Week 6
Nonlinear Algorithms

Content

- Nonlinear Algorithms
- Principal Component Analysis（PCA）
- Naive Bayes
- K-Nearest Neighbors
- Classification And Regression Trees (CART)
- Example: Algorithms in python

Nonlinear Algorithms

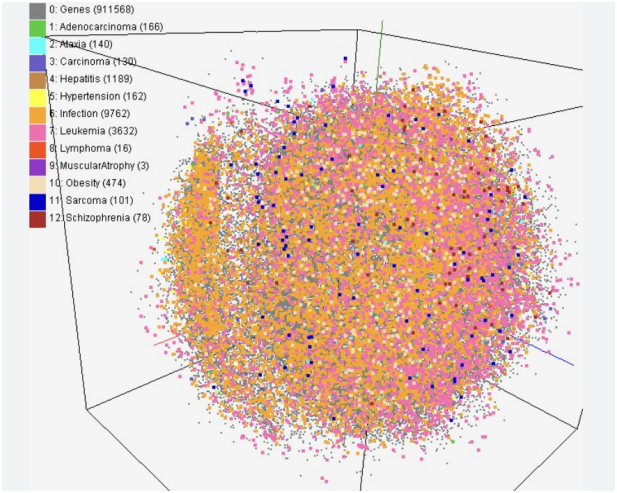
What are Nonlinear algorithms in machine learning?

- **Nonlinear algorithms** in machine learning refer to a broader category of regression models where the relationship between the dependent variable and the independent variables is not assumed to be linear
- Unlike linear regression, which assumes a linear relationship, nonlinear regression models capture more complex patterns.
- **Types of Nonlinear Regression**
    - **Parametric Nonlinear Regression:** Assumes that the relationship can be modeled using a specific mathematical function (e.g., polynomial regression, logistic regression, exponential regression).
    - **Non-parametric Nonlinear Regression:** Does not assume a specific mathematical function and allows for more flexible modeling.
- **Assumptions in Nonlinear Regression**
    - **Functional Form:** The chosen nonlinear model correctly represents the true relationship between the variables.
    - **Independence:** Observations are independent.
    - **Homoscedasticity:** Residual variance is constant across levels of the independent variable.
    - **Normality:** Residuals are normally distributed.
    - **Multicollinearity:** Independent variables are not perfectly correlated.

Principal Component Analysis (PCA)

Big & High-Dimensional Data

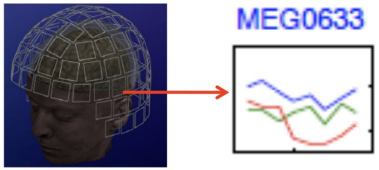
- **High-Dimensions means a Lot of Features**
- In the realm of machine learning and data science, high-dimensional data refers to datasets with a large number of features or attributes.
- These datasets can be challenging to work with due to their complexity and the sheer volume of information that each data point contains.
- High-dimensional data is common in many advanced applications, such as genomics, image processing, and natural language processing.

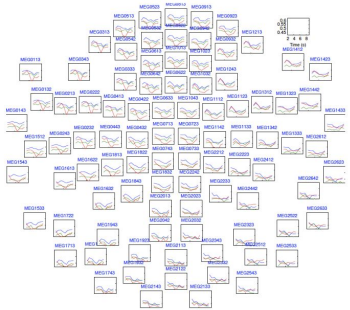


MEG Brain Imaging

120 locations x 500 time points

x 20 objects





Principal Component Analysis（PCA）

- ## Challenges of High-Dimensional Data
  - **Overfitting:** With a large number of features, machine learning models can become overly complex and may capture noise rather than the underlying pattern, leading to poor generalization on unseen data.
  - **Computational complexity:** High-dimensional datasets require more computational resources for processing and analysis, which can be costly and time-consuming.
  - **Visualization:** Visualizing data with more than three dimensions is not straightforward, making it difficult to gain intuitive insights from the data.
  - **Distance metrics:** In high-dimensional spaces, traditional distance metrics like Euclidean distance can become less meaningful, as the distance between data points tends to converge.
- ## Dealing with High-Dimensional Data

  When working with high-dimensional data, it is essential to apply appropriate preprocessing and feature selection techniques. Feature selection involves identifying the most relevant features for the task at hand, which can improve model performance and reduce overfitting.
  - **Filter methods:** These methods use statistical tests to select features that have the strongest relationship with the output variable.
  - **Wrapper methods:** Wrapper methods use a predictive model to score feature subsets and select the combination that results in the best model performance.
  - **Embedded methods:**
    Embedded methods perform feature selection as part of the model training process, such as LASSO and Ridge regression, which include regularization terms to penalize the inclusion of irrelevant features.

