn : input variables
  - F_i : a subset of the set of input variables
  - $E(F_i)$: error incurred on the validation sample when only the inputs in Fi are used

SEQUENTIAL FORWARD SELECTION

1. Set of features F initially Ø.
2. At each iteration, for all possible $x_{i,}$ train the model on training set and calculate $E(F \cup x_i)$ on validation set
3. *Choose the input x_j that causes the least error*

   $$j = \arg\min_i E(F \cup x_i)$$
4. Add x_j to F if $E(F \cup x_j) < E(F)$

- Stop adding feature if the decrease in E is not there or too small
- Algorithm also called *wrapper method*, process of feature extraction is thought to wrap around the learner

- to decrease the dimensions from n to k,
  - need to train and test the system following number of times

    $$n + (n - 1) + (n - 2) + \ldots\ldots + (n - k)$$
  - This process may be costly $O(n^2)$

SEQUENTIAL BACKWARD SELECTION

- Set $F_0 = \{x_1, \ldots\ldots, x_n\}$ and $E(F_0) = \infty$
- For i = 0,1,…., repeat the following until $E(F_{i+1}) >= E(F_i)$:
  - For all possible input variables x_j , train the model with the input variables $F_i - \{x_j\}$ and calculate $E(F_i - \{x_j\})$ on the validation set
  - Choose that input variable that causes the least error $E(F_i - \{x_j\})$:

$$m = \arg\min_j E(F_i - \{x_j\})$$

  - (c) Set $F_{i+1} = F_i - \{x_m\}$
- The set F_i is outputted as the best subset.

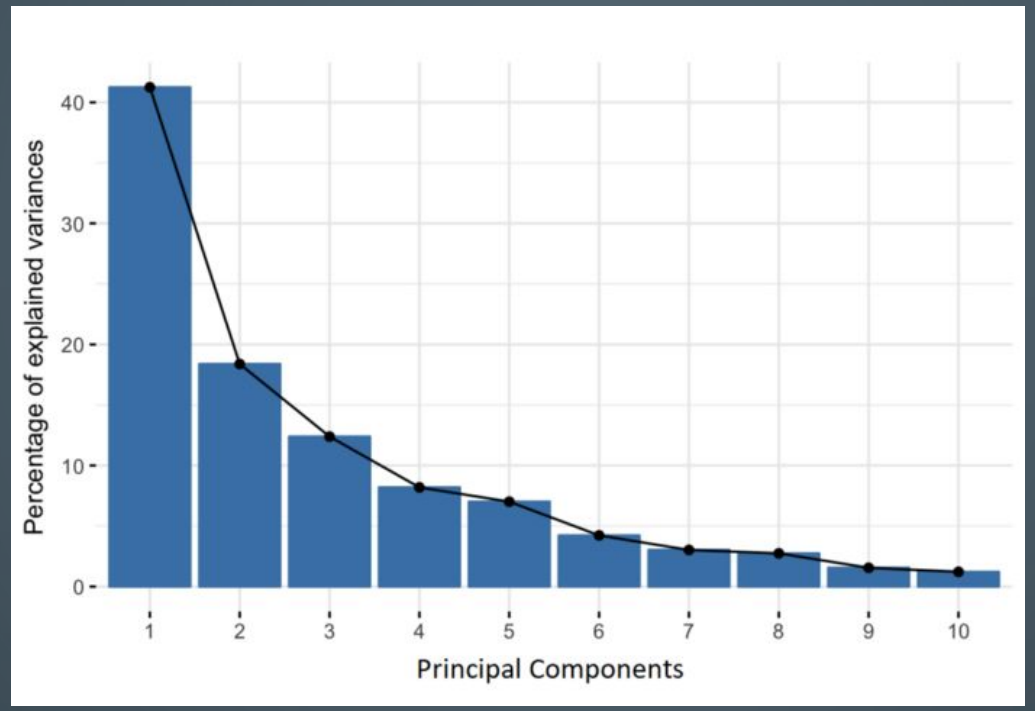
PRINCIPAL COMPONENT ANALYSIS

PRINCIPAL COMPONENT ANALYSIS

- **Principal Components Analysis (PCA)** is a technique that can be used to simplify a dataset
- Feature Extraction method
- Principal Components - Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables
  - the new variables (i.e., principal components) are uncorrelated
  - most of the information within the initial variables is compressed into the first components
- 10-dimensional data gives you 10 principal components
  - PCA tries to put maximum possible information in the first component
  - then maximum remaining information in the second and so on

