Variables

↳ Categorical → Nominal
→ Ordinal

↳ Numerical → Discrete
→ Continuous

KNN Classifier Implementation:

It takes test data X as input and returns predictions for each data point in X. For each test data point x, the method computes the Euclidean distance between x and each data point in the training set X_train.

Based on these distances, it identifies the indices of the k nearest neighbors. It sorts them according to their distances until the given k value (`sorted_indices[:self.k]`). It then retrieves the corresponding labels from the training labels y_train.

A majority vote is performed among these labels to determine the predicted label for the test data point x by counting the labels in loop.

This process is repeated for all test data points, and the predictions are collected into a list, which is then returned.

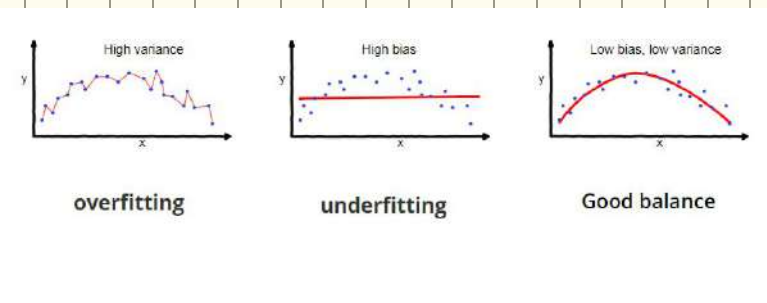
Assesing and Comparing Classification Algs.

Input Encoding:

100 data → M_{KNN} → depends heavily on 100 data points

" → SVM → 5 S.V. chosen from 100 d.p.

⁂ Changing data will change the model

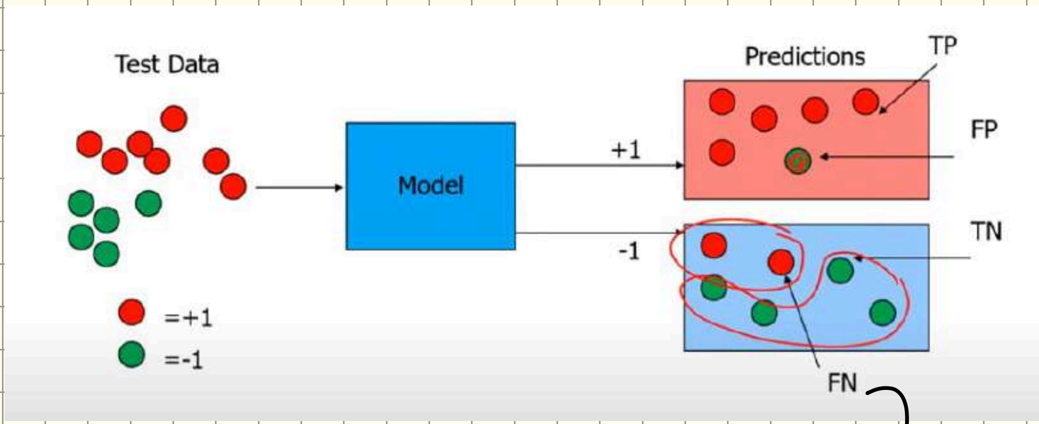


In 100 data you give someone 50 and you give other one 50 data. One of them result is higher because his data can be biased.

↓ Çözüm

Cross-validation.

Performance measurements.



→ persondies!

↓

FN önemli bir ölçüt.

$$\text{Accuracy} = \frac{TP + TN}{All}$$

Precision = $TP / (TP + FP)$, $\frac{\text{Gerçekten } +1}{\text{prediction } +1}$ } +1 Tahminlerimizin kaçı doğru?

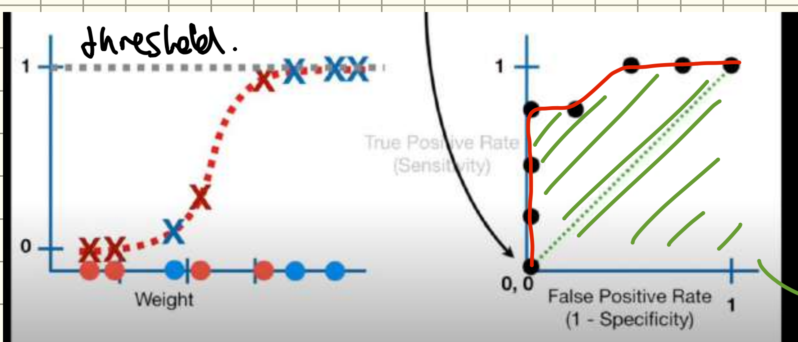
Recall = $TP / (TP + FN)$, $\frac{\text{gerçekten } +1}{\text{tüm } +1}$ } Gerçekten kanser olanların kaçını doğru bildik?

F1_score

ROC - Curve

Sensitivity: $TPR = \frac{TP}{TP + FN}$

Specifity: $FPR = (1 - TPR) = \frac{FP}{FP + TN}$



→ Summarizes all of the C.M. that each threshold produced.

→ AUC

Prediction ve actual karşılaştırması

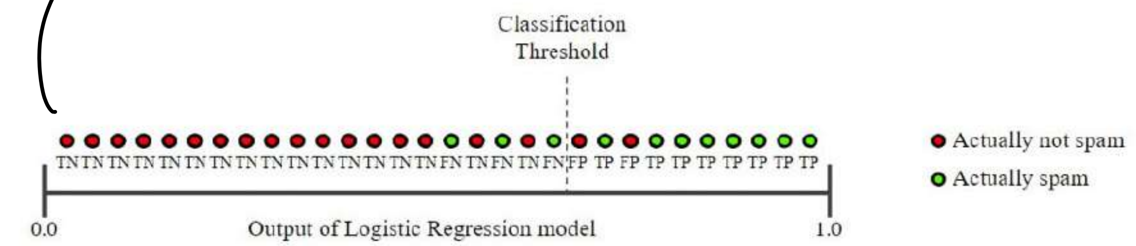


Figure 1. Classifying email messages as spam or not spam.

Let's calculate precision and recall based on the results shown in Figure 1:

True Positives (TP): 8	False Positives (FP): 2
False Negatives (FN): 3	True Negatives (TN): 17

Precision measures the percentage of **emails flagged as spam** that were correctly classified—that is, the percentage of dots to the right of the threshold line that are green in Figure 1:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{8}{8+2} = 0.8$$

Recall measures the percentage of **actual spam emails** that were correctly classified—that is, the percentage of green dots that are to the right of the threshold line in Figure 1:

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{8}{8+3} = 0.73$$

Decision Trees

$X \rightarrow$ Model $\rightarrow y$

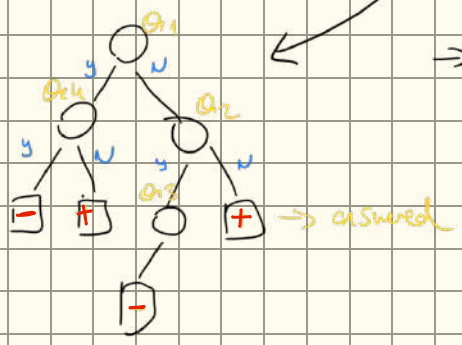
↓

tree

Training data

↓

DT algorithm



→ Binary classification example.

→ answered

"salary, savings → credit worthy" "not"

- Can I have an algorithm to build a tree like this?

Questions

$x \in \mathbb{R}^d$

↓

d dimensional space

$x_i > 10{,}000$ } yes or no questions

↓

salary

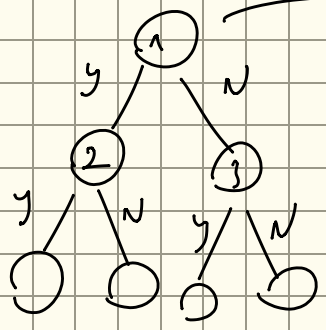
x_i = "March"

x_i = "March" || "April"

x_i =

⋮

- I can have infinite many questions.
- We have infinitely many possibilities.



→ Satış 1 M

Sayısı 10 M olsa, sadece root question için bile 11 M tane sorum olacak.

(possibility)

$Q_1 \rightarrow 11M$

$Q_2 \rightarrow 11M$ } $11M^3$

$Q_3 \rightarrow 11M$

impossible to solve.

We should find an optimal solution.

lets try with binary questions

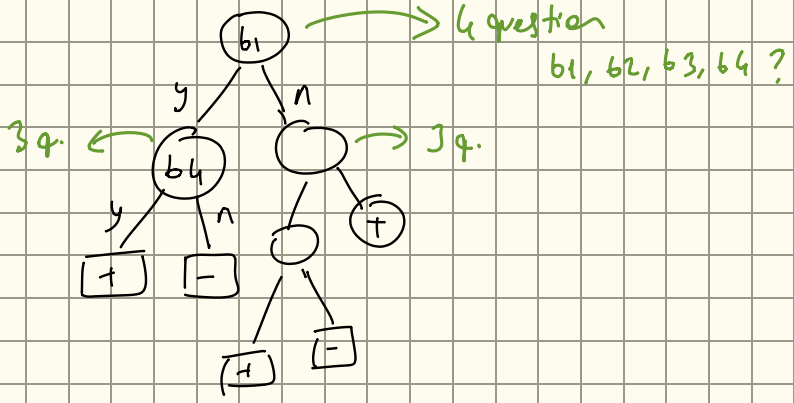
$x_i = \{ b_1, b_2, b_3, b_4 \}$

employed, children, ..., server cipher

try with categorical

$\vec{x} = \{ c_1, c_2, c_3, c_4 \}$

job, degree, ...



4 question

b1, b2, b3, b4 ?

is c_1 { teacher, employer, ... } proportional to data

binary / categorical → decision ≡

Integers → decision → <, ..., =

real → " → <, >, ...

1-) What kind of questions? } salary > 5000
savings > 10,000

2-) Limit those questions.

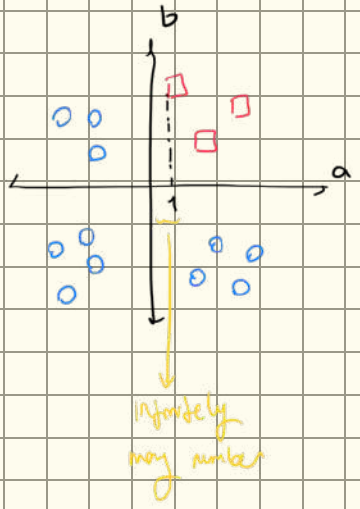
- For categorical only equal
- For numerical only greater.

Why limit?

- Real world questions are NP-hard.

↳ Helps to manage the computational complexity. Balance.

questions are only > or =



? → first node

how many question?

- I have 2 different but infinitely many of them.

a > 0.a

b > 0.b

b > 0

b > 1

b > -1

⋮

Lets say:

I can also say

a > 0,01

a > 0,02

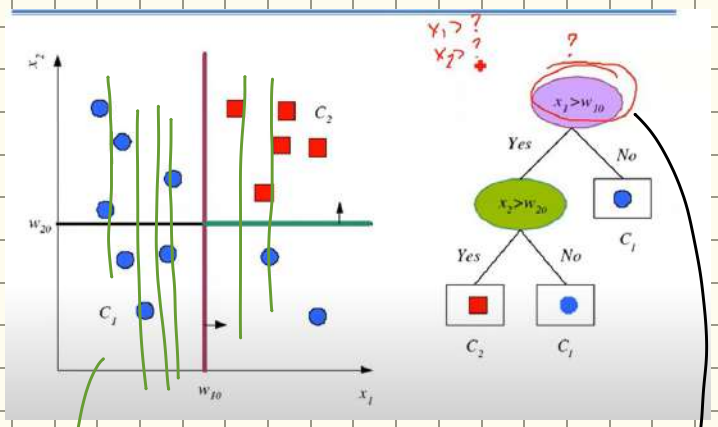
a > 0,555

⋮

I need to look middle point.

→ Sort a's in ascending order.

Midpoint of any pair is a candidate for 0.c.



feature x_1 feature x_2

$[x_{11}, x_{21}], [x_{12}, x_{22}], \ldots, [x_{11n}, x_{21n}]$

mid point mid point mid point

14 prediction data points

13+13 = 26 possibilities for Q_1. $(2n-2)$
$(n-1)$ $(n-1)$

Build trees → "hierarchically"

Start from root, ask question in a way that your job becomes easier in next one. Try to do this with fewer num. of questions.

Heuristic way, hope that previous choice was my best choice.

$(2n-2) * (2n-3) * (2n-4)$ $O(n^3)$

Alg $= O(n^3)$

- Build all possible trees
- Assess performance of each
- Get the tree with best performance.

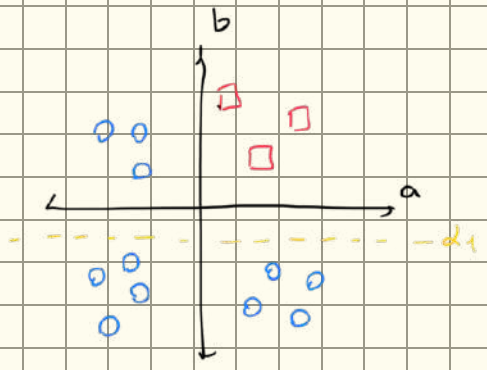
Choose the best split:

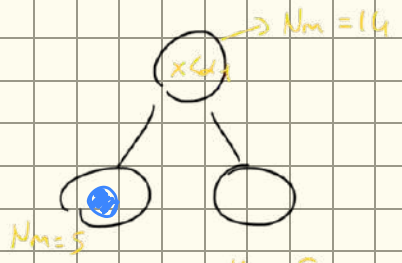
$$P(C_i \mid x_{i,m}) = p^i_m = \frac{N^i_m}{N_m}$$

N^i_m → nm of blue points.

N_m → the nm of points

Probability of blues.





$P_b = \frac{5}{5}$

$P_a = \frac{0}{5}$

This node is pure node

$P_b = \frac{4}{9}$

$P_r = \frac{5}{8}$

entropy higher than

→ Measure of impurity is entropy

Entropy will be 0.

$$H_M = -\sum_{i=1}^{k} p_m{}^i \log_2 p_m{}^i$$

→ If you add information, entropy ↓

Best split?

Maximize information gain

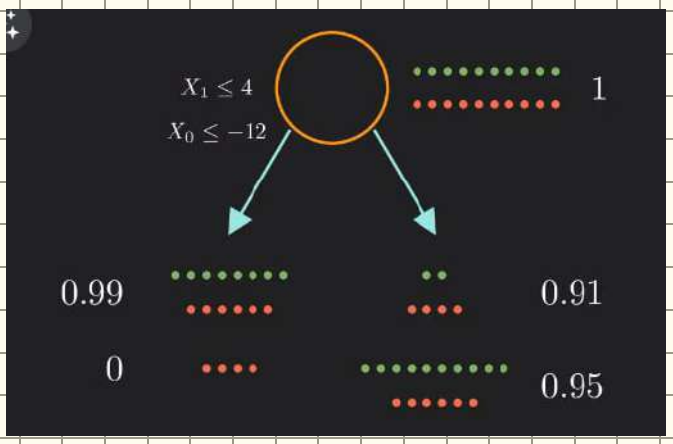
$$\text{Entropy} = \sum -p_i \cdot \log(p_i)$$

p_i → probability of ith class.

↑ entropy ↑ uncertainty

$$\text{Information Gain} = E(\text{parent}) - \sum w_i E(\text{child}_i)$$

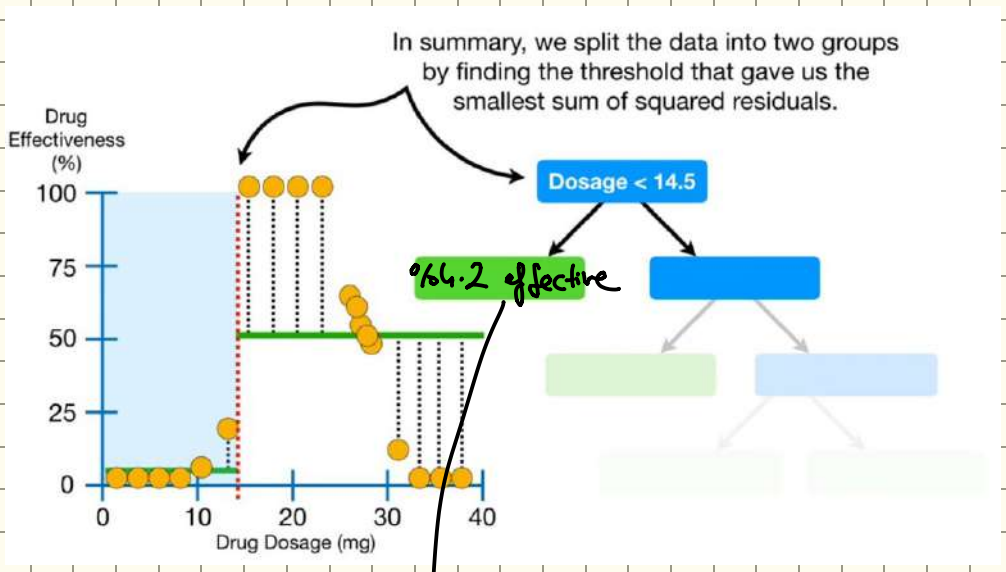
w_i → $\frac{\text{child num}}{\text{parent num}}$



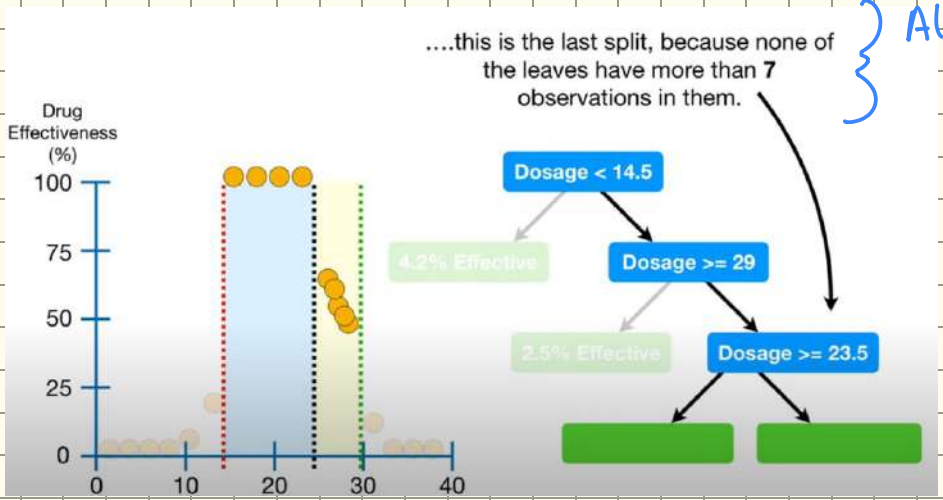
BIAS - VARIANCE TRADEOFF.