Principal Component Analysis（PCA）

Dimensionality Reduction Techniques

- **PCA**
  - PCA is a linear method that aims to reduce the dimensionality of data while preserving as much variance as possible.
  - It identifies the directions (principal components) in the data with the highest variance.
  - Useful for visualization, noise reduction, and feature selection.
- **Kernel PCA**
  - An extension of PCA that uses kernel functions to handle nonlinear data.
  - It maps data into a higher-dimensional space and then applies PCA. Useful for nonlinear dimensionality reduction.
- **ICA**
  - ICA aims to find statistically independent components in the data.
  - Unlike PCA, which finds orthogonal components, ICA seeks non-Gaussian, independent components.
  - Commonly used in blind source separation (e.g., separating mixed audio signals).
- **Autoencoders**
  - **Neural network**-based models for **unsupervised feature learning.**
  - Consist of an encoder (maps input to a lower-dimensional representation) and a decoder (reconstructs input from the representation). Nonlinear and can learn complex features.
- **t-Distributed Stochastic Neighbor Embedding (t-SNE)**
  - Used for **visualization by mapping high-dimensional data to a lower-dimensional space** (usually 2D or 3D).
  - Preserves pairwise distances between data points. Great for visualizing clusters and identifying patterns.

Principal Component Analysis（PCA）

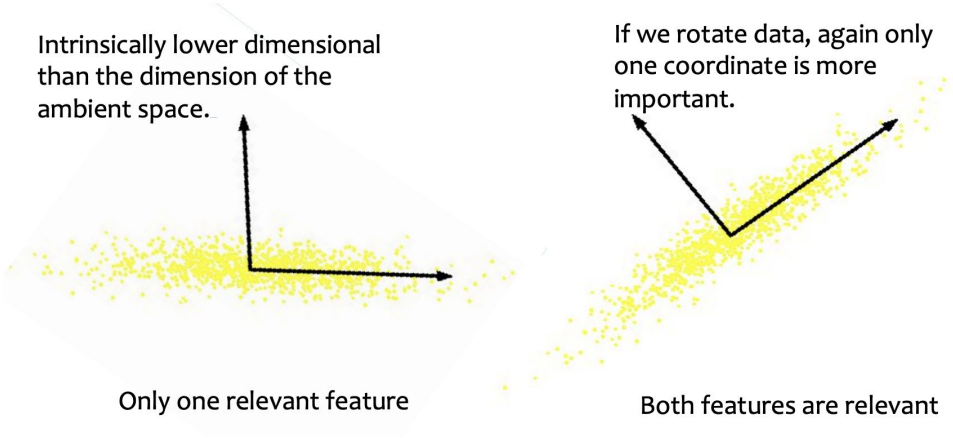
Why do we use dimensionality reduction techniques?

- PCA, Kernel PCA, ICA: Powerful unsupervised learning techniques for extracting hidden (potentially lower dimensional) structure from high dimensional datasets.
- Useful for:
  - Visualization
  - More efficient use of resources (e.g., time, memory, communication)
  - Statistical: fewer dimensions -> better generalization
  - Noise removal (improving data quality)
  - Further processing by machine learning algorithms

Principal Component Analysis (PCA)

What is PCA?

- Unsupervised technique for extracting variance structure from high dimensional datasets.
- PCA is an orthogonal projection or transformation of the data into a (possibly lower dimensional) subspace so that the variance of the projected data is maximized.

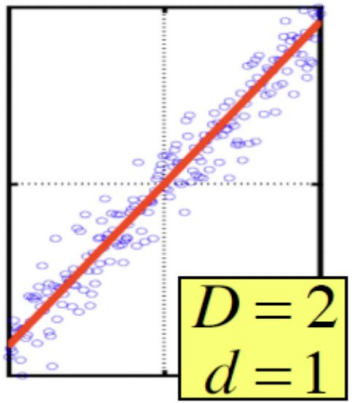


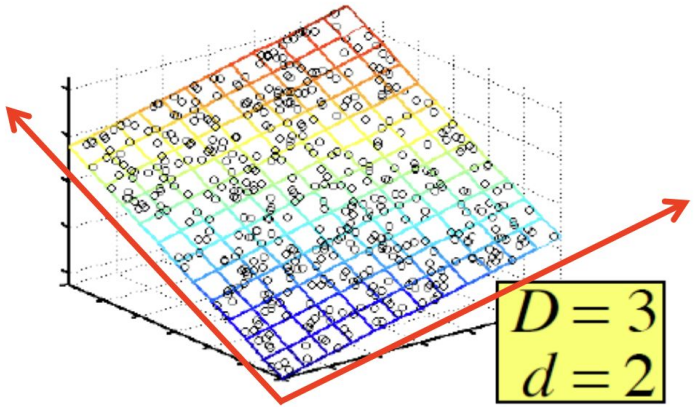
Question: Can we transform the features so that we only need to preserve one latent feature?

Principal Component Analysis (PCA)

What is PCA?