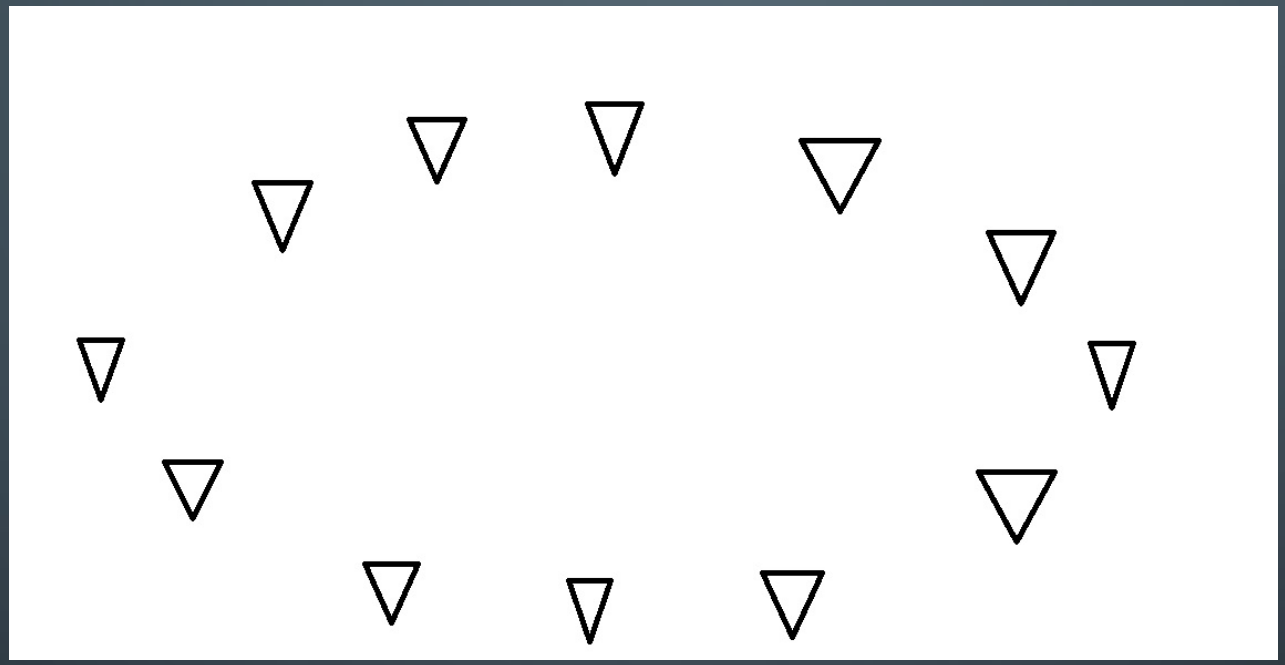
PCA

- It is unsupervised as it does not use output info but maximizes variance
- It is a linear transformation that chooses a new coordinate system for the data set such that
  - greatest variance by any projection of the data set comes to lie on the first axis (then called the first principal component)
  - the second greatest variance on the second axis, and so on
- PCA can be used for reducing dimensionality by eliminating the later principal components



- The principal components are less interpretable and don't have any real meaning
- Principal components represent the directions of the data that explain a **maximal amount of variance**
  - the larger the variance carried by a line, the larger the dispersion of the data points along it
  - and the larger the dispersion along a line, the more the information it has
- The lines that capture most information of the data
- After projection onto principal components the sample is most spread out