Regression Trees.

Building trees:



We set min_num_split as 20 to prevent overfit. So this node will be a leaf. (left pods average is %64,2.)



Also to prevent overfit.

→ Burden bağlı feature ile yapmak için hepsinde ilk root için ikişer gibi olan değer ölçülür. (Sum of squared residuals) - Min olan root olarak seçilir.

min error.

"Best" candidate for the root"

→ Overfitting engellemek için pruning:

Prepruning

- used before construction of tree
- can be done using Hyperparameter tuning
  ↳ eniyi skor için eniyi parametreler seçimek (max_depth, min_sample_split) gibi
→ faster

Postpruning

↳ used after construction
↳ used when tree has very large depth
→ more accurate

dahi var bak stackquest?

Advantages of DT:

- Non parametric
- Classification and regression
- Classification fast once rules are developed.

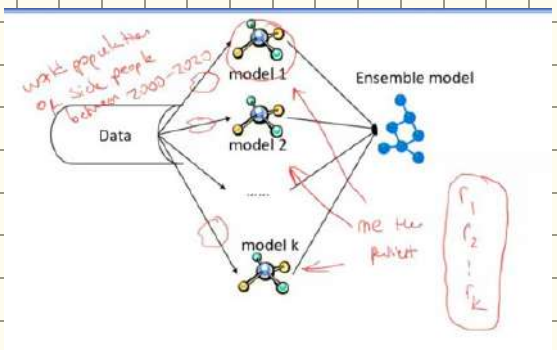
Disadvantages of DT:

- Tend to overfit even pruning.
- Instability

ENSEMBLE LEARNING

Combine multiple models together

- Committees – Bagging
  - *Regression*: Take an average of the predictions made by each model
  - *Classification*: Make classification by voting over a collection of classifiers
- Boosting – Adaboost
  - Train multiple models in sequence
- Decision trees
  - Different models are responsible for making predictions in different regions of input space



✱ Majority vote.

✱ Combine multiple models.

✱ Combining diverse, independent opinions.

- How can I get different models ?

→ different data + same models } Bagging (Bootstrapping)

→ same data + diff. models → not good idea } Boosting

→ change feature space for each model. (sampling features)

→ Randomness

↳ Random forest.

✱ Choose random features and calculate the best split (gini)

If I have enough training data:

- How can I get different (independent) models ?

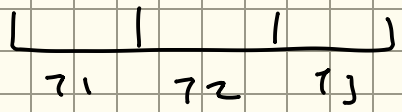
1-) Different dataset. (train data)

$T_1 \rightarrow$ (BML) $\rightarrow M_1$ / P_1

$T_2 \rightarrow$ .. $\rightarrow M_2$ / P_2

$T_3 \rightarrow$ .. $\rightarrow M_3$ / P_3

$T_1 \cap T_2 = \emptyset$ $T_1 \cap T_3 = \emptyset$ $T_2 \cap T_3 = \emptyset$



Performance of each models?

$T_1 \cup T_2 \cup T_3 \rightarrow$ (BML) $\rightarrow M$ / P

P > P_1 P_2 P_3 → because they have less data

↓ combine data.

$P_{1,2,3}$ > P_1 P_2 P_3

↓ Majority voting

P ? $P_{1,2,3}$

Tradeoff

Majority vote yapınca belki P_1 doğrulukları P_2, P_3 %20 olabilir. Bu yüzden $P > P_{1,2,3}$ olabilir.

↓

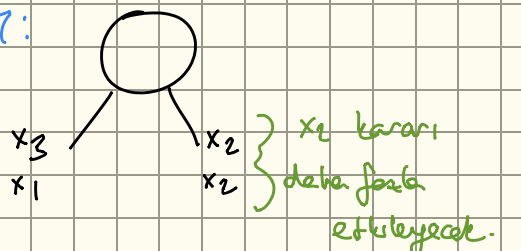
Let's make Subsampling

x_1, x_2, x_3, x_4 M_1

x_3, x_1, x_2, x_2 M_2

SVM: Data pointlere yakın soft margin belirlediğimiz için bir önemi olmadı.

DT:

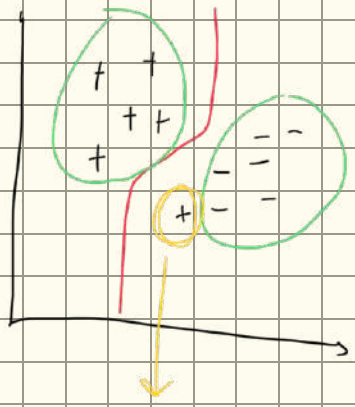


1NN: etkilenir.

3NN: x2'ye 2 kere rastladığımız datada test data x2'ye yakın gelirse etkilenir.


Samples
M1
S Samples
M2
S samples
M_L
Bootstrapping
Bootstrapping
Building dataset with same amount with randomly selected samples with replacement
Bootstrapping → Sampling (random) with replacement
Bagging (Bootstrapping + Aggregation)
Regression: Take ave. of the predictions
Classification: Majority voting

2-) Use different models → Boosting



- AdaBoost
  - Initially, set uniform weights on all the records
  - At each round
    - Create a bootstrap sample based on the weights
    - Train a classifier on the sample and apply it on the original training set
    - Records that are wrongly classified will have their weights increased
    - Records that are classified correctly will have their weights decreased
    - If the error rate is higher than 50%, start over
  - Final prediction is weighted average of all the classifiers with weight representing the training accuracy

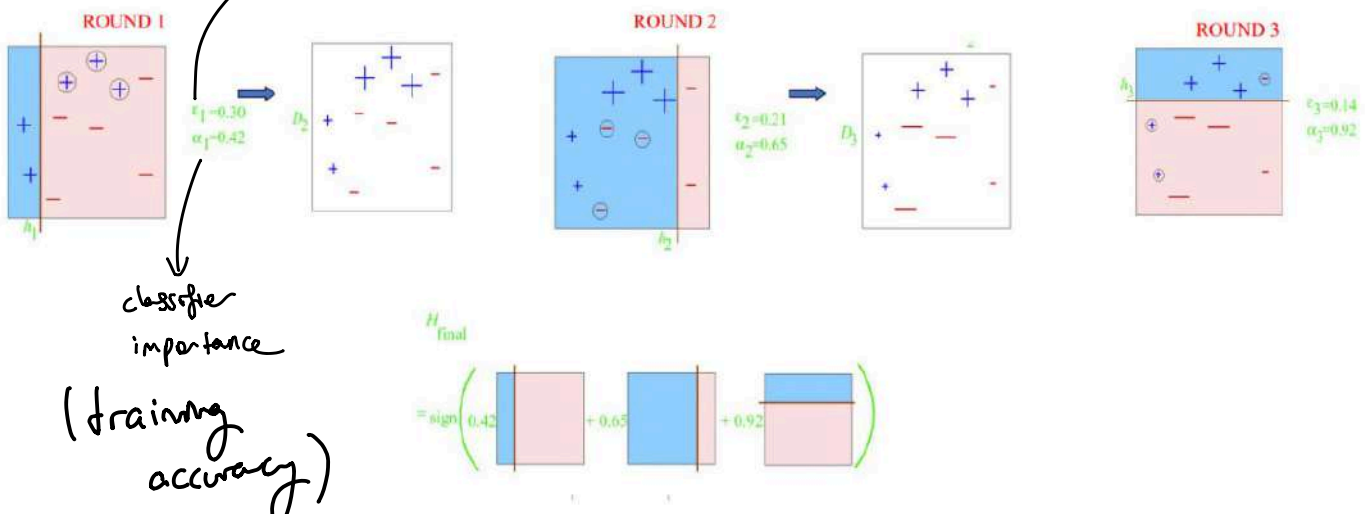
replicate this more for other bootstrapping.

Its gonna concentrate on this data → boosting (problem areas)

- Boost weak learners to a strong learners.

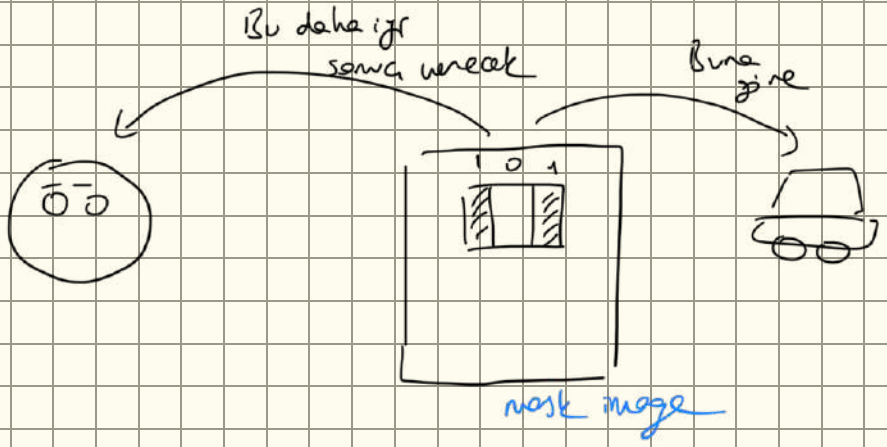
Example: AdaBoost

classifier error $(\frac{3}{10})$



classifier importance (training accuracy)

Boosting example: face detection.

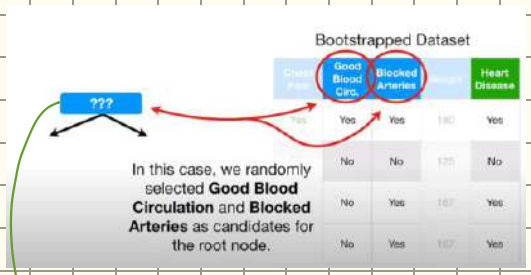


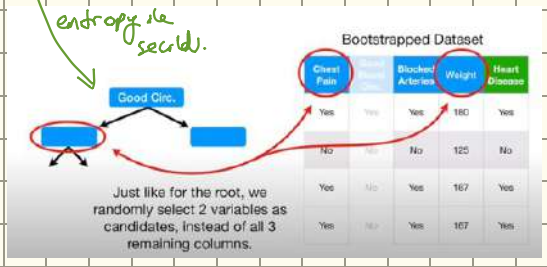
Sonuç iyi

RANDOM FORESTS

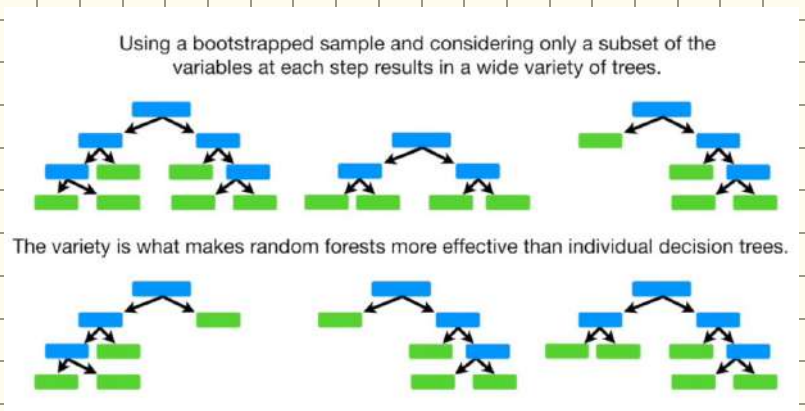
We build a tree:

- Bootstrapped dataset
- Only considering a random subset of features each step.





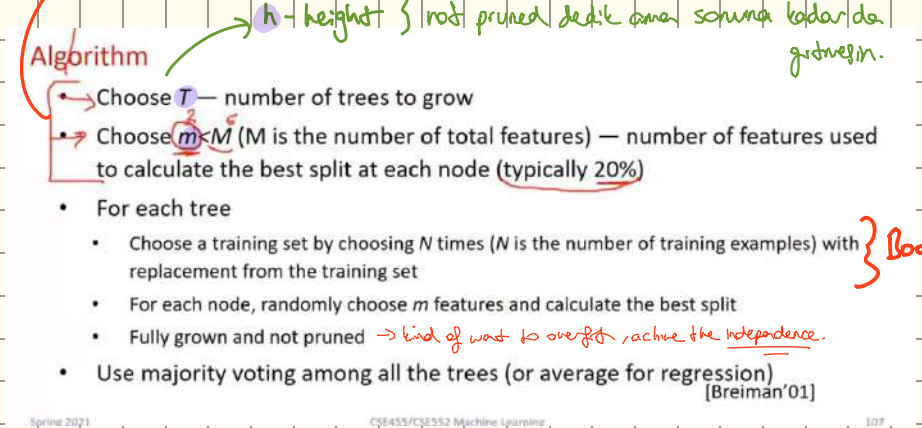
Repeat: Make a new bootstrapped dataset and build a tree with random features each step.



→ then majority vote

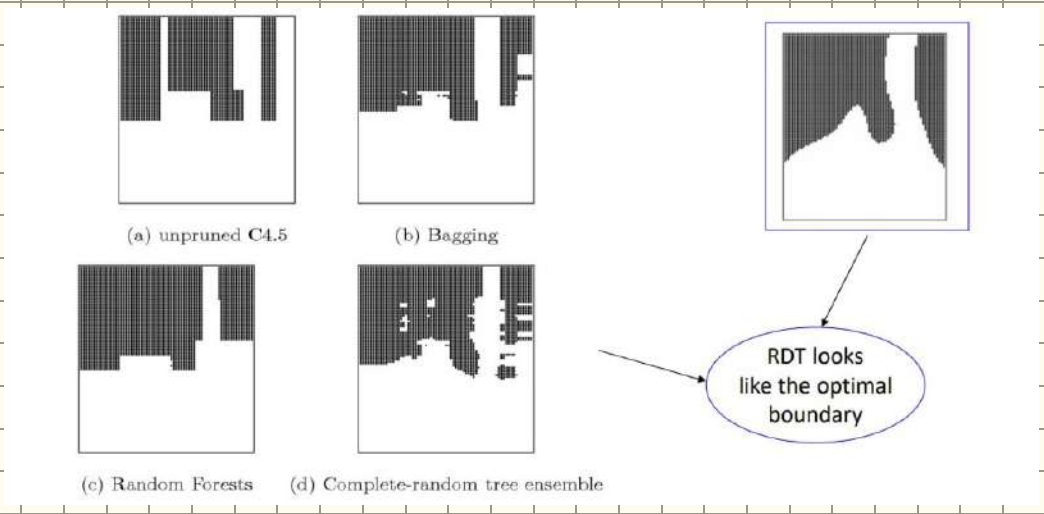
- different dataset
- different models
- subset of features
- randomness

hyperparameters



→ Improve accuracy: Diversity, reduce variances

→ Improve efficiency: Searching among subsets of features is much faster than searching among the complete set.



Decisions
$x_1 > 0$

You can make "multivariate" decisions!

$ax_1 + bx_2 + c > 0$

RF with Clustering:
(unsupervised learning)

apples
pears

Images → Feature Extraction → { color, shape, size, ... }

{r, g, b} {sq, ..} {1, .., 10}

first image → {r, oval, 5 .. } → Label 1

resample (pick smth different)

{b, circle, 7}

Label 2

- RF to solve unsupervised learning problems, in particular, clustering problems and missing value imputation in the general sense
- RF generates a synthetic target variable in order to proceed with a regular run:
  - Give class label 1 to the original data
  - Create a copy of the data such that each variable is sampled independently from the values available in the original dataset
  - Give class label 2 to the copy of the data
  - RF on two class → dissimilarity measure for clustering...
  - Note that the second copy has marginal distributions identical to the first copy, whereas the possible dependency among predictors is completely destroyed
  - A necessary drawback is that the resulting dataset is twice as large as the original

SUPERVISED ML

Feature Reduction, Mapping high dimensional data to lower-dimensional space without losing essential information.