- In case where data lies on or near a low d-dimensional linear subspace, axes of this subspace are an effective representation of the data.
- Identifying the axes is known as Principal Components Analysis, and can be obtained by using classic matrix computation tools (Eigen or Singular Value Decomposition).

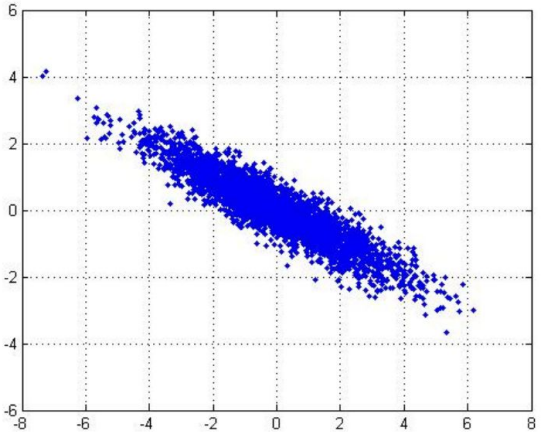




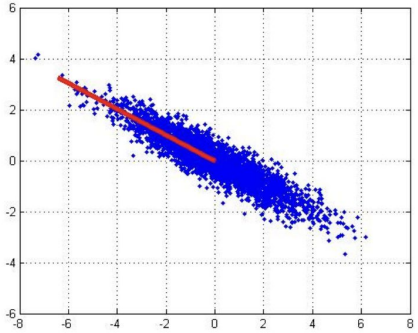
Principal Component Analysis (PCA)

- **Example**

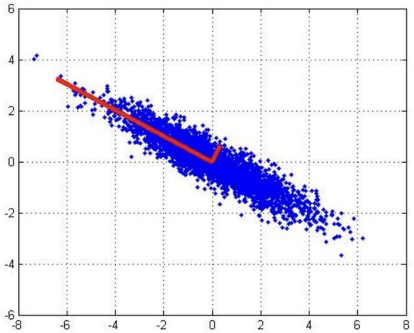
2D Gaussian dataset



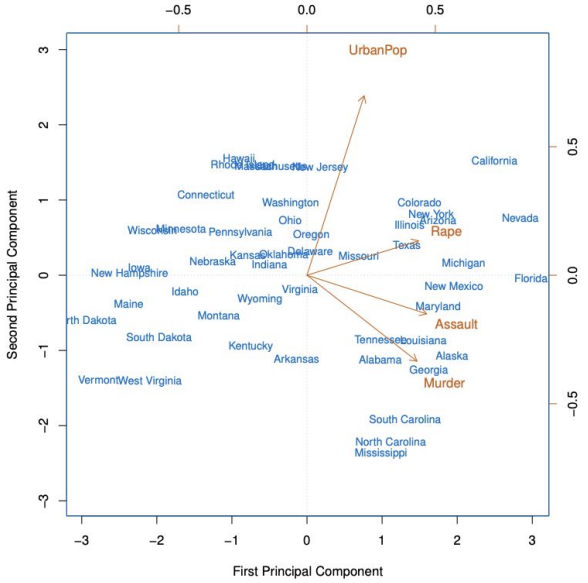
1st PCA axis



2nd PCA axis



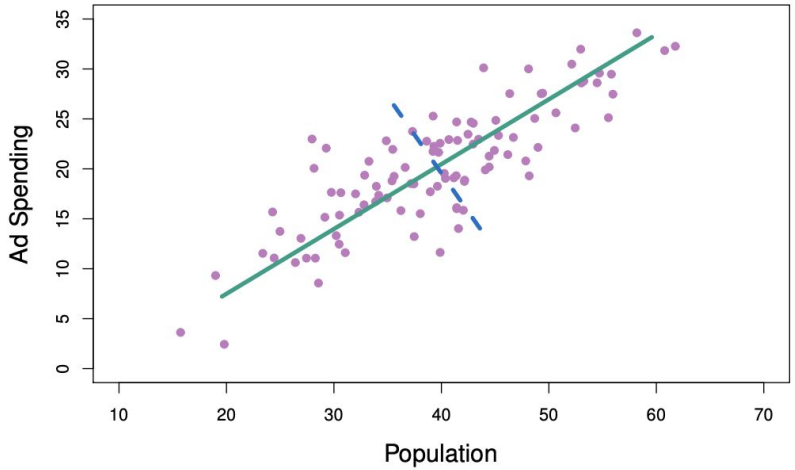
Biplot (courtesy of PCA)



Principal Component Analysis (PCA)

What is the first principal component (PC)?