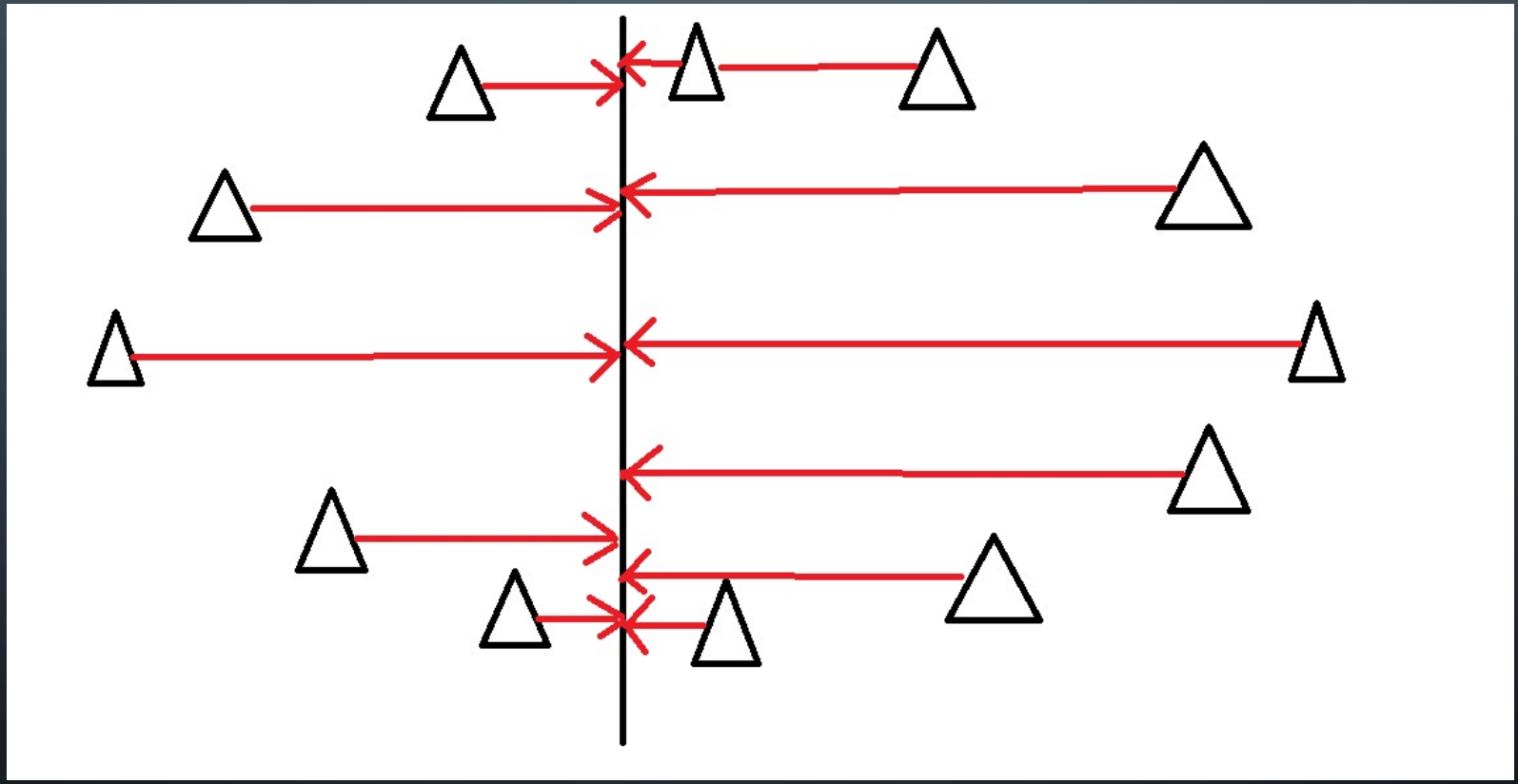
WHAT IS PRINCIPAL COMPONENT ANALYSIS?



- They are the directions where there is the most variance, the directions where the data is most spread out.