Save problem

100 features.	5 features.
$y = w^T x + b$ (100 parameters) $+1$	$y = w^T x + b$ ($5_{param} + 1$) } simpler model (less accuracy) I can have
- more jobs to do	- computational ↓
	- less things to find ✓ optimization ican iyf.
	- simplicity

- faster
- noise can handle better.
- can improve accuracy (prevent overfit)

f_1 f_2 . . . . f_{100} sensors

f_2 → ? is it needed

standard deviation = 0 or low } its same data } not useful because no variation

or F_2, F_3 is highly correlated. I dont need both of them.

→ Given a set of points, compute linear transformation

$$G \in \mathbb{R}^{d \times k} : x \in \mathbb{R}^d \rightarrow y = G^T x \in \mathbb{R}^k \quad (k << d)$$

originally were d dimensional space

reduced to k dimensional space.

Why Reduce Dimensionality?

1. Reduces time complexity: Less computation
2. Reduces space complexity: Less parameters (compression)
3. Saves the cost of observing the feature
4. Simpler models are more robust on small datasets
5. More interpretable; simpler explanation
6. Noise removal
7. Data visualization (structure, groups, outliers, etc) if plotted in 2 or 3 dimensions

"Curse of dimensionality"

$x \xrightarrow{f} y \rightarrow$ classification

When your class. problem is in linear space but reduced one gives you non-linear space, it may not be a good idea to reduce it.

- **Feature selection:** Choosing $k<d$ important features, ignoring the remaining $d-k$
  - Subset selection algorithms
- **Feature extraction:** Project the original x_i, $i=1,\ldots,d$ dimensions to new $k<d$ dimensions, z_j, $j=1,\ldots,k$
  - Principal components analysis (PCA), (unsupervised)
  - Linear discriminant analysis (LDA), (supervised)
  - Factor analysis (FA) ...

Subset Selection:

$d=5$

$\{x_1, x_2, x_3, x_4, x_5\}$

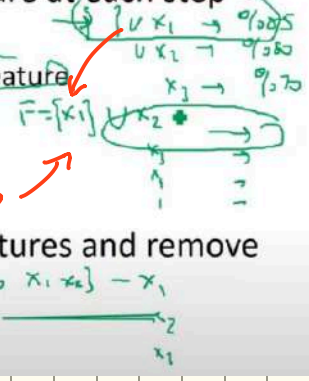
2^d subsets = ↓ subset → classification → ↓ measure

make this a few times until to get best subset

Complexity:

2^d çok fazla. Hadi make with ~~greedy~~ approach!

- **Forward search**: Add the best feature at each step
  - Set of features F initially $\emptyset$
  - At each iteration, find the best new feature $j = \arg\min_i E(F \cup x_i)$
  - Add x_j to F if $E(F \cup x_j) < E(F)$
  - Greedy $O(d^2)$ algorithm → nested loop
- **Backward search**: Start with all features and remove one at a time, if possible
- **Floating search:** Add k, remove l

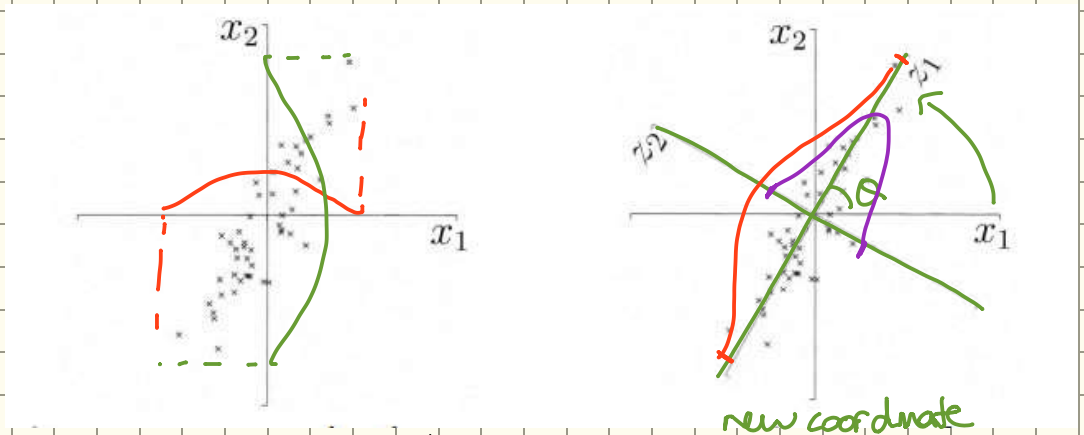


⚠ feature selection çok değişik edilmiş. $\{x_1, x_2\}$ x_3 ile bir bağı varsa, greedy algo. bunu görmeyebilir. There are non-heuristic methods that can find those type of linear dependencies))

PCA (Principal Component Analysis):

✱ Reduce the dimensionality of a dataset by finding a new set of variables, smaller than the original.

↳ Still retains most of the information.



$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

iki si de geniş bir alan kaplıyor. Kolayca reduction yapamayız.

new coordinate system

linear transformation

$$z_1 = \cos\theta \cdot x_1 - \sin\theta \cdot x_2 = w_1^T x$$

$$z_2 = \sin\theta x_1 + \cos\theta x_2 = w_2^T x$$

$\sigma(z_1) > \sigma(x_1)$ or $\sigma(x_2)$ } I have a new feature has bigger variance than old ones.

$\sigma(z_2) << \sigma(x_1)$ or $\sigma(x_2)$ } variation is smaller.

$\{x_1, x_2 \cdots, x_n\} \in \mathbb{R}^d$

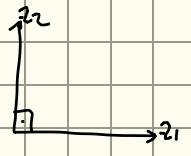
$z_1 = a_1^T \cdot x_i$ $\quad (z_1 = w^T x)$

Linear Combination.

where $\text{var}[z_1]$ is maximum.

a_1 is an eigenvector of S corresponding to the largest eigenvalue $\lambda = \lambda_1$

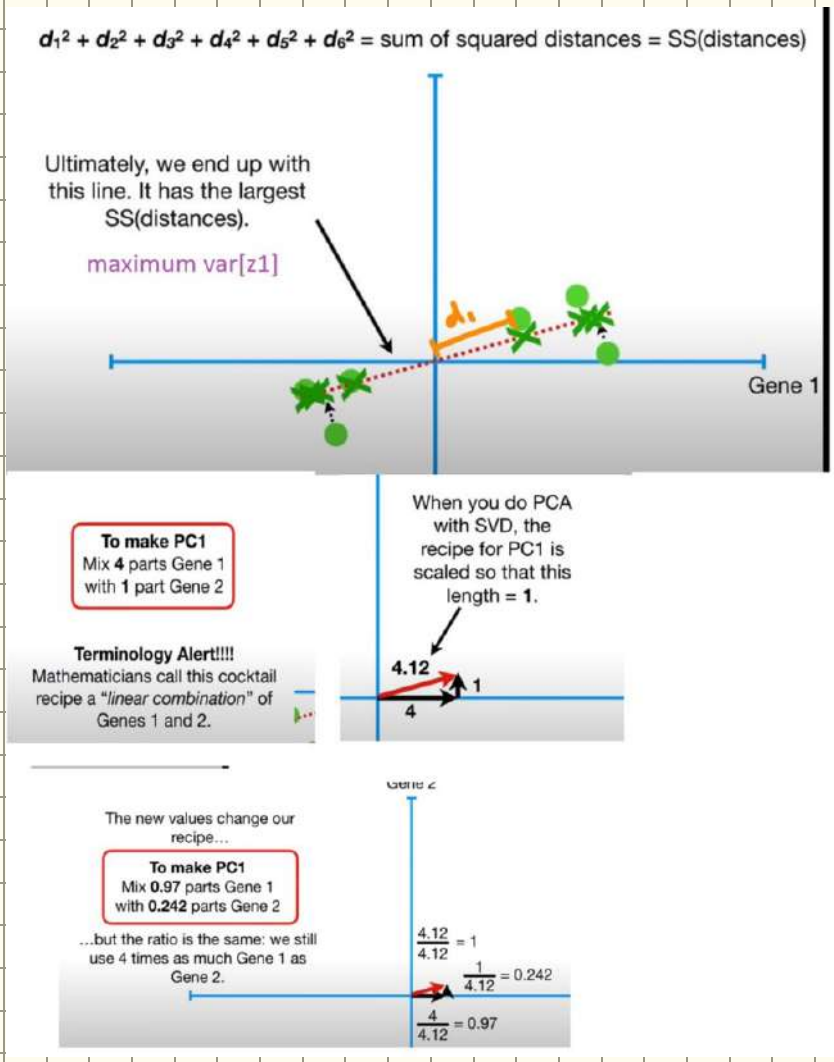
To find next vector a_2 maximizing $\text{var}[z_2]$, subject to $\text{cov}[z_2, z_1] = 0$ } uncorrelated

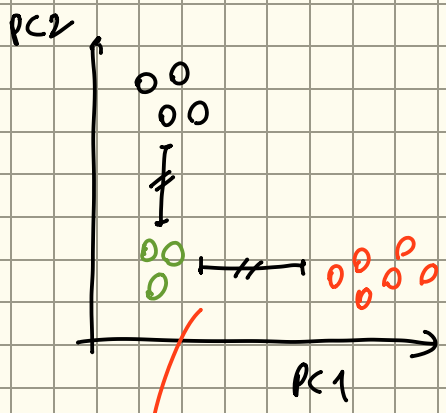


a_2 also eigenvector of S whose eigenvalue $\lambda = \lambda_2$ second largest.

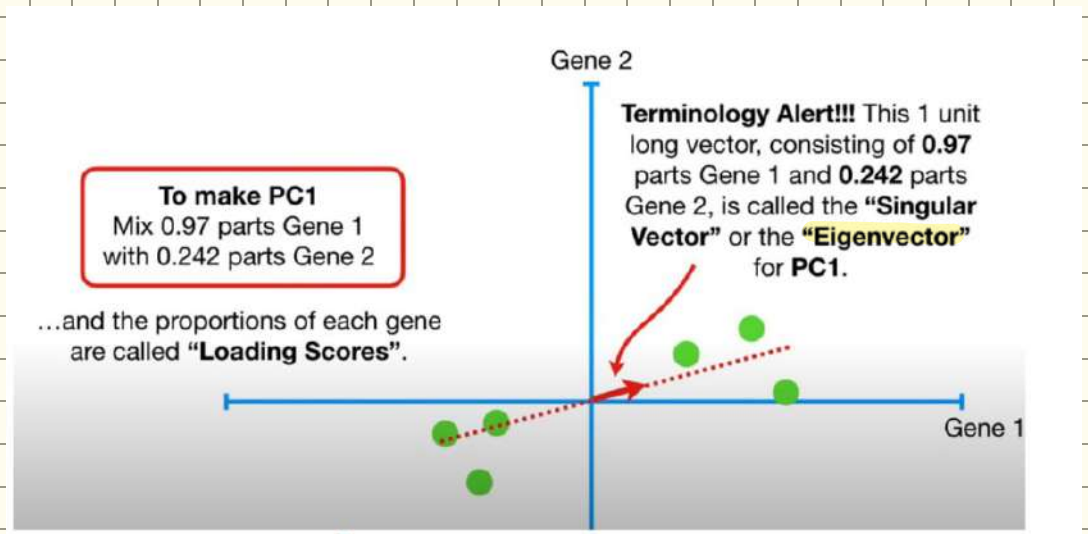
In general;

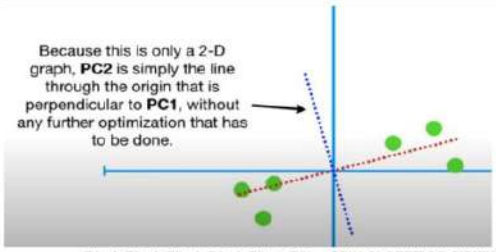
$\text{var}[z_k] = a_k^T \cdot S \cdot a_k = \lambda_k$





Kırmızı ve yeşilin farkı, yeşil ve siyahtan daha fazla.



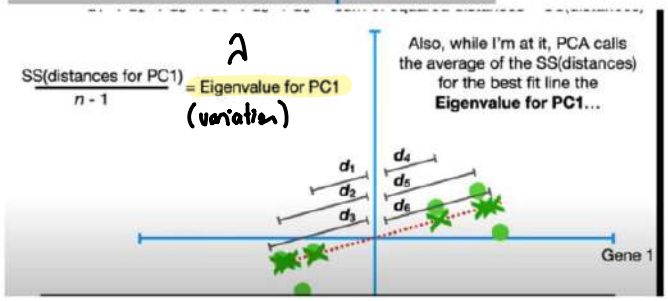


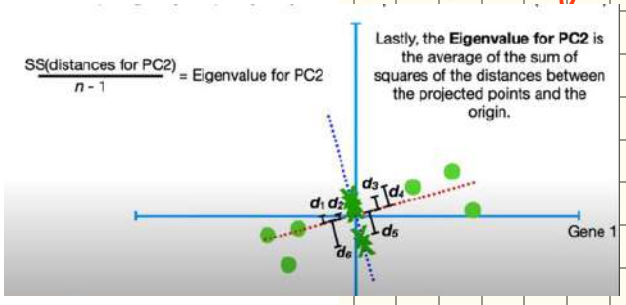
$Ax = \lambda x$

↓ ↳ find eigenvalue.

find eigenvector

"largest eigenvalue"

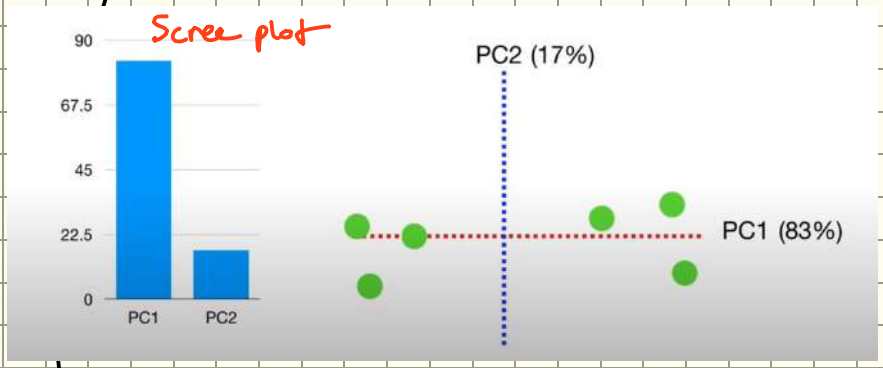




* Eigenvalues are just measures of "variation"

Rotate

Find variations



covariance matrix

$$\text{var}[z_k] = a_k^T S a_k = \lambda_k$$

$$S = \frac{1}{n} \cdot X \cdot X^T \in \mathbb{R}^{d \times d}$$

$$X = [x_1, x_2 \cdots, x_n] \in \mathbb{R}^{d \times n}$$

Obtain eigenvectors of S by computing the SVD of X:

MNIST digit recognition.
10 x 4k → 40,000
30x30 → 900 features

$$X = U \Sigma V^T$$

$$z_{k \times 1} = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_k^T \end{bmatrix}_{k \times d} X_{d \times 1} \quad (k << d)$$

All vectors

How can I find k value?

When $\text{var}[z_{k+1}] = 0$ stop. (We ignore this, useful olmayabilir.)

Ama her zaman 0 olmayabilir.

$\vec{x} = [x_1, x_2, x_3]$

$x_3 = a x_1 + b x_2$ olduğu durumda

$\lambda_3 = 0$ olur, PCA bunu hesaplayabilir.

Ama eğer noise varsa?

$x_3 = a x_1 + b x_2 + a \xi_1 + b \xi_2$

Değişen değerler var.

PCA hesaplayamayabilir.

Assume $\bar{x} = 0$

Form the matrix: $X = [x_1, x_2, \cdots, x_n] \in \Re^{d \times n}$

then $S = \frac{1}{n} X X^T$

Obtain eigenvectors of S by computing the SVD of X:

$$X = U \Sigma V^T$$

* Step "elbow" of scree graph.

$x_k' = A_{k \times d} \; x_d$

orthonormal transformation

$x_d' = T \times d$

variations decreasing, when 0 ignore it.

- Proportion of Variance (PoV) explained

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

when λ_i are sorted in descending order

- Typically, stop at PoV>0.9

$X \in \mathbb{R}^{d \times n} \rightarrow G^T \cdot X \in \mathbb{R}^{k \times n} \rightarrow \bar{X} = G(G^T \cdot X) \in \mathbb{R}^{d \times n}$

↓ Reconstruction

$\vec{x} = [x_1, x_2, x_3]$

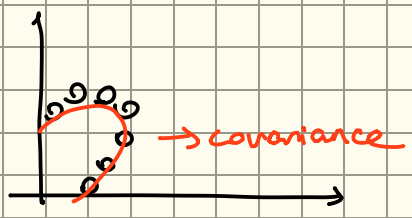
$x_3 = \underbrace{a x_1 + b x_2}$ doğru durumda

$(x_1, x_2, x_3) \xrightarrow{PCA} (z_1, z_2)$

↓

Because there are linear relationship.