- First PC is the direction of the line that is closest to our data points on average

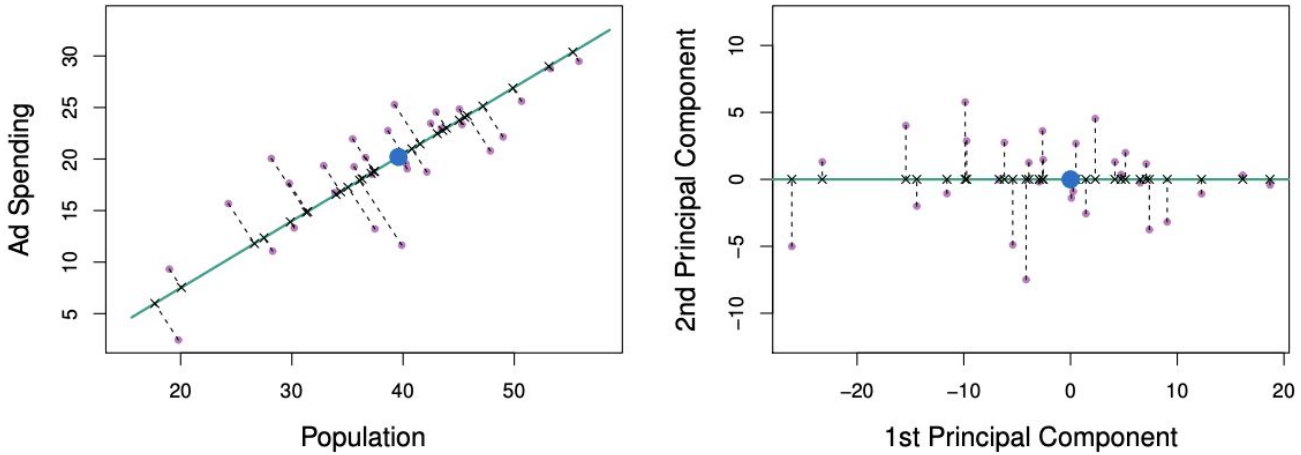


"Closeness" measured by squared (Euclidean) distance

Principal Component Analysis (PCA)

What is the first principal component (PC)?

- That is, the PC direction (in teal) minimizes the average squared length of the dotted lines.

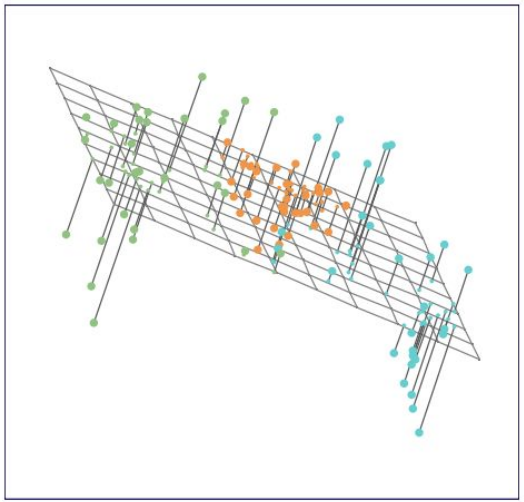


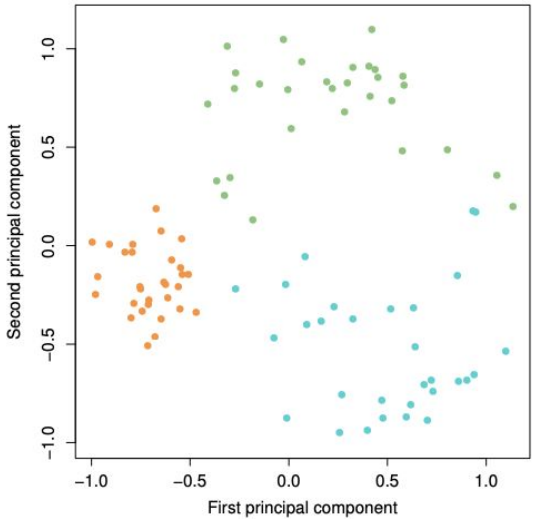
Intuition: Closest line offers the “best” 1-dimensional compression of our data

Principal Component Analysis（PCA）

What does this look like with 3 variables?

- The first two principal components span a plane which is closest to the data on average.