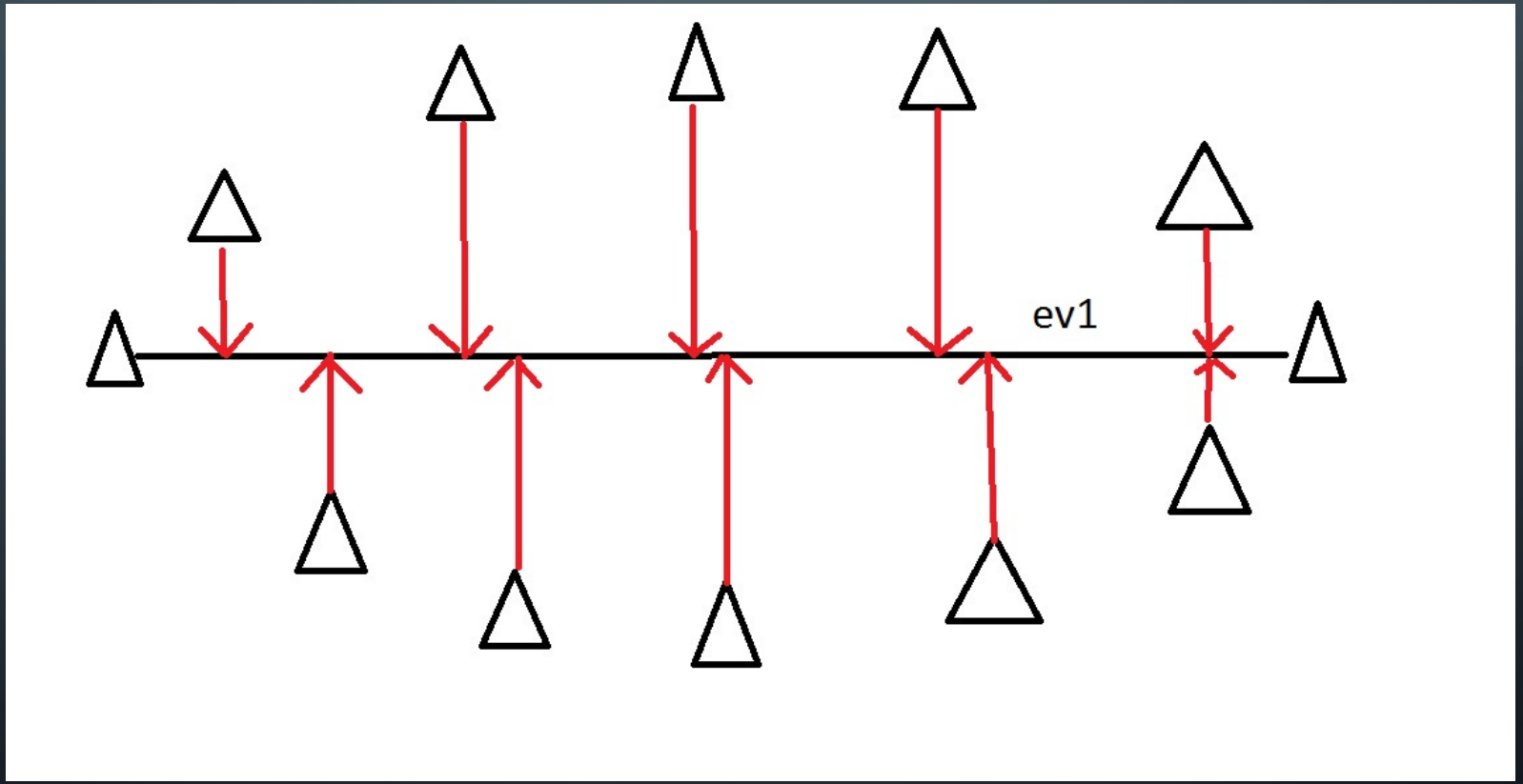
TO FIND THE DIRECTION WHERE THERE IS MOST VARIANCE

Find the straight line where the data is most spread out when projected onto it.
A vertical straight line with the points projected on to it will look like this:

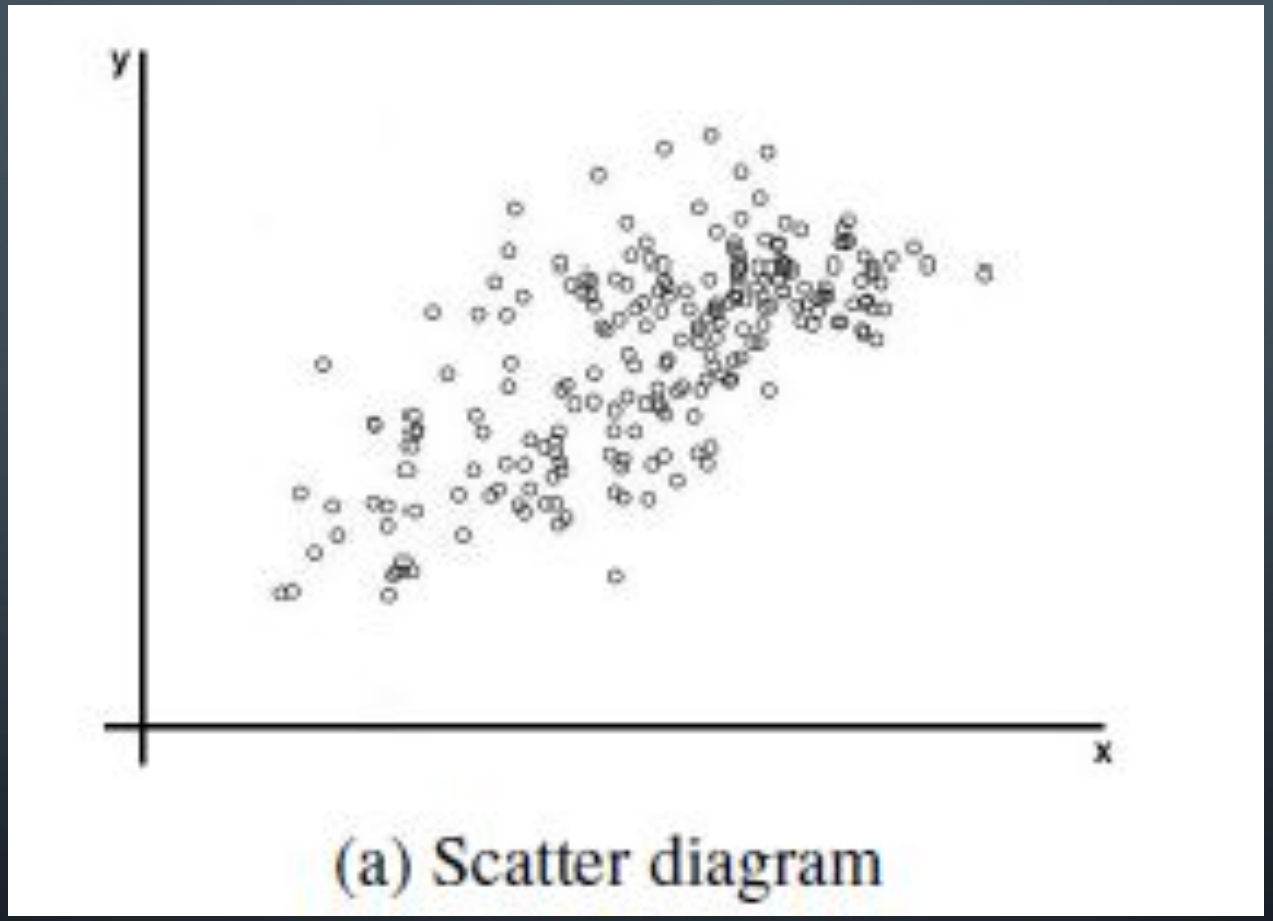


ON THIS LINE THE DATA IS WAY MORE SPREAD OUT, IT HAS A LARGE VARIANCE.

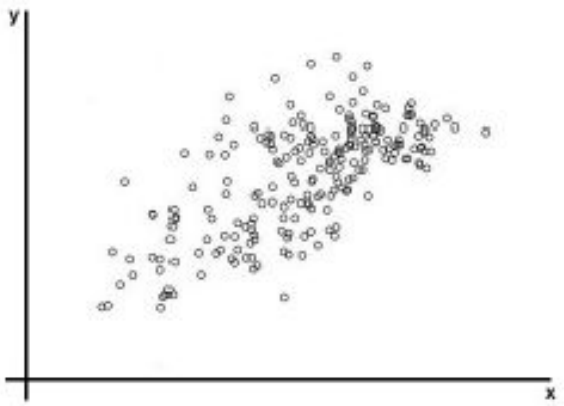
In fact there isn't a straight line you can draw that has a larger variance than a horizontal one. A horizontal line is therefore the principal component in this example.