$x_3 = x_1^2 + x_2$ → not linear? PCA



$(x_1, x_2, x_3) \rightarrow (x_1, x_1^2, x_2, x_2^2, x_3, x_3^2)$

Now, I can reduce them.

↓ PCA

(x_1, x_1^2, x_2)

high dimension

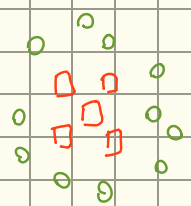
apply non-linear trans. to high-dimensional space.

↓

restricted set of trans. with good computational properties

↓

"kernel trick" $(k\alpha = \lambda\alpha)$



PCA → No loss in lower eigenvalues

↓

Pick a kernel function RBF (to high dimension)

↓

Does it now have lower eigenvalues? No

$x_i = \{t, h, l, p, w\}$

reduce to ↓ — weekday, not gonna include it → categorical

$x_i' = \{t, h, l, p\}$

→ leave w aside and deal with real data.

$[t, h, l, p, w] \rightarrow [f_1, f_2, w], y$ — push it to the row → (w)

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = A \begin{bmatrix} t \\ h \\ l \\ p \\ w \end{bmatrix}$$ } PCA does.

→ I can't multiply with w. Not include w in this process.

t	h	p	l	w
30	50	1000	1	Mon
⋮	⋮	⋮	⋮	⋮

] 1000 data

real numbers ($\mathbb{R}$) — categorical

PCA says take this measurements

X^T (1000 x 5 matrix)

$X \cdot X^T \rightarrow$ eigen values. (PCA)

$x_1 \cdots x_5$ (≈0)

eigen vector

$e_1 \cdots e_5$

$$\vec{x_i} = \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \\ e_4^T \end{bmatrix} \begin{bmatrix} t \\ h \\ p \\ l \end{bmatrix}$$

4 features, 1000 data

$x_5 = 0$ → we dropped it.

w' → greater, says you consider

New feature 1: $\alpha_1 \cdot t + \alpha_2 h + \alpha_3 \cdot p + \alpha_4 l + \alpha_5 \cdot w'$

$1.2 * w'$ → is this meaningful?

Numerically it will give you smth, but that doesn't mean anything semantically.

✗ In PCA, we drop the categorical values, add them later.

F_1	F_2	F_3	F_4	F_5	F_6

(F_1–F_4: $\mathbb{R}$, F_5–F_6: C) → same number of rows →

F_1'	F_2'	F_5	F_6

PCA — 2 features are enough.

SVM, KNN ✓, DT ✓, RF ✓

$F_1' > a_1$, $F_5 = c_1$

This is better

DT → $4 * (n-1)$ comparisons (interval) + $2 * k$ (num of iterations)

after PCA: $2 * (n-1) + 2 * k$

- computational advantage
- easier to model (6 feature to 4 feature)

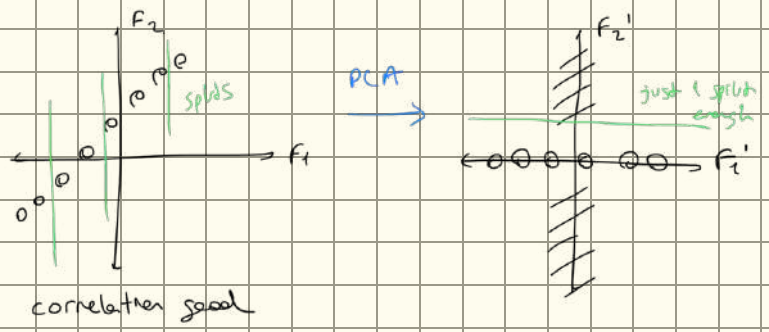
Faydaları

1-) Faster.

2-) When there is noise, PCA give clean data in next step.

3-) high dimensional overfit easily. prevent overfitting

Can we do this with DT?



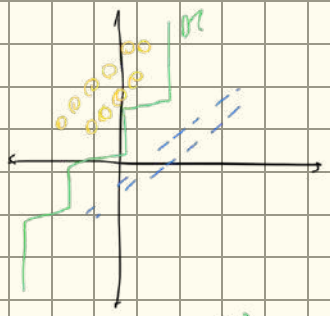
correlation good

which one is better?

if ($f_1 > a_1$)

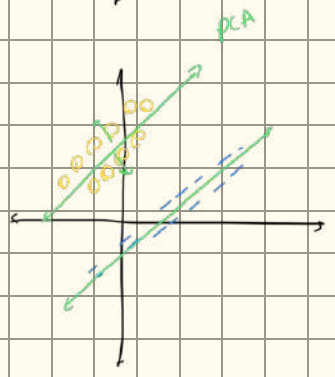
if ($f_2 > a_2$)

Binary classification problem:

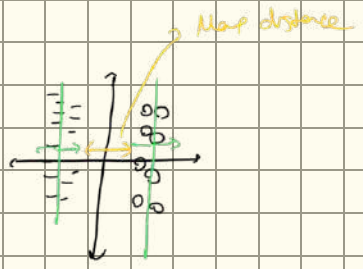


SVM makes very well also.

Lets think DT.



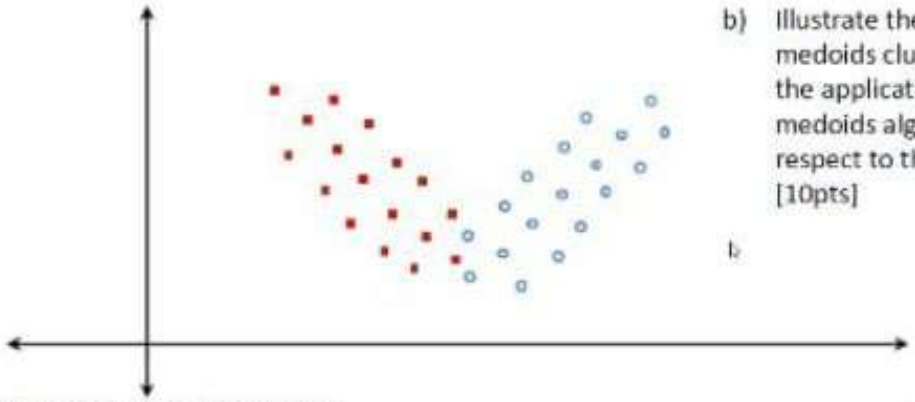
→



-2 classes

If I rotate this, distance calculation

a) Given the following samples in a 2D Euclidean space, what does principal component (PCA) analysis reveal about the data? Show the principal components graphically. Explain your answer. [10pts]



b) Illustrate the first two iterations of k-medoids clustering algorithm? Compare the application of k-means and k-medoids algorithms on this data with respect to their runtime complexities. [10pts]

b.

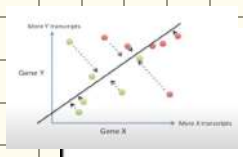
find the a_1 vector which maximizes $\text{var}[z_1]$

LDA (Linear Discriminant Analysis)

→ PCA → reduces dimensions with focuses on with most variation ones.
- no class / labeled data (unsupervised)

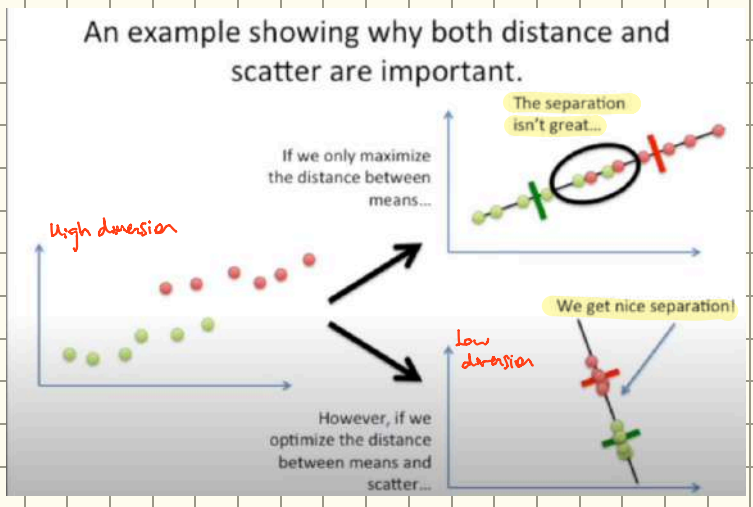
→ LDA → maximizing the seperability between the known categories.
- labeled data (supervised)

LDA projects the data onto this new axis in a way to maximize the seperation of 2 categories.

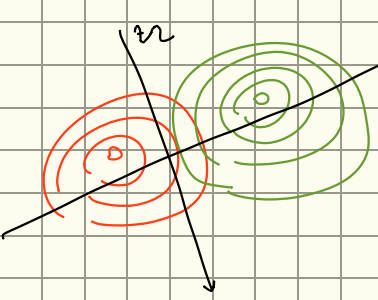


→ Maximize the distance between means, while minimizing the scatter.

$$z = (w_1\ w_2)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

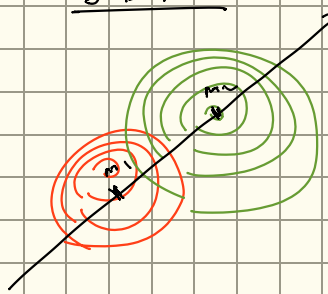


PCA



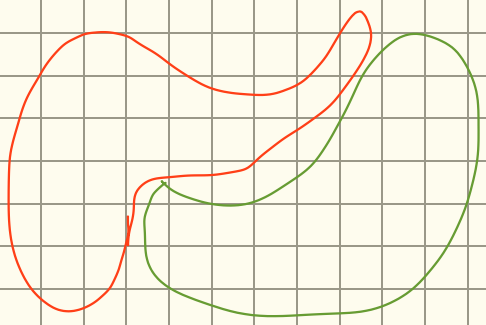
with max variation we seperate them not knowing labels!

LDA



Now, it looks seperate categories means and projects them on axis.

we know labels!



→ LDA can't seperate good in here.

More then 2 classes:

Fishers linear Discriminant: LDA with 2 classes.

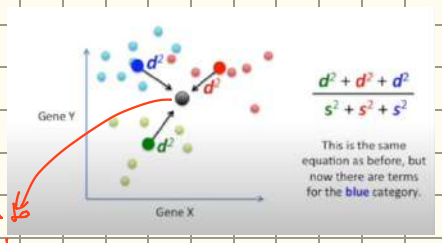
Again, maximize the distance means, minimize the scatter:

Find w that maxp

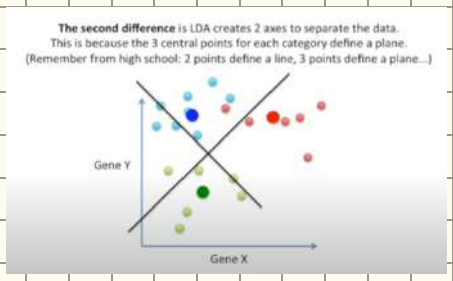
$$J(w) = \frac{w^T \cdot S_B \cdot w}{w^T \cdot S_W \cdot w} = \dots$$

(max over S_B, min over S_W)

$$w = c \cdot S_W^{-1} (m_1 - m_2)$$



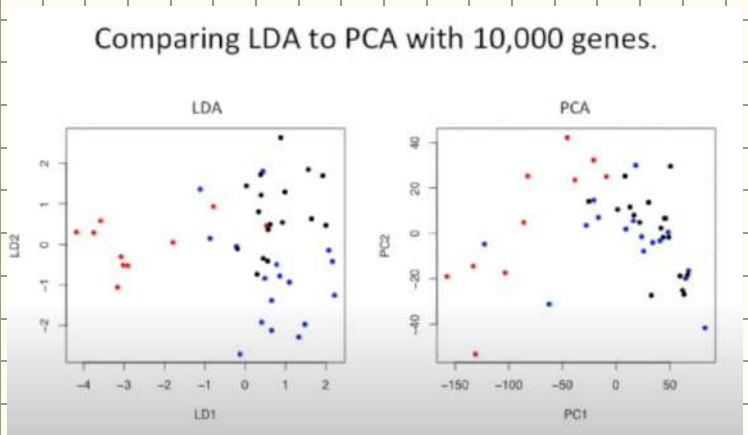
Central to all data.



- Only applicable with linear problems.

→ LDA doesnt have "soft margin" option like in SVM.

- PCA looks at the data with most variation
- LDA maximize the seperation of known categories



Eigenfaces: PCA application to specific case.

Calculation of eigenfaces:

- Calculate average face v

Mean image: μ / top eigenvectors: $u_1, \ldots, u_k$

$$I_i = \alpha_0 \mu + \alpha_1 u_1 + \alpha_2 u_2 + \cdots$$

} mean image + linear combination of images

$$\mu = \sum_i a_i \phi_i$$

a_i: some values

μ: recalculate images

→ I have principal component of faces, then I can make your face with some parameters.

μ is common to everything.

$[w_1, \ldots, w_k]$ a signature of the faces.

$\hat{x}$ = $\mu + w_1 u_1 + w_2 u_2 + \cdots$

↓ your face.

$w_1 u_1$: right amount of each component.

$= \mu + [w^T] \begin{bmatrix} u_1 \\ u_2 \\ \vdots \end{bmatrix}$

$k < d$ feature extraction } signature part

dimension

other parts are common.

You can make a new face with this.

→ I'm just doing simple analysis of data.

$[1.2 \quad -1.0 \quad 1.5 \quad 0.2]$ → this is my face

↓

if I increase this a little

I can make new face similar to me.

PCA part

$x_1 \cdots x_N$ → our data of faces.

↓

μ, Σ } covariance matrix

↓

$u_1 \cdots u_k$ } most k eigenvalues relevant for us

↓

We can represent new data as:

$x_i \rightarrow \mu + w_1 u_1 + w_2 u_2 + \cdots$

x_i: 10k dimension

$[w_1 \cdots w_k]$: 100 dimension (lower)

DT algo:

Build tree
- ↳ sort
- ↳ take middle points
- ↳ choose best split

Evaluate performances

Choose with best perf

Adaboost algo

Bootstrapped dataset

Train classifier on orginal dataset

Increase weigh weak learners

Decrease " good learners

Better bc errorden leader deren ef weight değişimi

Training accuracy ile modellerin ortalamasını al.

RF algo:

Build trees!

Bootstrapped data

Random feature selection

Majority vote

NEURAL NETWORKS

→ The curved called "Activation functions".

→ Available "Hidden layer"

→ Parameters we multiply } weights
" " add } biases.

→ Neural networks starts with identical Activation functions. → sigmoid, ReLU, softmax

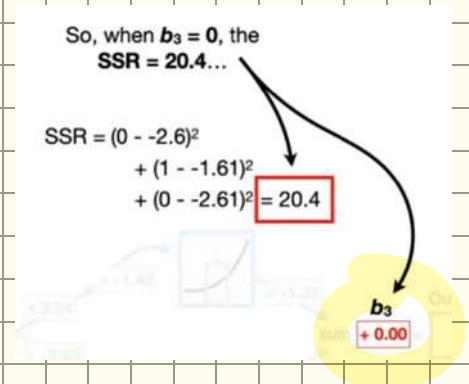
But using weights and biases, it flips and stretches the Act. fune. into new shapes.

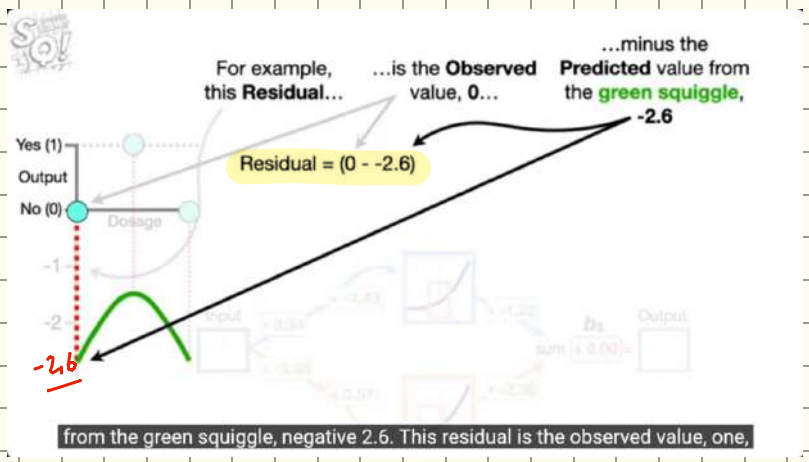
which are then added together to get a squiggle.