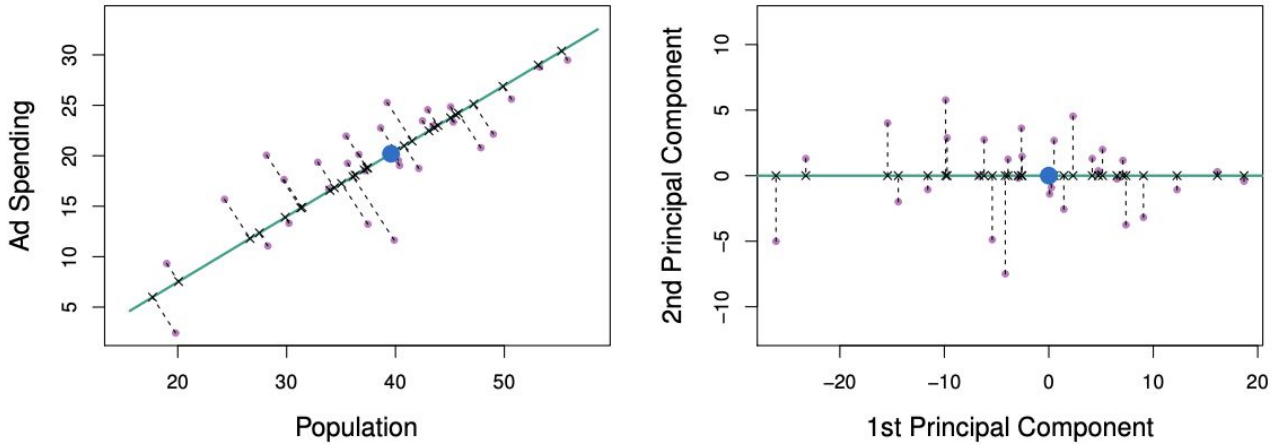




Principal Component Analysis (PCA)

A second interpretation

- First PC is the direction in feature space along which our data varies the most



Intuition: High variance directions are often interesting directions

Principal Component Analysis (PCA)

PCA algorithm I (sequential)

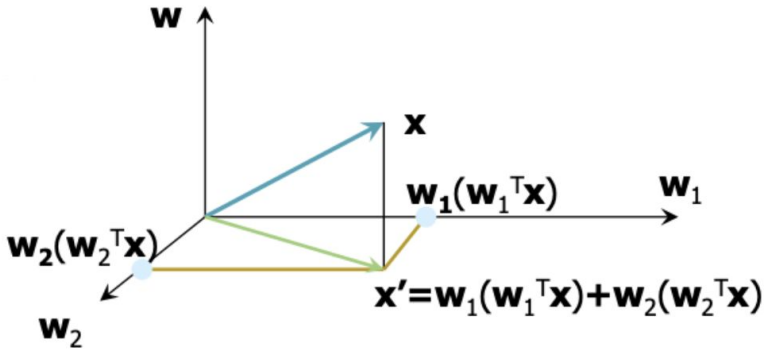
Given the **centered** data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, compute the principal vectors:

$$\mathbf{w}_1 = \arg\max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{(\mathbf{w}^T \mathbf{x}_i)^2\}$$ 1st PCA vector

We maximize the variance of projection of $\mathbf{x}$

$$\mathbf{w}_k = \arg\max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{[\mathbf{w}^T (\mathbf{x}_i - \underbrace{\sum_{j=1}^{k-1} \mathbf{w}_j \mathbf{w}_j^T \mathbf{x}_i}_{\mathbf{x}' \text{ PCA reconstruction}})]^2\}$$ k^{th} PCA vector

We maximize the variance of the projection in the residual subspace

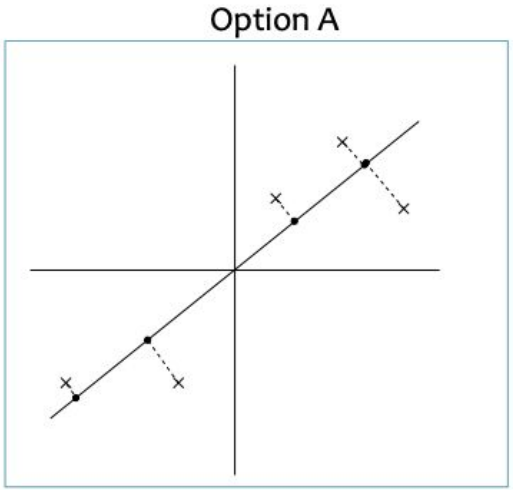


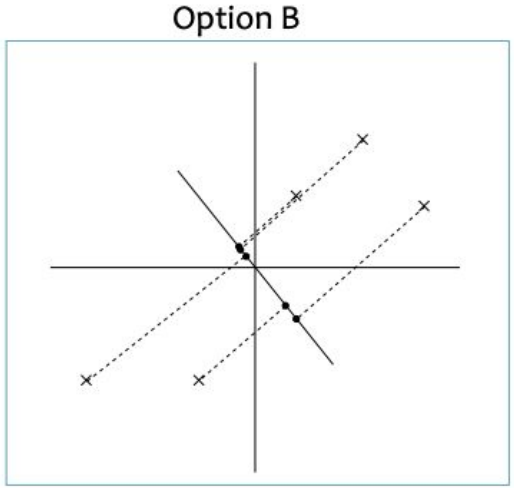
Slide from Barnabas Poczos

Principal Component Analysis（PCA）

Maximizing the Variance

Consider the two projections below, which maximizes the variance?