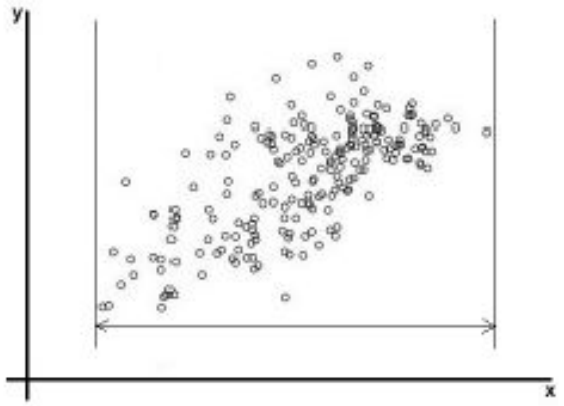
PCA



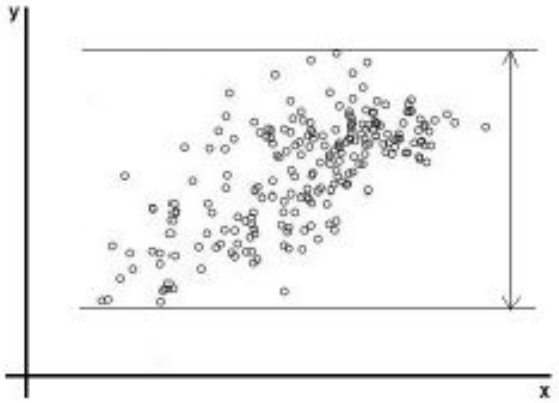
(a) Scatter diagram



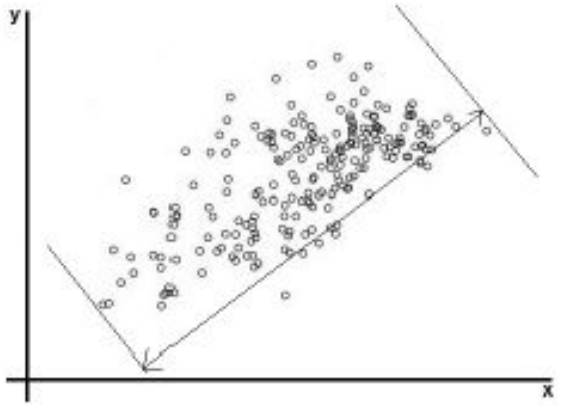
(a) Scatter diagram



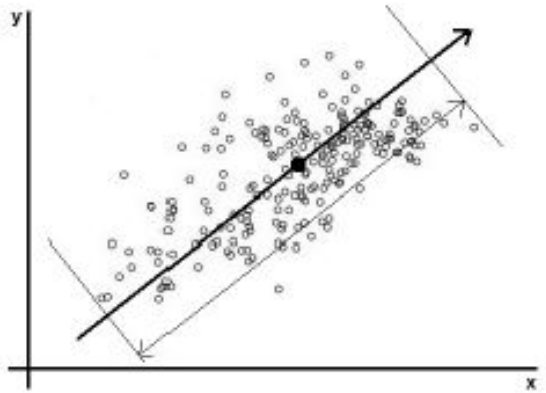
(b) Spread along x-direction



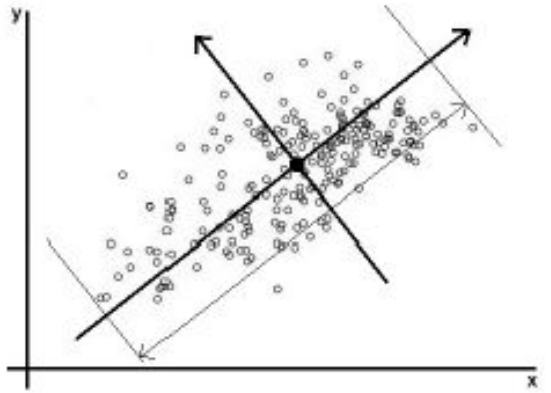
(c) Spread along y-direction



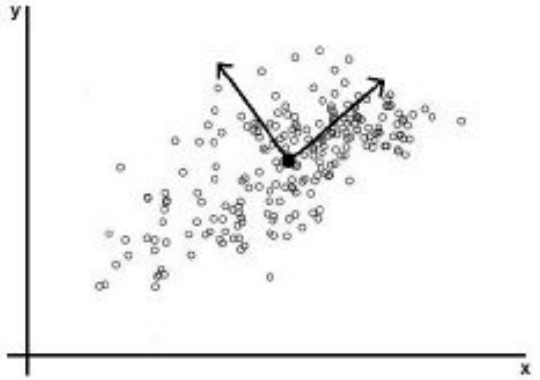
(d) Largest spread



(e) Direction of largest spread : Direction of the first principal component (solid dot is the point whose coordinates are the means of x and y)



(f) Directions of principal components



(g) Principal component vectors (unit vectors in the directions of principal components)

1st Principal Component:

- Graphically: the line that is most spread out
- Mathematically: the line that maximizes the variance
  - i.e. average of squared distances from the projected points to the origin

2nd Principal Component:

  - Uncorrelated with 1st PC
  - Next highest covariance

and so on..

EXAMPLE APPLICATIONS

- Face Recognition
- Image Compression
- Pattern finding
- Gene Expression Analysis
- Data Reduction
- Data Classification
- Trend Analysis
- Factor Analysis
- Noise Reduction

PCA COMPUTATION

- Heavily dependent on mathematical concepts
- Basic idea:
  - Data has n features and N samples
  - Compute mean of the variables(features) from sample
  - Calculate covariance matrix
  - Calculate eigenvalues and eigenvectors of the covariance matrix
    - Eigenvalues and eigenvectors come in pairs
    - Rank the eigenvectors in order of their eigenvalues in descending order
      - Eigenvector with highest eigenvalue is 1st Principal Component
      - Next highest 2nd Principal Component and so on.
    - Feature vector is formed from the set of eigenvectors
  - Recast the data along the Principal Components –

    reorient the data from the original axes to the ones represented by the principal components

EIGENVALUES AND EIGENVECTORS (ALGEBRA)

Square matrix-A ; eigenvector v ; eigenvalue λ

$$Av = \lambda v$$

Example: For this matrix

$$\begin{bmatrix} -6 & 3 \\ 4 & 5 \end{bmatrix}$$

an eigenvector is:

$$\begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

with a matching eigenvalue of 6

Av gives us:

$$\begin{bmatrix} -6 & 3 \\ 4 & 5 \end{bmatrix}\begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} -6\times1+3\times4 \\ 4\times1+5\times4 \end{bmatrix} = \begin{bmatrix} 6 \\ 24 \end{bmatrix}$$

λv gives us :

$$6\begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 6 \\ 24 \end{bmatrix}$$

Yes they are equal! So Av = λv as promised.