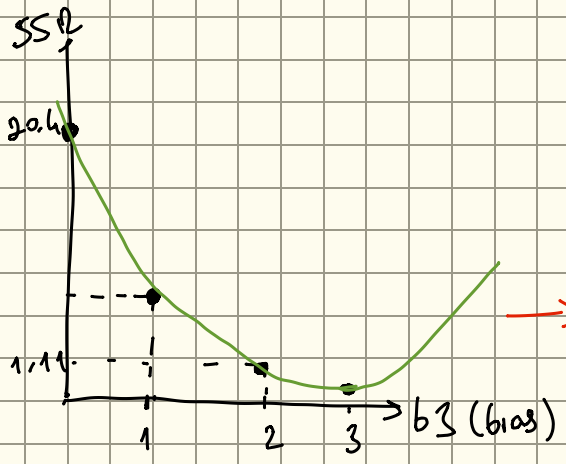
Backpropagation: Predict weights and biases.

→ To quantify how good is the squiggle } Sum of the Squared Residuals.

Residual = Observed - Predicted.







→ Take the smallest and find it with gradient-descent.

$$SSR = \sum (Observed_i - Predicted_i)^2$$

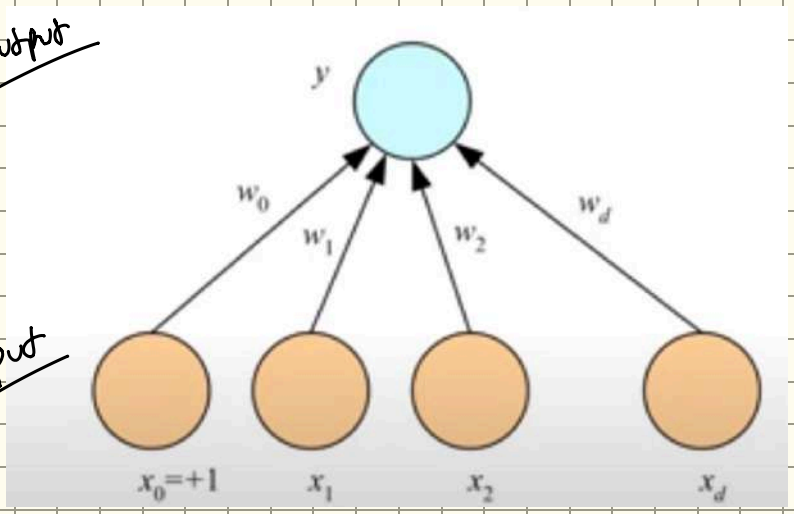
Large numbers of neurons: 10^{10}

Large connectivity: 10^5 ← available.

Parallel processing

Distributed computation / memory: They have to remember

Perceptron!



$y = w^T x + b$ lets assume this

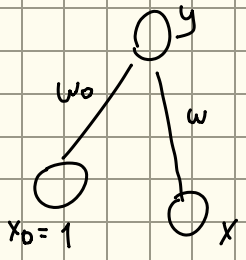
$= w^T x$

$w = [w_0, w_1, \dots, w_d]^T$

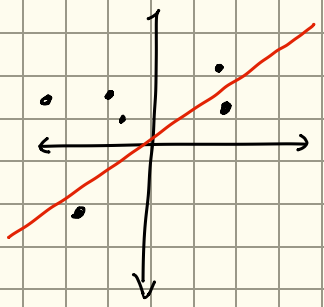
$x = [1, x_1, \dots, x_d]^T$

Can I use it for regression ?

" " classification ?

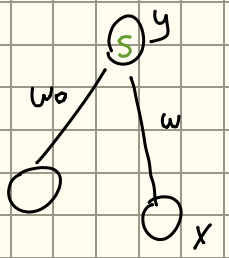


$y = wx + w_0$ → (linear regression)

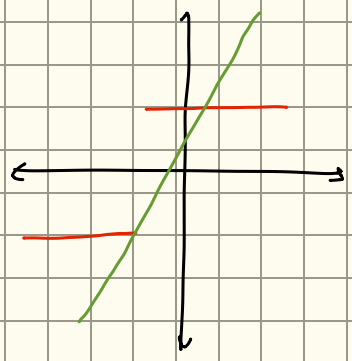


$(x_1, x_2, x_3) \rightarrow w_1x_1 + w_2x_2 + w_3x_3 + w_0$

So, we can do linear regression ✓



$y = 1(wx + w_0 > 0)$ → (classification)

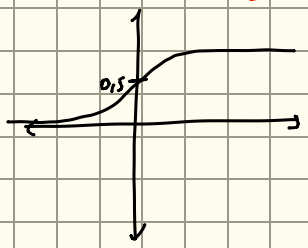


$y \in \{0, 1\}$

↓ Can be continuity problem

Use sigmoid

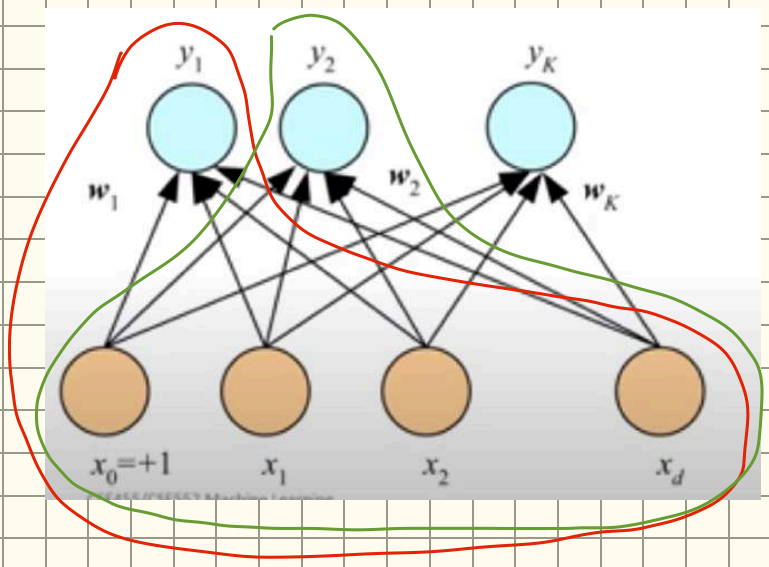
$y = \text{sigmoid}(o) = \frac{1}{1 + e^{-w^T x}}$



So it become $y \in [0, 1]$ (any value between them)

from $\{0, 1\}$ → logistic reg.

Regression with K outputs:



$$y_1 = w_1^T x + w_{10}$$

$$y_2 = w_2^T x + w_{20}$$

$$\vdots$$

$$y_k = w_k^T x + w_{k0}$$

$$y = Wx$$

$$\vec{y}_{(k)} = w_{kxd} \cdot x_{(d)}$$

Classification with K outputs:

More than 2 classes → Generate a categorical value

↳ Map these categorical values to a binary vector

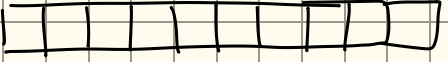
Image → Classifier → {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

(Digit problem)

0 ? 8
0 ? 1
1 ? 8

} Same comparisons
Not ordinal,
categorical numbers.
bcs they are just labels
in this problem.

One hot encoding:

$y' =$ 

binary {0, 1}

So,

"0" = {1, 0, 0, 0, 0, 0, 0, 0, 0, 0}

"8" = = {0, 0, 0, 0, 0, 0, 0, 0, 1, 0}

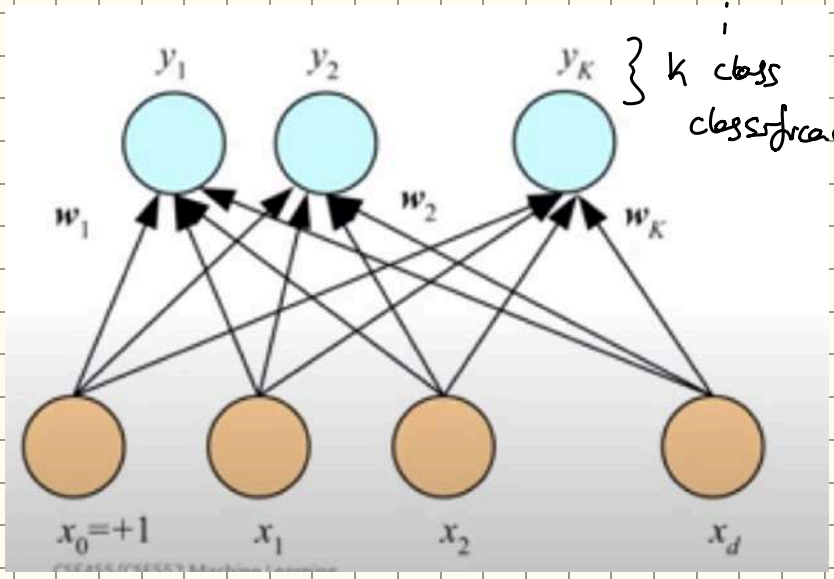
⋮

If you apply this:

1 0 ... 0 → digit 1

0 1 ... 0 → digit 2

⋮



} k class classification

You can do the same thing with regression:

$o_i = w_i^T x$

$y_i = \frac{e^{o_i}}{\sum_k e^{o_k}}$ } softmax

If one of $y_1, y_2 \dots y_k$ are much larger than others e^∞ will take them

↓

$\frac{e^\infty}{e^\infty + e^0 + e^0 + \dots}$

$I \rightarrow [8.0, 1.0, 2.0]$

A way to use signoid function in one-hot encoding.

↓

?? Softmax function (as opposed to signoid.)

For regression: $y = w^T x$

" classification: $y = \sigma(w^T x)$

Training :

- Online learning (instances seen one by one) vs Batch learning (whole sample)
  ↳ red iron vs red apple

$$\arg\min_w \sum_{i=1}^{\hat{}} |y_i - w^T x_i|^2 \rightarrow \text{Batch}$$

↓ simpler.

$$\arg\min_w |y_i - w^T x_i|^2 \rightarrow \text{online}$$

↳ Current model explains the data perfectly.

↓ Calculate the difference btw. apple and iron.

Next time you see apple or iron, try to minimize the difference?

Regression

$$E'(w|x^t, r^t) = \frac{1}{2}(r^t - y^t)^2 = \frac{1}{2}(r^t - (w^T x))^2$$

$$\Delta w_{ij}^t = \eta (r_i^t - y_i^t) \cdot x_j^t$$ per observation

↳ hyperparameter

Update = Learning Factor * (Desired output - Actual output) * Input.

↓ Gradient Descent Update.

↓ error

How big an update should be to correct that mistake.

- If you overreact : (making learning factor too large) not desirable
  ↳ prevent you from learning.

$$\Delta w_{ij}^t = \mu \left(r_i^t - y_i^t \right) \cdot x_j^t$$

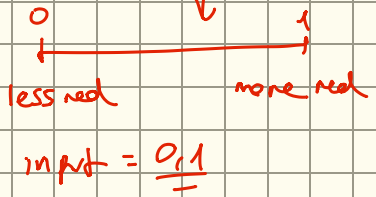
→ Delta to your weight is proportional to your mistake.

(error you make with your current w)

→ If mistake is 0, you learnt it.

→ If " " big, you should update your mistake.

↳ If error (mistake) is big but input is small, you should make small update.



input = 0,1

Because it probably doesn't effect that much.

↳ If error and input is big, I should make big update.

Algorithm

Step 0: Start with a random w. (small)

Step 1: Take a sample (x_i, y_i) error random

$-y_i = w^T x$

- loss = error = $r_i - y_i$

Update w with $w + \mu (r_i - y_i) x_i$

Step 2: Repeat Step 1 as long as necessary.

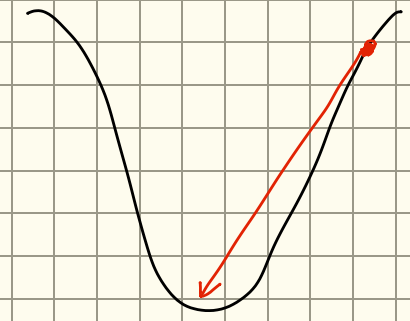
It converges, you can stop (w doesn't change anymore)

randomness

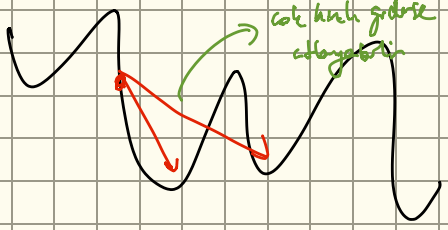
Stochastic Gradient-Descend Alge.

or whenever you get tired

* Regression is easy bcs we are just try to minimize that value.
(gradient-descent minimization)

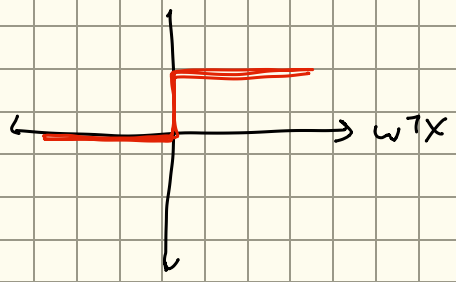


M ne kadar büyük olursa, o kadar hızlı gradient



Gradient descent kontrol altında değilse local minimumda takılır.

Classification:



→ We try to make sure that it is mapped to a logical value (with sigmoid)

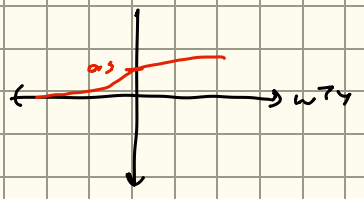
↓

logistic regression

if $(w^T < 0) : 0$

other way : 1

↓ Then we applied some approximation to make continuity.



$y^t = \text{sigmoid}(w^T x)$ → Single sigmoid output.

(linear part)

$E'(w \mid x^t, r^t) = -r^t \cdot \log y^t - (1-r^t) \cdot \log(1-y^t)$ → loss func. I want to minimize

$\Delta w_j^t = \eta (r^t - y^t) x_j^t$

↳ same formula with regression

Logların sebebi: 0,1 ve 0,2 arasındaki hata, 0,2 ve 0,3 'ten daha yanlış.

Bu yüzden sadece r-y bakmak yerine

(1-r, 1-y) 'de bakıyoruz.

↓

cross entropy.

K>2 softmax outputs

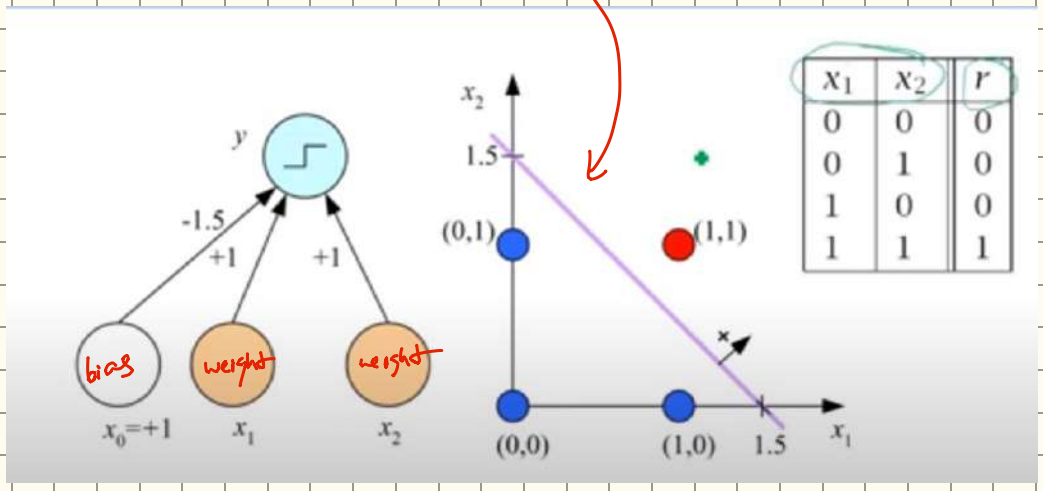
$y^t = -\sum r_i^t \log y_i^t$

$\Delta w_{ij}^t = \eta (r_i^t - y_i^t) x_j^t$

Learning Boolean AND

-> Lets try Basic linear classifiers

$x_1 + x_2 - 1.5 = 0$ } line } $y = \text{sigmoid}(w^T x)$

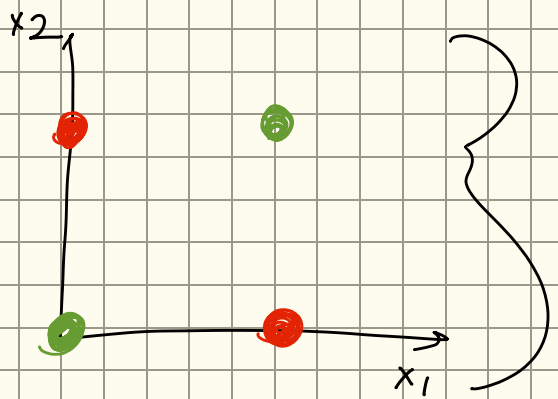


-2 input neural network

XOR

$w_1 x_1 + w_2 x_2 + w_0 \geq 0$

How to draw this line? Infinite



Positive negatives ayri teraflara dusurecek bir line cizelim.