Principal Component Analysis (PCA)

PCA algorithm II (sample covariance matrix)

- PCA basis vectors = the eigenvectors of Σ
  - We get the eigvectors using an eigendecomposition. Power iteration (Von Mises iteration is a standard algorithm for this)
- Larger eigenvalue ⇒ more important eigenvectors

Given data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, compute covariance matrix Σ

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T \quad \text{where} \quad \bar{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i$$

Slide from Barnabas Poczos

Principal Component Analysis (PCA)

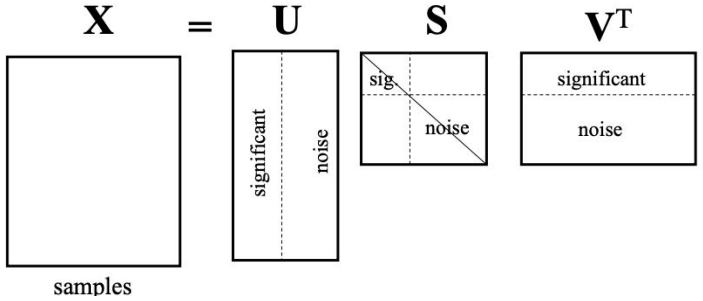
PCA algorithm III (SVD of the data matrix)

Singular Value Decomposition of the **centered** data matrix **X**.

$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m] \in \mathbb{R}^{N \times m}$,

m: number of instances,
N: dimension

$\mathbf{X}_{\text{features} \times \text{samples}} = \mathbf{USV}^T$



- **Columns of U**
  - the principal vectors, { $\mathbf{u}^{(1)}$, …, $\mathbf{u}^{(k)}$ }
  - orthogonal and has unit norm – so $U^TU = I$
  - Can reconstruct the data using linear combinations of { $\mathbf{u}^{(1)}$, …, $\mathbf{u}^{(k)}$ }
- **Matrix S**
  - Diagonal
  - Shows importance of each eigenvector
- **Columns of V^T**
  - The coefficients for reconstructing the samples

Slide from Barnabas Poczos

Principal Component Analysis (PCA)

Finding PCs with optimization

- Let X be a data matrix with n samples and p variables.
    - We assume that each column is centered, i.e., that the mean of the column has been subtracted away from each entry

To find the first principal component $\phi_1 = (\phi_{11}, \ldots, \phi_{p1})$, we solve the following optimization problem

$$\max_{\phi_{11},\ldots,\phi_{p1}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2 \right\}$$

$$\text{subject to} \sum_{j=1}^{p} \phi_{j1}^2 = 1.$$

Scalar projection of x_i onto ϕ_1. Also known as **the score** z_{i1}.

Sample mean of the scores is 0:

$\frac{1}{n}\sum_{i=1}^{n} z_{i1} = \sum_{j=1}^{p} \phi_{j1} \frac{1}{n}\sum_{i=1}^{n} x_{ij} = 0$ Sample variance of the scores z_{i1}

(i.e., sample variance of projections of datapoints onto ϕ_1).

Mackey, L. (n.d.). Statistics 202: Lecture 3 Condensed Notes [PDF]

Principal Component Analysis (PCA)

Finding PCs with optimization

- Let X be a data matrix with n samples and p variables.
    - We assume that each column is centered, i.e., that the mean of the column has been subtracted away from each entry

To find the second principal component $\phi_2 = (\phi_{12}, \ldots, \phi_{p2})$, we solve the following optimization problem

$$\max_{\phi_{12},\ldots,\phi_{p2}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j2} x_{ij} \right)^2 \right\}$$

$$\text{subject to } \sum_{j=1}^{p} \phi_{j2}^2 = 1 \quad \text{and} \quad \sum_{j=1}^{p} \phi_{j1}\phi_{j2} = 0.$$

First and second principal components must be orthogonal.

Equivalent to saying that the scores $(z_{11}, \ldots, z_{n1})$ and $(z_{12}, \ldots, z_{n2})$ are uncorrelated.

Mackey, L. (n.d.). Statistics 202: Lecture 3 Condensed Notes [PDF]

Principal Component Analysis (PCA)

Solving the optimization problems

These optimization problems are standard in linear algebra. They are solved by:

- The singular value decomposition (SVD) of $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{\Phi}^T$$

  where the ith column of $\mathbf{\Phi}$ is the ith principal component ϕ_i, and the ith column of $\mathbf{U}\mathbf{\Sigma}$ is the ith vector of scores $(z_{1i}, \ldots, z_{ni})$.

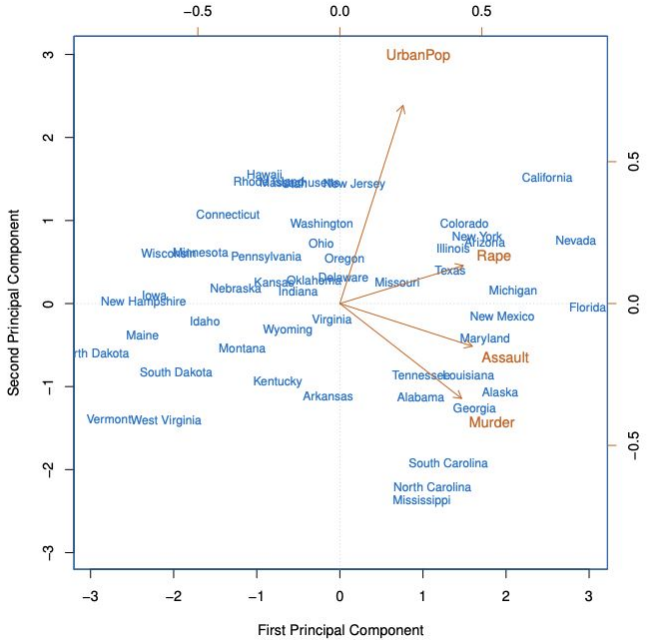
- The eigendecomposition of $\mathbf{X}^T\mathbf{X}$:

$$\mathbf{X}^T\mathbf{X} = \mathbf{\Phi}\mathbf{\Sigma}^2\mathbf{\Phi}^T$$

Mackey, L. (n.d.). Statistics 202: Lecture 3 Condensed Notes [PDF]

Principal Component Analysis (PCA)

PCA in practice: The biplot



- j-th variable represented by its PC loadings (ϕ_{1j}, ϕ_{2j})
- i-th datapoint represented by its scores (z_{i1}, z_{i2})

Mackey, L. (n.d.). Statistics 202: Lecture 3 Condensed Notes [PDF]

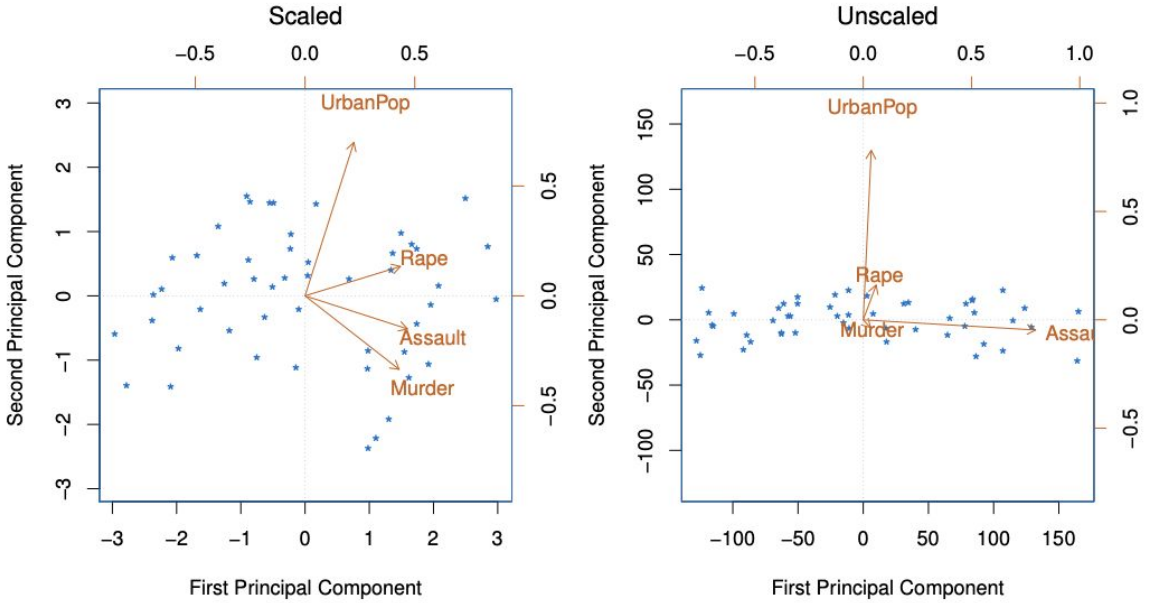
Principal Component Analysis（PCA）

Scaling the variables

- We have already discussed centering our variables to have mean 0. In some instances we will also want to rescale our variables prior to performing PCA.
    - e.g., if different variables are measured in different units.
    - Or if we're principally interested in the correlations between variables with very different scales.
- When a variable is centered and rescaled to have standard deviation 1, we say it has been **standardized**.
- In other instances, initial variable scalings are meaningful and should be preserved I
    - e.g., Gene expression levels for different genes all measured in the same units.

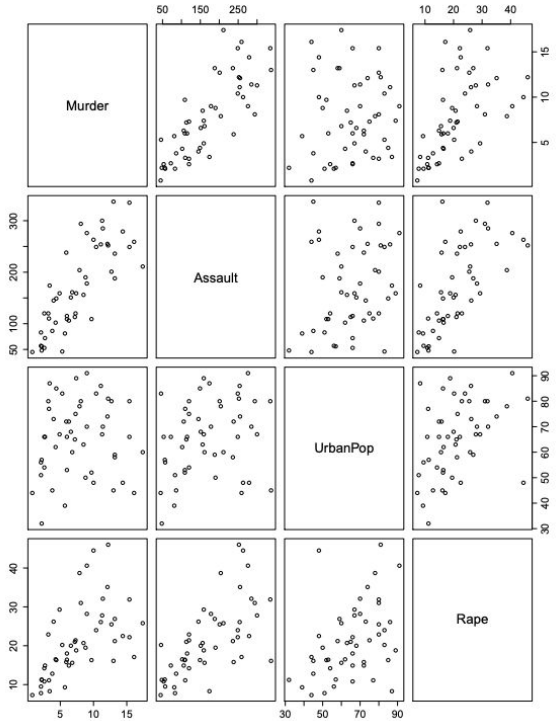
Principal Component Analysis（PCA）

Example: scaled vs. unscaled PCA



Principal Component Analysis（PCA）

How many principal components are enough?