To find eigenvalues and eigenvectors:

- Start by finding the eigenvalue: the equation must be true

$$Av = \lambda v$$

- Place identity matrix in the equation so that we are deling with matrix-vs-matrix

$$Av = \lambda Iv$$

- Bring all to left hand side

$$Av - \lambda Iv = 0$$

- If v is non zero we can solve for lambda using just the determinant

$$| A - \lambda I | = 0$$

- Solving for the example:

Start with $|A - \lambda I| = 0$

$$\left|\begin{bmatrix} -6 & 3 \\ 4 & 5 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right| = 0$$

Which is:

$$\begin{vmatrix} -6-\lambda & 3 \\ 4 & 5-\lambda \end{vmatrix} = 0$$

Calculating that determinant gets:

$$(-6-\lambda)(5-\lambda) - 3\times4 = 0$$

Which then gets us this Quadratic Equation:

$$\lambda^2 + \lambda - 42 = 0$$

And solving it gets:

$$\lambda = -7 \text{ or } 6$$

Example (continued): Find the Eigenvector for the Eigenvalue $\boldsymbol{\lambda = 6}$:

Start with:

$$Av = \lambda v$$

Put in the values we know:

$$\begin{bmatrix} -6 & 3 \\ 4 & 5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = 6\begin{bmatrix} x \\ y \end{bmatrix}$$

After multiplying we get these two equations:

$$\begin{aligned} -6x + 3y &= 6x \\ 4x + 5y &= 6y \end{aligned}$$

Bringing all to left hand side:

$$\begin{aligned} -12x + 3y &= 0 \\ 4x - 1y &= 0 \end{aligned}$$

Either equation reveals that **y = 4x**, so the **eigenvector** is any non-zero multiple of this:

$$\begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

And we get the solution shown at the top of the page:

$$\begin{bmatrix} -6 & 3 \\ 4 & 5 \end{bmatrix}\begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} -6\times1+3\times4 \\ 4\times1+5\times4 \end{bmatrix} = \begin{bmatrix} 6 \\ 24 \end{bmatrix}$$

... and also ...

$$6\begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 6 \\ 24 \end{bmatrix}$$

So $Av = \lambda v$

EIGENVALUES AND EIGENVECTORS IN PCA

- suppose that our data set is 2-dimensional with 2 variables ***x,y***
- the eigenvectors and eigenvalues of the covariance matrix are as follows:

$$v1 = \begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \qquad \lambda_1 = 1.284028$$

$$v2 = \begin{bmatrix} -0.7351785 \\ 0.6778736 \end{bmatrix} \qquad \lambda_2 = 0.04908323$$

- If we rank the eigenvalues in descending order, we get λ1>λ2,
  - the eigenvector that corresponds to the first principal component (PC1) is *v1*
  - the one that corresponds to the second component (PC2) is*v2*
- To compute the percentage of variance (information) accounted for by each component
  - divide the eigenvalue of each component by the sum of eigenvalues

$$\lambda_1/(\lambda_1+\lambda_2) \text{ and } \lambda_2/(\lambda_1+\lambda_2)$$

- If we apply this on the example above
  - PC1 carries 96% and

PCA COMPUTATION PROCESS

Step 1. Data

We consider a dataset having n features or variables denoted by $X_1, X_2, \ldots, X_n$. Let there be N examples. Let the values of the i-th feature X_i be $X_{i1}, X_{i2}, \ldots, X_{iN}$ (see Table 4.1).

Features	Example 1	Example 2	$\cdots$	Example N
X_1	X_{11}	X_{12}	$\cdots$	X_{1N}
X_2	X_{21}	X_{22}	$\cdots$	X_{2N}
$\vdots$				
X_i	X_{i1}	X_{i2}	$\cdots$	X_{iN}
$\vdots$				
X_n	X_{n1}	X_{n2}	$\cdots$	X_{nN}

Table 4.1: Data for PCA algorithm

Step 2. Compute the means of the variables

We compute the mean $\bar{X}_i$ of the variable X_i:

$$\bar{X}_i = \frac{1}{N}(X_{i1} + X_{i2} + \cdots + X_{iN}).$$

- Subtract the mean from each of the data dimensions.
- All the x values have $\bar{x}$ subtracted and y values have $\bar{y}$ subtracted from them.
- This produces a data set whose mean is zero.
- Subtracting the mean makes variance and covariance calculation easier by simplifying their equations.
- The variance and co-variance values are not affected by the mean value.