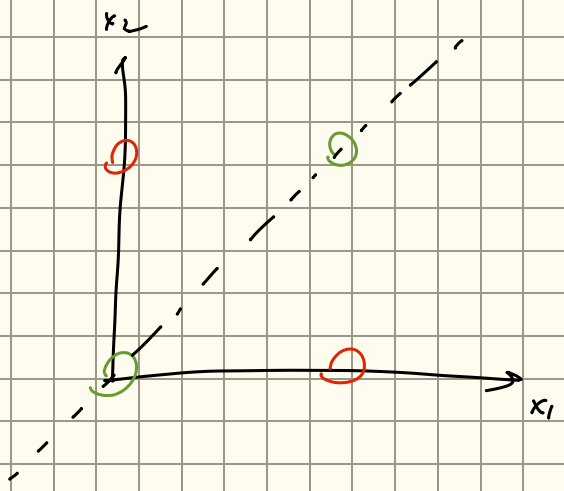
↓

Not a linear problem. } A perceptron cant learn this with any linear classifier

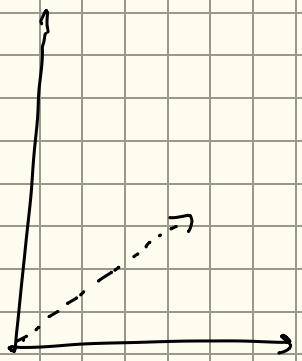
1. ↙ Upgrade to higher dimension

↓ 2. Non linear method

1-)



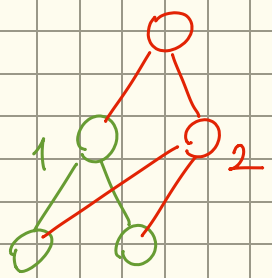
3D space

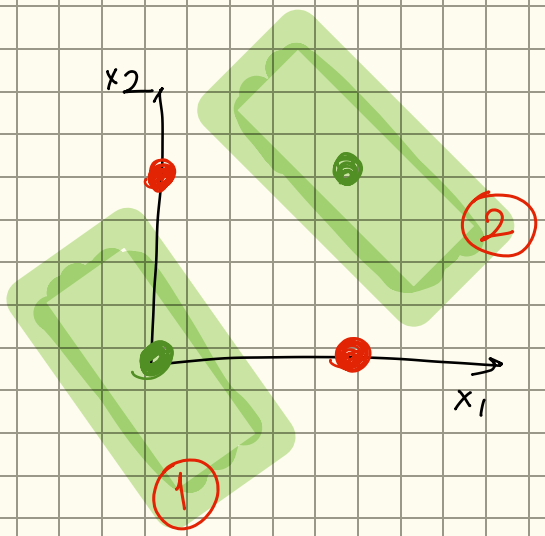


$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \varphi(x_1, x_2)$$

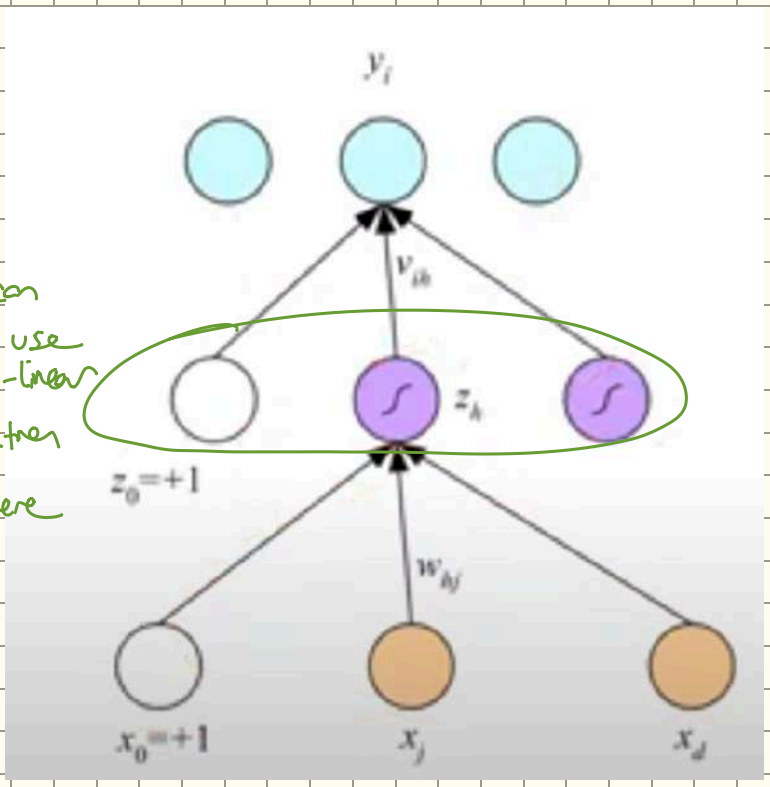
Yüksek boyutlarda tahmin edmeyi zor.

2-) Perceptron itself is a single linear classifier. So, lets combine 2 linear classifiers. (perceptron)

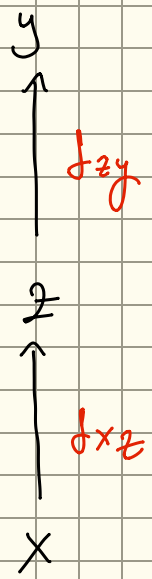




Multilayer Perceptron

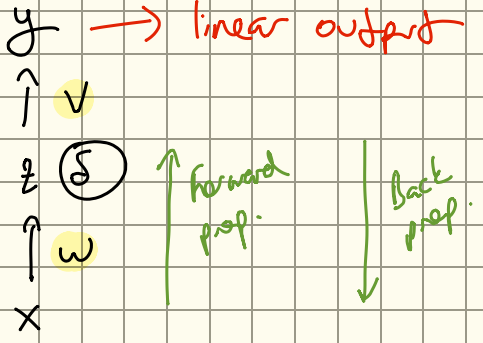


we can only use the non-linear activation func. here



* Multiple classifier combined to get a full classifier and you are doing this at the same time. If it converges, you're done.

How can I minimize the loss in this?



→ Take x and calculate z and y with your current w, v values.

(initialized random)

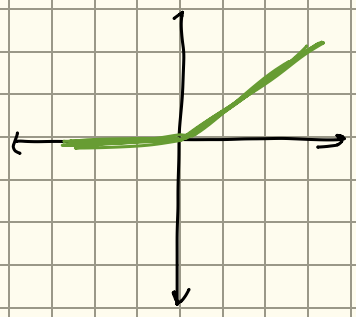
Activation functions: It maps the resulting values in between 0 to 1 or -1 etc.

- Sigmoid: Differentiable $\frac{1}{1+e^{-x}}$

- Tanh: Similar to sigmoid

- Relu $(w^T x) = \begin{cases} \text{if } w^T x < 0 & , 0 \\ \text{else} & , w^T x \end{cases}$

↓ linear



↓ very simple non-linearity

Stochastic Gradient-Descent:

Initialize all v_{ih} and w_{hj} to rand$(-0.01, 0.01)$

Repeat

For all $(\boldsymbol{x}^t, r^t) \in \mathcal{X}$ in random order

For $h = 1, \ldots, H$

$z_h \leftarrow \text{sigmoid}(\boldsymbol{w}_h^T \boldsymbol{x}^t)$

For $i = 1, \ldots, K$

$y_i = \boldsymbol{v}_i^T \boldsymbol{z}$

} Forward prop. Now I have y, z.

For $i = 1, \ldots, K$

$\Delta \boldsymbol{v}_i = \eta(r_i^t - y_i^t)\boldsymbol{z}$

For $h = 1, \ldots, H$

$\Delta \boldsymbol{w}_h = \eta(\sum_i (r_i^t - y_i^t) v_{ih}) z_h (1 - z_h) \boldsymbol{x}^t$

Back prop.

For $i = 1, \ldots, K$

$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \Delta \boldsymbol{v}_i$ // add v to your existing ones

gradient part

For $h = 1, \ldots, H$

$\boldsymbol{w}_h \leftarrow \boldsymbol{w}_h + \Delta \boldsymbol{w}_h$ // add w "

"

Until convergence

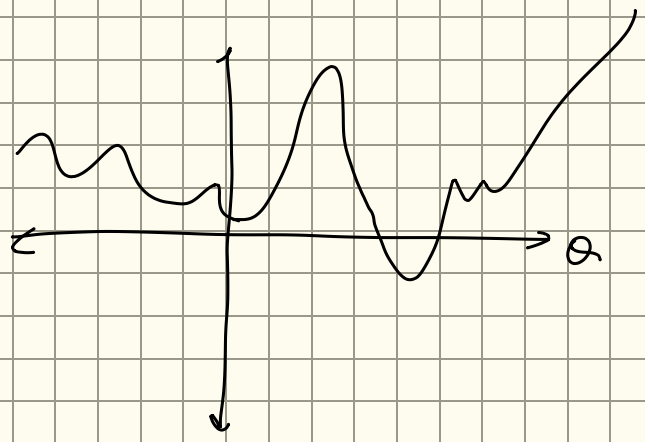
First randomness: Choose small numbers bcs big numbers means you know smth.

Second " : No batch learning. So, we need to decide how to feed the data, and we feed random.

Forward prop + Back prop = Iteration

Use all data in exactly 1 iteration = Epoch

→ Repeat epochs until a convergence.



loss function for input (x_i, r_i)

→ The loss function will be different for input (x_j, r_j)

We hope that our local minimums are closer.

But if my loss function is changing so much depending inputs, I have problem.

↳ Maybe I can use batch learning → But it takes much time

↳ Alternatively, mini batch } n data → take randomly k of it

↓

more stable loss functions

k = n } epoch = 1 iteration

k ↑ ise n kadar dolanırsak

1 epoch tamamlanmış olunur.

n = 100

k = 5

Nasıl iteration yapmak istiyorsun?

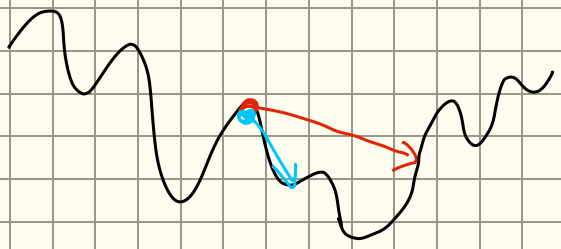
epoch * 100/5

iterations = 100/5 x 100

k seçimi büyük olursa data GPU'ya sığmaz.

M is hyperparameter.

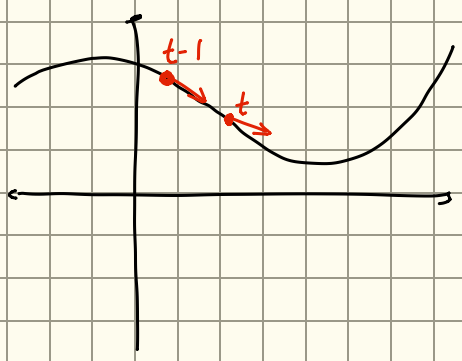
Larger M	Smaller M
- Faster convergence	- Slower "
- less stable	- Stable
- Better to be avoid local min.	- not good with local min.



Strategies to set M:

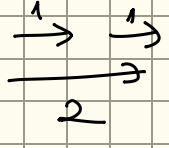
1-) Start with big steps (M↑) and slow it down as you go

2-) Momentum:



- If both of them going same way, go faster
- " " opposite way, go slower.

Gradient yönüne bakarak momentumu ilerletebiliriz.



3-) Adaptive learning rate:

$$\Delta\mu = \begin{cases} +\alpha & \text{if } E^{t+1} < E^{t} \\ -b\mu & \text{otherwise} \end{cases}$$

// If previous step has more error than current step, go faster

ilk başta eğrafta dolanayım.

μ : Mevcut olduğum yerde ne kadar öğrendiklerime göre güncelliyorum

Çok hızlı gidersem hızlanırım
Çok yavaş ", yavaşlarım

↓

Convergence iyileştirmek için

μ ne kadar büyük olursa, o kadar hızlı giderim

Gradient descent fonksiyonu altında dolaşırken bu durumlarda takılır.

4-) ADAM

⋮

* They all depending on your loss function and your model shape

Multiple Hidden Layers:

Theoritically, you can solve any problem with 2 hidden layer MLP as opposed to K layer MLP.

↓

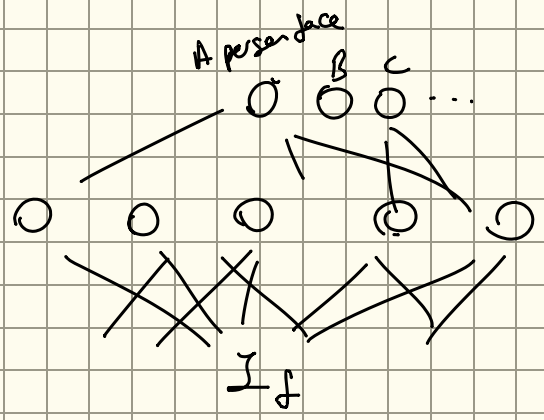
You would need less # of parameters.

MLP$_2$ ↓ $n(\theta_2)$ >> MLP$_5$ ↓ $n(\theta_5)$

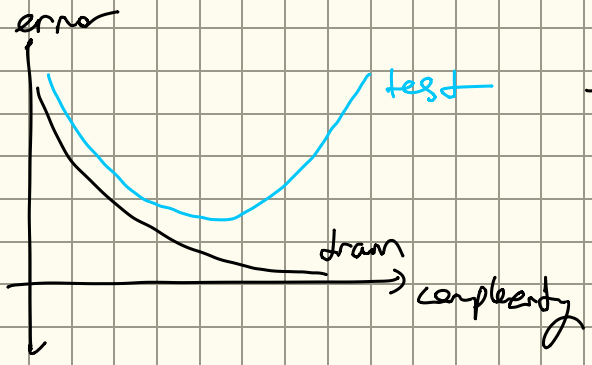
↓

neurons

face detection problem



- I can make this single hidden layer to memorize the faces.



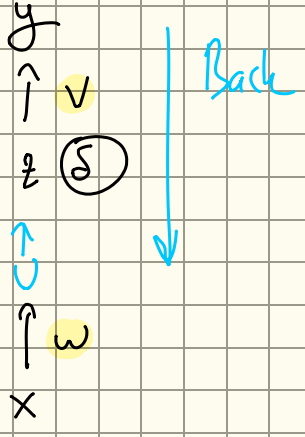
→ Test data can give me high variance in this case.

→ It is not practical.

* So, you can use simpler networks but it will require more complex networks.

* Using multiple layers may lead to simpler networks.

↳ But # of multiplication will increase in back prop.



→ If you multiply number > 1, results gonna increase

↳ 2 × 2 × 2 = very big number

↳ Exploding gradient } If you add very big number

0,2 × n ...

You cannot learn

→ " " " < 1, " " decrease

$0,2 \times 0,2 \times 0,2 =$ ↓ closer to 0.

↳ Dimnishing gradient } When weights become 0, you cannot update the weights

↓

You cannot learn.

✱ In both cases, there are disadvantages of adding more layers.

↓ solution

Deep learning = Artificial neural networks with multiple layers.

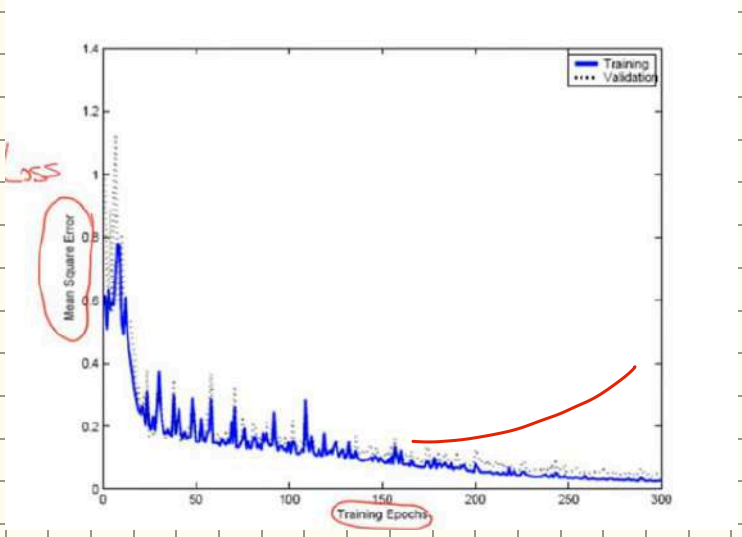
↳ Solutions to the problem of dimnishing gradient, regularization...

AI → ML → NN → DL

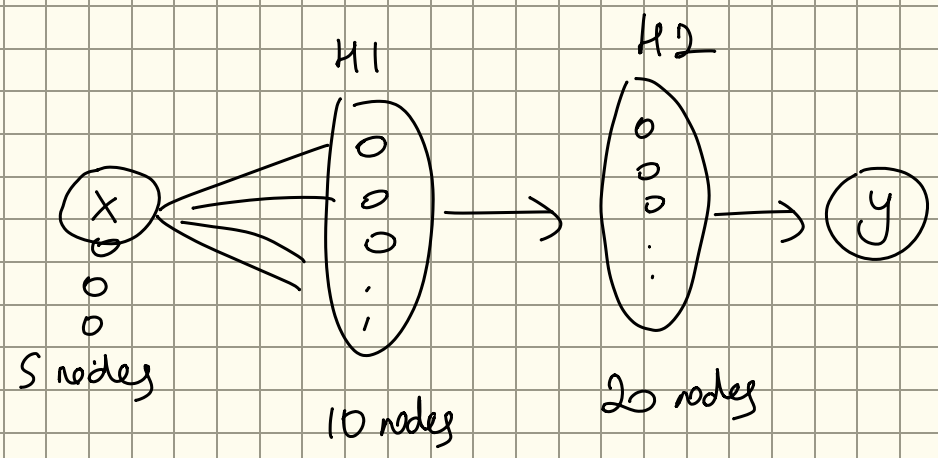
SS

AI

✱ The best regularization is adding more data.



- Validate ile cross validasyon } overfitting



5×10 connections

10×20 "

Solution:

- Structured MLP:

Not fully but repeated connections 10^9 neurons

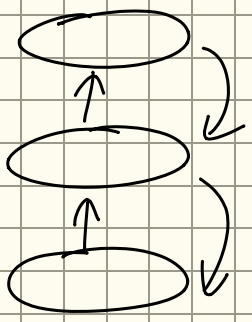
10^5 connections

- Weight Sharing:

- Feed forward Networks = MLP

" "

find its output, feed the next layer.



Back prop.

Updated weights.

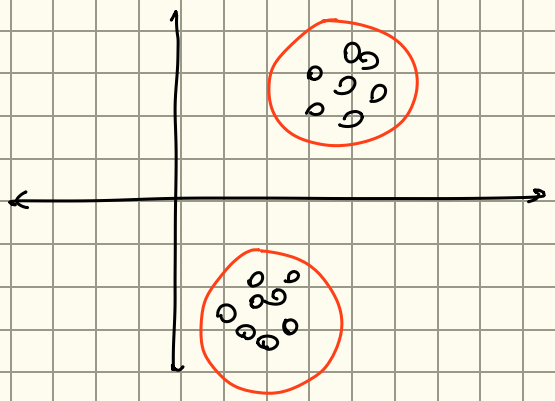
CLUSTERING

$x_i \in \mathbb{R}^d \quad i = 1, \ldots, N$

x_i: # classes

- I don't know how many classes. (fruit types)

Cluster analysis: finding similarities in data



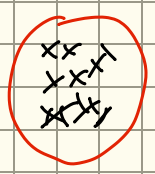
Based on 1 measurement

↓

Distances

Good clustering: High quality clusters

- High intra-class similarity → in cluster
- Low inter-class " → between "



with Euclidean distance

Quality Measures:

Data matrix

Dissimilarity matrix

$$d(i,j) = d(j,i)$$

Nominal Attributes: Categorical without ordering (red, blue, ...)

Simple Matching:

$$d(i,j) = \frac{p - m}{p}$$

$m \rightarrow$ # of matches

$p \rightarrow$ # total variables

Binary Attributes:

Contingency Table for Binary Attributes

		Object j		
		1	0	sum
Object i	1	q	r	$q+r$
	0	s	t	$s+t$
	sum	$q+s$	$r+t$	p

→ This will be big number generally.

→ for **symmetric binary attr.** each state is equally valuable.

$$d(i,j) = \frac{r+s}{q+r+s+t}$$

Symmetric binary dissimilarity

→ **Assymetric** " " , positive and negative result of disease (not equally important)

$$d(i,j) = \frac{r+s}{q+r+s}$$

} 1-1 matching is more important the 0-0.

Example 2.18 **Dissimilarity between binary attributes.** Suppose that a patient record table (Table 2.4) contains the attributes *name*, *gender*, *fever*, *cough*, *test-1*, *test-2*, *test-3*, and *test-4*, where *name* is an object identifier, *gender* is a symmetric attribute, and the remaining attributes are asymmetric binary.

For asymmetric attribute values, let the values *Y* (*yes*) and *P* (*positive*) be set to 1, and the value *N* (*no* or *negative*) be set to 0. Suppose that the distance between objects

Table 2.4 Relational Table Where Patients Are Described by Binary Attributes

name	gender	fever	cough	test-1	test-2	test-3	test-4
Jack	M	Y	N	P	N	N	N
Jim	M	Y	Y	N	N	N	N
Mary	F	Y	N	P	N	P	N
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

sym. asym.

$d(Jack, Mary) = d_s(J, M) + d_a(J, M)$

↓

We can make normalization bcs d_s have 1 entry, d_a have 6.

(patients) is computed based only on the asymmetric attributes. According to Eq. (2.14), the distance between each pair of the three patients—Jack, Mary, and Jim—is

$$d(Jack, Jim) = \frac{1+1}{1+1+1} = 0.67,$$

$$d(Jack, Mary) = \frac{0+1}{2+0+1} = 0.33,$$

$$d(Jim, Mary) = \frac{1+2}{1+1+2} = 0.75.$$

dissimilarity with binary

These measurements suggest that Jim and Mary are unlikely to have a similar disease because they have the highest dissimilarity value among the three pairs. Of the three patients, Jack and Mary are the most likely to have a similar disease. ■

Combining binary and real number?

$\Delta w + \Delta c$

↓ ↓

$[0,1]$ $\{0,1\}$

Interval features : age [0, 70] , height [20, 150] etc.

Let assume there are 2 features in set.

[-50, 70]

[700, 1200]

They are not comparable.

We should standardize data

1-) z-score

$$z = \frac{x - m}{s}$$

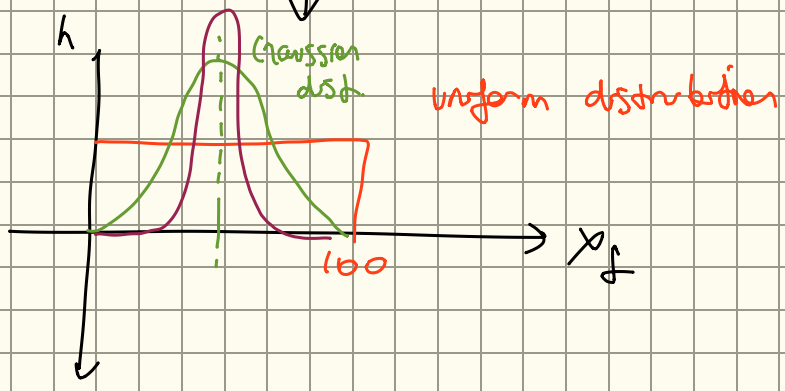
$$s = \frac{1}{n}(|x_1 - m| + \dots + |x_n - m|)$$

mean absolute deviation

or

Why not this?

$$\frac{x - m}{dist(max - min)}$$



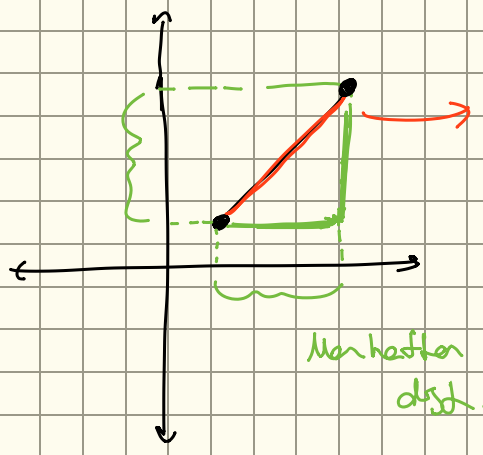
How can I normalize it between -1, 1 ?

$\frac{x_f - 50}{100}$ → mean

→ distribution

Gaussian için /100 yapmak mantıklı olmaz. Çünkü eşit dağılmamıştır.

Bu yüzden $\frac{x-m}{s}$ kullanmak $\frac{x-m}{dist(max-min)}$'e göre daha iyi olur.



→ Euclidean dist. shortest

or Minkowski distance.

What happens on binary features ? (Not intervals)

1 → is more "important" than 0 !

(1 1) is more important to contribute
0 0 similarity.

} asimetrik

Build contingency matrix

Dissimilarity for symmetric :

" " asymetric :

Jaccard distance : important is 1-1

$$sim_J(i,j) = \frac{a}{a+b+c}$$

Asymmetric binary similarity

Decide symmetric or asymmetric ?

↳ look at the data

→ Meselâ "cough" bazı durumlarda daha farklı olabilir.

→ "gender" her zaman symmetric olmayabilir → meselâ engineering olanlarında

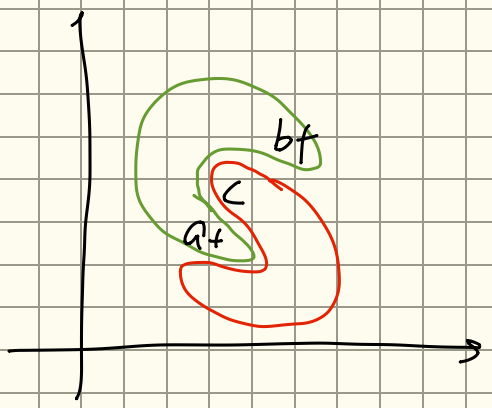
Categorical features? (Normal variables)

Simple Matching:

$$d(i,j) = \frac{p - m}{p}$$

m → # of matches

p → # total variables

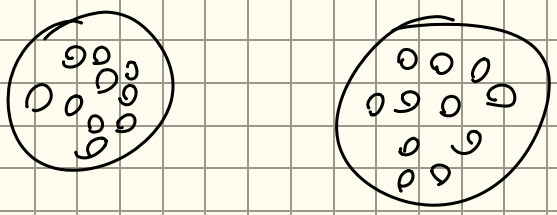


$$d(a,b) < d(a,c) < d(b,c)$$

$$d_e(a,b) \gg d(a,c), d(b,c)$$

No way to use these metrics.

Calculating distances between Clusters



Single link: Shortest dist.
↳ Complexity: n^2

Complete ": Longest " n^2

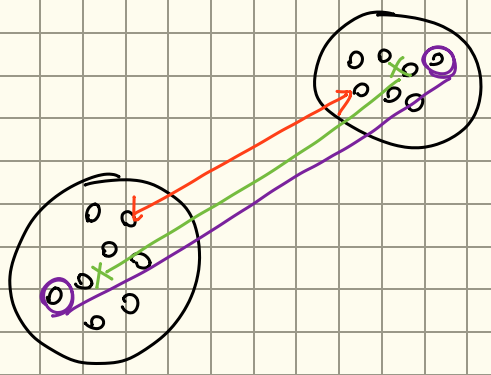
Average ": Avg dist. n^2

Centroid: $\frac{1}{2}\sum_{i=1}^{n} x_i$ $\underline{\underline{n}}$

Medoid: An actual point $\underline{\underline{n^2}}$

one chosen, centrally located object.

How can I measure the quality of within cluster?



1-) average distance to center ? } works only when $x_i \in \mathbb{R}^d$

gender, age, place, salary

2-) $\frac{1}{n \times n} \sum_{i,j} d(i,j)$ } dissimilarity

3-) maximum distance } k-means

Evaluate clusters } Goodness within + Goodness across

↓

Depends on shape of clusters

Distance / dissim. metric

Supervised:

$$\arg\max_{f \in F} \sum_{i=1}^{N} g(y_i, f(x_i))$$

↳ This goodness is difference btw. what you generate and the actual.

Unsupervised: (# of clusters)

$$\arg\max_{k,\, C_k \in F} \sum_{k=1}^{K} \sum_{i=1}^{N} g(C_k(x))$$

some mapping func.

↳ This goodness is actually calculating.

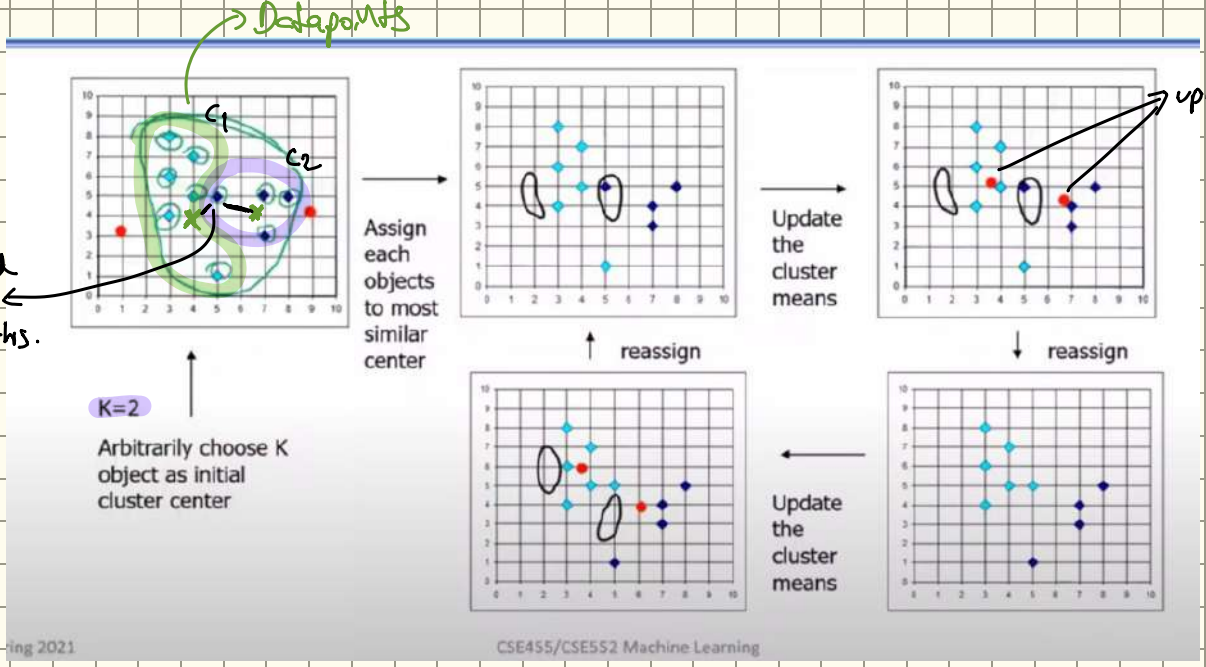
Partitioning Algorithms

Kmeans: Start with random center

Update center in each iteration

$l \times (n \times 2 + n)$ → $O(n \times l \times k)$ → clusters

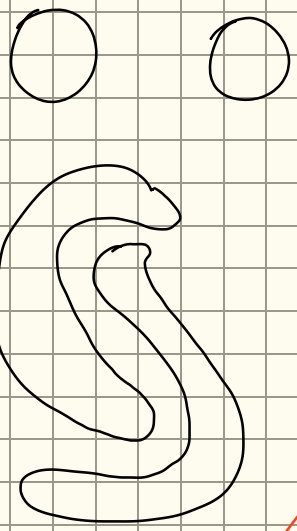
l = data, n = num of iterations



→ Greedy algo.

→ find means, label it with its closer mean.

→ Algo starts with random 2 points as centers.



fine with kmeans, convex shape

Weakness

- Applicable only when mean is defined
- Sensitive to outliers
- Determining k
- Only with numeric variables (can calculate distance)
  - one hot encoding

k-medoids: Instead of taking the mean, we can use medoids,

↓

most centrally located object

kmeans → calculate mean $O(n)$, fast on certain problems

kmedoids → " medoid $O(n^3)$?

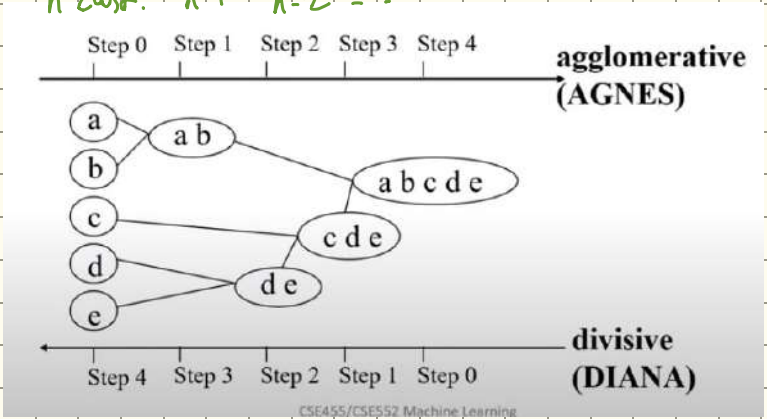
|| appr

PAM: $O(k(n-k)^2)$ aims to minimise the sum of dissimilarities btw points and their medoids.

CLARA: $O(ks^2 + k(n-k))$ large datasets.

Hierarchical Clustering!

n clust. n-1 n-2 - - -



- In the beginning, Eaach data point 1 cluster.

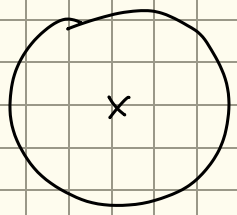
- Step 0 → n
- Step 1 → n^2 combinations

* Doesnt require any number k, but needs termination condition.

- Dont scale well. → $O(n^2)$ at least
- When you make decision, you cant undo (like decision trees)

* Combine hierarchical with distance-based clustering } - BIRCH (tree like search)

- CHAMELEON



outlier

- kmeans sensitive to outliers

↓

- kmedoid ✓ → $O(n^3)$ to find medoid.