PCA PROCESS

- Calculate the covariance matrix

Step 3. Calculate the covariance matrix

Consider the variables X_i and X_j (i and j need not be different). The covariance of the ordered pair (X_i, X_j) is defined as[1]

$$\text{Cov}\,(X_i, X_j) = \frac{1}{N-1}\sum_{k=1}^{N}(X_{ik} - \bar{X}_i)(X_{jk} - \bar{X}_j). \tag{4.1}$$

We calculate the following $n \times n$ matrix S called the covariance matrix of the data. The element in the i-th row j-th column is the covariance $\text{Cov}\,(X_i, X_j)$:

$$S = \begin{bmatrix} \text{Cov}\,(X_1, X_1) & \text{Cov}\,(X_1, X_2) & \cdots & \text{Cov}\,(X_1, X_n) \\ \text{Cov}\,(X_2, X_1) & \text{Cov}\,(X_2, X_2) & \cdots & \text{Cov}\,(X_2, X_n) \\ \vdots & & & \\ \text{Cov}\,(X_n, X_1) & \text{Cov}\,(X_n, X_2) & \cdots & \text{Cov}\,(X_n, X_n) \end{bmatrix}$$

Step 4. Calculate the eigenvalues and eigenvectors of the covariance matrix

Let S be the covariance matrix and let I be the identity matrix having the same dimension as the dimension of S.

i) Set up the equation:

$$\det(S - \lambda I) = 0. \tag{4.2}$$

This is a polynomial equation of degree n in λ. It has n real roots (some of the roots may be repeated) and these roots are the eigenvalues of S. We find the n roots $\lambda_1, \lambda_2, \ldots, \lambda_n$ of Eq. (4.2).

$$| A - \lambda I | = 0$$

ii) If $\lambda = \lambda'$ is an eigenvalue, then the corresponding eigenvector is a vector

$$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$$

such that

$$(S - \lambda' I)U = 0.$$

(This is a system of n homogeneous linear equations in $u_1, u_2, \ldots, u_n$ and it always has a nontrivial solution.) We next find a set of n orthogonal eigenvectors $U_1, U_2, \ldots, U_n$ such that U_i is an eigenvector corresponding to λ_i.[2]

$$Av - \lambda Iv = 0$$

iii) We now normalise the eigenvectors. Given any vector X we normalise it by dividing X by its length. The length (or, the norm) of the vector

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

is defined as

$$\|X\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

Given any eigenvector U, the corresponding normalised eigenvector is computed as

$$\frac{1}{\|U\|} U.$$

We compute the n normalised eigenvectors $e_1, e_2, \ldots, e_n$ by

$$e_i = \frac{1}{\|U_i\|} U_i, \quad i = 1, 2, \ldots, n.$$

$\|v_i\|$

Step 5. Derive new data set

Order the eigenvalues from highest to lowest. The unit eigenvector corresponding to the largest eigenvalue is the first principal component. The unit eigenvector corresponding to the next highest eigenvalue is the second principal component, and so on.

i) Let the eigenvalues in descending order be $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ and let the corresponding unit eigenvectors be $e_1, e_2, \ldots, e_n$.

ii) Choose a positive integer p such that $1 \leq p \leq n$.

iii) Choose the eigenvectors corresponding to the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ and form the following $p \times n$ matrix (we write the eigenvectors as row vectors):

$$F = \begin{bmatrix} e_1^T \\ e_2^T \\ \vdots \\ e_p^T \end{bmatrix},$$

where T in the superscript denotes the transpose.

iv) We form the following $n \times N$ matrix:

$$X = \begin{bmatrix} X_{11} - \bar{X}_1 & X_{12} - \bar{X}_1 & \cdots & X_{1N} - \bar{X}_1 \\ X_{21} - \bar{X}_2 & X_{22} - \bar{X}_2 & \cdots & X_{2N} - \bar{X}_2 \\ \vdots & & & \\ X_{n1} - \bar{X}_n & X_{n2} - \bar{X}_n & \cdots & X_{nN} - \bar{X}_n \end{bmatrix}$$

v) Next compute the matrix:

$$X_{\text{new}} = FX.$$

Note that this is a $p \times N$ matrix. This gives us a dataset of N samples having p features.

Step 6. New dataset

The matrix X_{new} is the new dataset. Each row of this matrix represents the values of a feature. Since there are only p rows, the new dataset has only features.

Step 7. Conclusion

This is how the principal component analysis helps us in dimensional reduction of the dataset. Note that it is not possible to get back the original n-dimensional dataset from the new dataset.

PROPORTION OF VARIANCE

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d}$$

PRINCIPAL COMPONENT ANALYSIS

Lets walk through the illustrative example in the notes for a better understanding

PCA REFERENCES

- https://builtin.com/data-science/step-step-explanation-principal-component-analysis
- https://www.mathsisfun.com/algebra/eigenvalue.html
- Notes pdf shared
- Alpydin reference book