Density-Based Methods: $O(n^2)$

DBSCAN

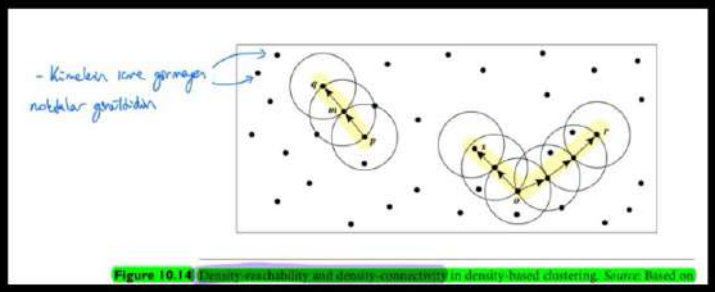


Figure 10.14 Density reachability and density connectivity in density-based clustering. Source: Based on

Algorithm:

1-) Select point p → visited

2-) density-reachable

3-) If p is core point, a cluster is formed

4-) If not, no points density reachable, find new point

5-) Continue until visit all points

Parameters:

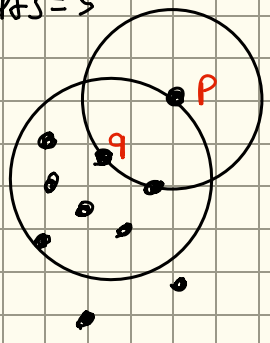
Eps: Map radius of neighborhood

MinPts: Min num of points in an Eps-neighborhood of that point

(According to this I can say this border is dense enough.

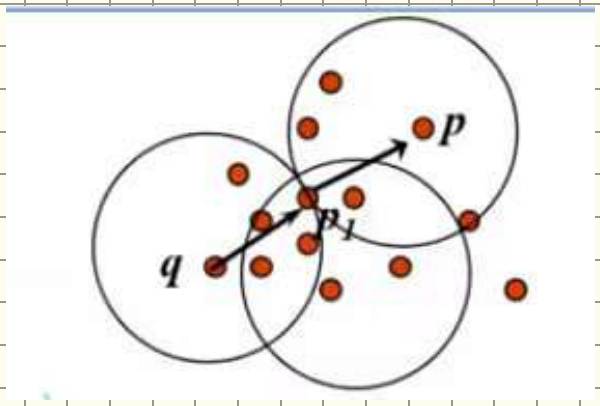
Eps = 1cm

MinPts = 5

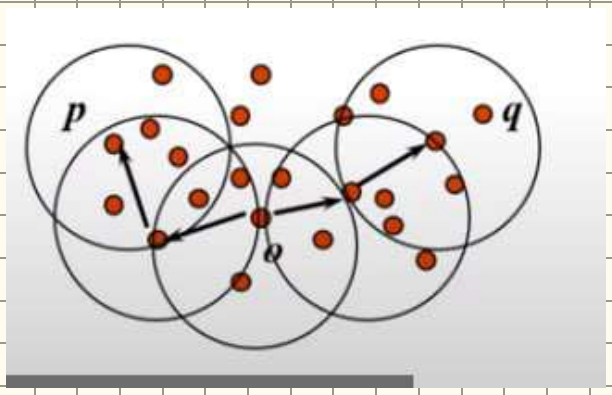


Density reachable if p belongs to $N_{eps}(q)$

then, Core point if $|N_{eps}(q)| \geq MinPts$



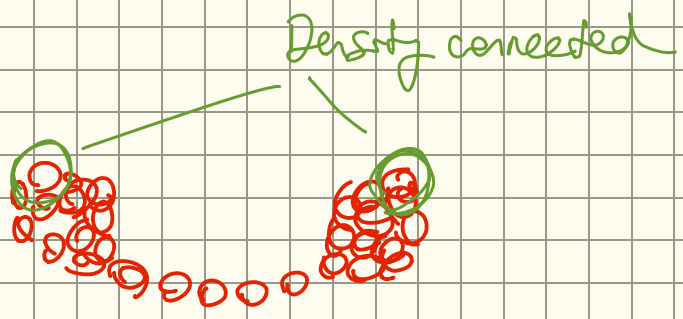
$q - p_1 - p$ } density reachable
chain of points



Density connected

Both p and q are density reachable from point o.

So p and q are density connected.



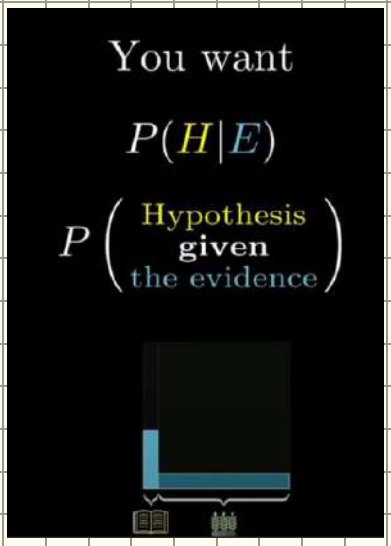
BAYES

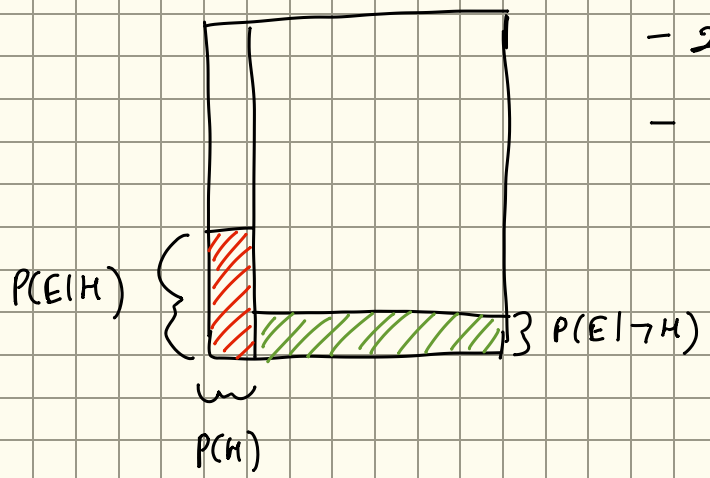
- Verilerdeğerken kararı koşula bağladığımız zaman, ölçülmesi oldukça kontroversial olur.

$$P(\text{crafter} \mid \text{Steven}) = \frac{20}{24}$$

$$P(\text{librarian} \mid \text{Steven}) = \frac{4}{24}$$

$$P(H|E) = \frac{P(H) \cdot P(E|H)}{P(E)}$$





- 200 farmer
- 10 librarian

Prior

$$P(H) = \frac{\text{librarians}}{\text{All people}} = \frac{1}{21}$$

$$P(E|H) = \frac{4}{10} \quad \left.\right\} \text{The prob. of Steven in given librarians}$$

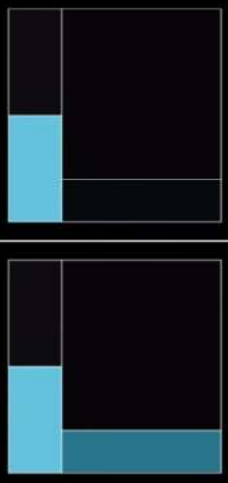
$$P(E|\neg H) = \frac{20}{200} \quad \left.\right\} \text{the prob. of Steven in given farmers.}$$

$$P(H|E) = \frac{\frac{4}{10}}{\frac{4}{10} + \frac{20}{200}}$$

The prob. of librarians in given Steven.

Bayes' theorem

$$P(H|E) = \frac{P(H)P(E|H)}{P(E)} = \frac{}{}$$



Binary Classification

$P(+|x)$ +

$P(-|x)$ $= 1$

arg max $P(c|x)$
c

Problem:

T	P	H	Conf / Not C.
high			
low			
medium			

$P(C \mid T, P, H)$?

Now, lets just look at T.

(Using Logformal ion)

$$P(C \mid T) = \frac{P(C) \cdot P(T \mid C)}{P(T)}$$

$$P(C \mid T = high) = \frac{P(C) \cdot P(T_h \mid C)}{P(T = high)}$$

T	C
h	+
h	+
h	−
l	+
l	+

$$= \frac{4/5 \cdot 2/4}{3/5}$$

$$P(C \mid T = med) = \frac{4/5 \cdot 0}{0} =$$

$$P(C \mid T = low) = \frac{4/5 \cdot 2/4}{2/5}$$

x_1 = this digit has circle

x_2 = " " a single 1

x_3 = " " 2 single 1's

x_4 = " " 3 " " "

$D_8 = 100$

$D_9 = 110$

⋮

For all 0's what is the prob. of the see this pattern?

$$P(d_0 \mid x_1 x_2 x_3 x_4) = \frac{P(x_1 x_2 x_3 x_4 \mid d_0)\, P(d_0)}{P(x_1 x_2 x_3 x_4)}$$

($P(d_0)$: $\frac{1}{10}$)

$P(h_1)$ $P(h_2)$ → $P(h_1 \mid \ldots)$ ↑

0.99 0.1

Naive Bayes

$P(x_1 = +, x_2 = + \mid \text{cancer})$

$P(x_1 = -, x_2 = + \mid \text{canc})$

⋮

So, $P(x_1, x_2, x_3 \mid V)$

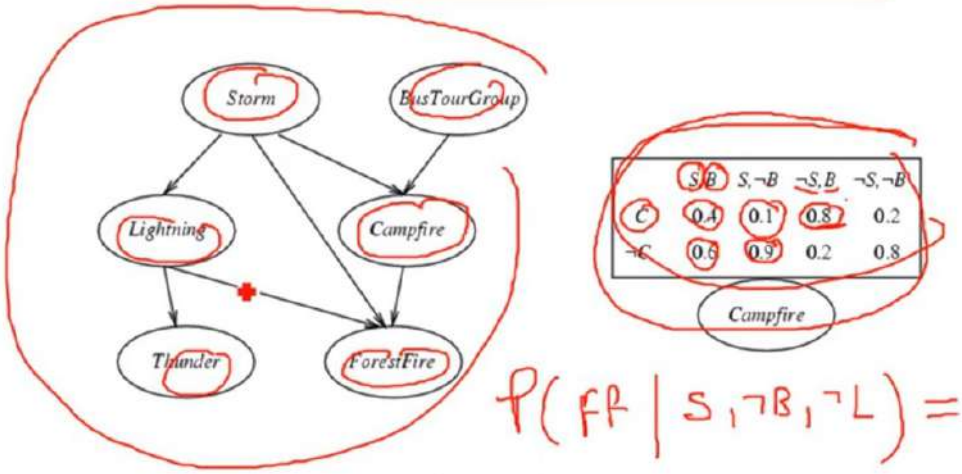
$= P(x_1 \mid V) \cdot P(x_2 \mid V) \cdot P(x_3 \mid V)$

conditionally independence.

$$P(h \mid x_1 \ldots x_d) = \underset{h \in H}{\arg\max}\; P(h) \cdot \prod_{i=1}^{d} P(x_i \mid h)$$

(∏: compute)

Bayesian Belief Network ; No loops



$P(FF \mid S, \neg B, \neg L) =$

In continuous data prob → 1.) Discretize it

↳ if age make groups

↳ if salary " "

$P(x_1 \mid ?, +/-)$

↓

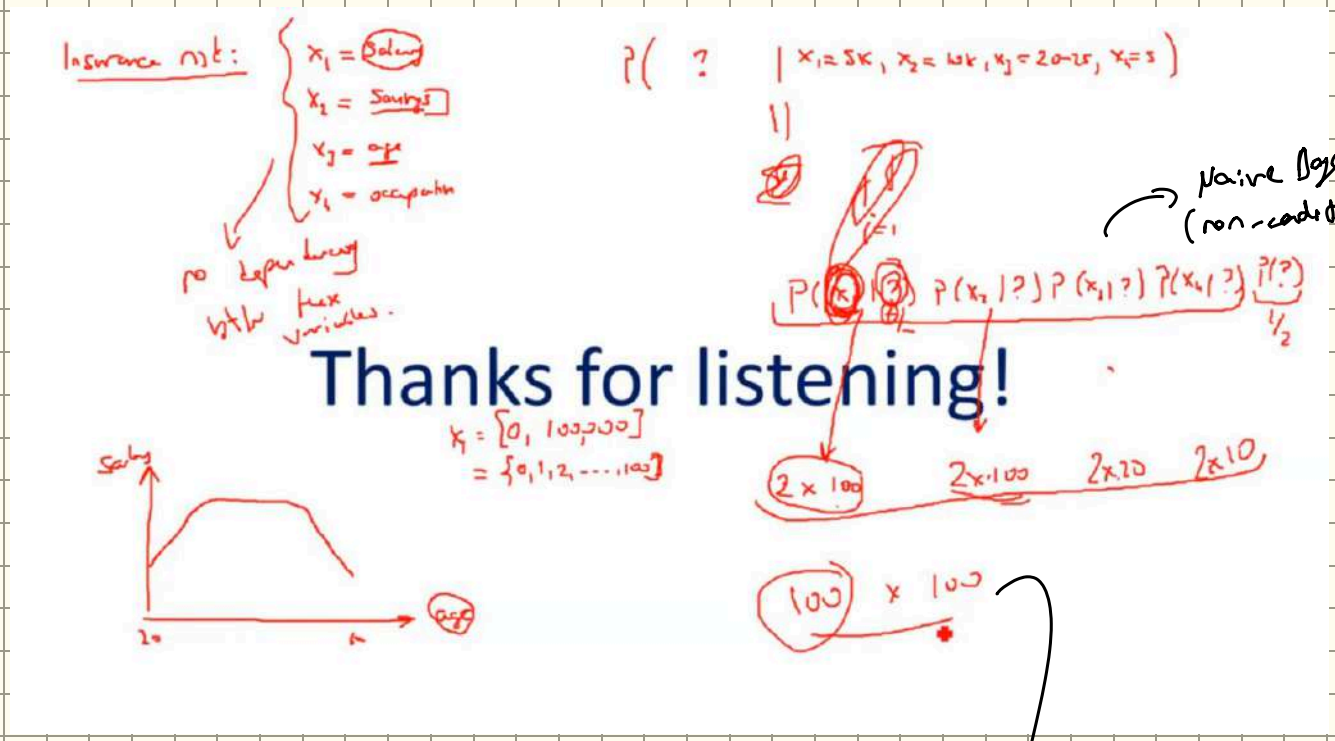
2×100 num of prob.

↓ hypothesis +/−

↓ group num for salary

discretize ($x_1 = [0, 100000]$

↳ $= \{0, 1, \cdots, 100\}$


Insurance ppt:
X_1 = Salary
X_2 = Savings
X_3 = age
X_4 = occupation
no dependency with these variables.
P(? | X_1=5K, X_2=10K, X_3=20-25, X_4=3)
Naive Bayes (non-conditional)
P(?) 1/2
Thanks for listening!
X_1 = [0, 100,000]
= {0, 1, 2, ... 100}
Salary
age
2 x 100
2x100
2x20
2x10
100 x 100
If conditional, they will grow very quickly.

Final Cikmaz

Q2) Dissim

Asym	Sym	Jaccard
3/3	1/3	0

Q4) <No, Married, Middle>

P(No | <No, M-, M>) vs P(Yes | <---->)

↓

$\frac{P(No) \cdot P(<..>|No)}{P(...)}$

$\frac{7}{10} \cdot \frac{4}{7} \cdot \frac{4}{7} \cdot \frac{3}{7}$

$\frac{P(Yes) \cdot P(<...>|Yes)}{P(<---->)}$

$\frac{3}{10} \cdot \frac{1}{3} \cdot 0$

0

6) Gaussion distribution

Low = 1 Middle = 2 High = 3

~~Mean~~ 1-) Calculate mean and variance for Yes and No of Income.

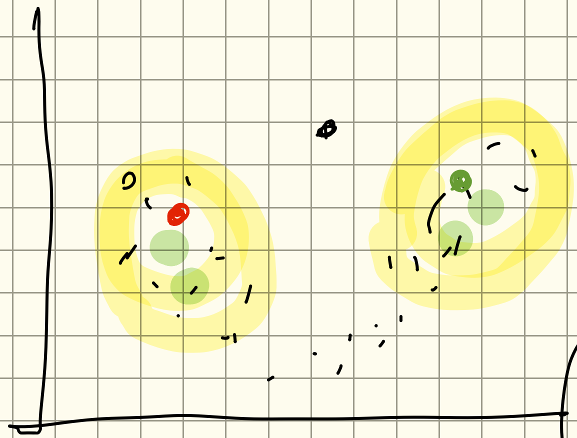
2-) Calculate $P(\text{Middle} = 2 \mid \text{Yes})$ with prob. density function.

$P(\text{M--} = 2 \mid \text{No})$

3-) $\underbrace{P(\text{No} \mid \langle \text{No, Mid..} \rangle)}$ = Diğer şeyler aynı, income yerine farklı yazarsak.

$$\frac{P(\text{No}) \cdot P(\langle .. \rangle \mid \text{No})}{P(\langle ... \rangle)}$$

Q3) b)

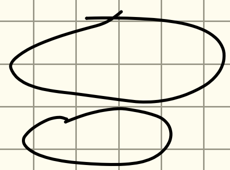


1-) Decide k and choose random 2 medoid.

2- Find clusters

3-) Look data points, if quality better swap medoid

k means

Başlıkta olsaydı mean,  böyle olabilirdi.