Principal Component Analysis（PCA）

The proportion of variance explained

- We can think of the top principal components as directions in space in which the data vary the most.
- The ith **score vector** $(z_{1i}, \ldots, z_{ni})$ can be interpreted as a new variable. The variance of this variable decreases as we take i from 1 to p. However, the total variance of the score vectors is the same as the total variance of the original variables:

$$\sum_{i=1}^{p} \frac{1}{n} \sum_{j=1}^{n} z_{ji}^2 = \sum_{k=1}^{p} \mathsf{Var}(x_k)$$

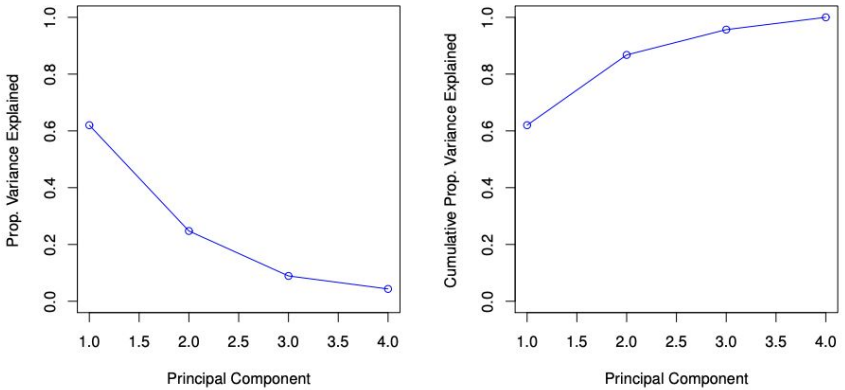
- We can quantify how much of the variance is captured by the first m principal components/score variables.

Principal Component Analysis（PCA）

The proportion of variance explained

- The variance of the *mth* score variable is:

$$\frac{1}{n}\sum_{i=1}^{n} z_{im}^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{jm}x_{ij}\right)^2 = \frac{1}{n}\mathbf{\Sigma}_{mm}^2.$$



Principal Component Analysis（PCA）

Generalizations of PCA

- PCA works under a Euclidean geometry in the space of variables. Often, the natural geometry is different:
  - We expect some variables to be “closer” to each other that to other variables.
  - Some correlations between variables would be more surprising than others.
- Examples:
  - Variables are pixel values, samples are different images of the brain. We expect neighboring pixels to have stronger correlations.
  - Variables are rainfall measurements at different regions. We expect neighboring regions to have higher correlations.

Naive Bayes

Recap of approaches we've seen so far

- Nearest neighbor is widely used
  - Super-powers: can instantly learn new classes and predict from one or many examples
- Logistic Regression is widely used
  - Super-powers: Effective prediction from high-dimensional features
- Linear Regression is widely used
  - Super-powers: Can extrapolate, explain relationships, and predict continuous values from many variables
- Almost all algorithms involve nearest neighbor, logistic regression, or linear regression
  - The main learning challenge is typically **feature learning**

Naive Bayes

Recap: Probability

- **Probabilistic model**

$$y^* = \underset{y}{\operatorname{argmax}}\, P(y|x)$$

- **Joint and conditional probability**

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

$$P(a, b, c) = P(a|b, c)P(b|c)P(c)$$

- **Bayes Rule**

$$P(x|y) = \frac{P(x, y)}{P(y)} = \frac{P(y|x)P(x)}{P(y)}$$

Naive Bayes

Recap: Probability

- **Law of total probability**

$$\left[\sum_{v \in x} P(x = v)\right] = 1$$

For continuous variables, replace sum over possible values with integral over domain

- **Marginalization**

$$\left[\sum_{v \in x} P(x = v, y)\right] = P(y)$$

- **Estimate probabilities of discrete variables by counting**

$$P(x = v) = \frac{1}{|N|} \sum_{n} \delta(x_n = v)$$

Naive Bayes

Example

x: Larger than 10 lbs?

y	F	T
Cat	15	25
Dog	5	40

$$P(y = Cat) =$$

$$P(y = Cat|x = F) =$$

$$P(x = F|y = Cat) =$$

Naive Bayes

A is independent of B if (and only if)

$$P(A, B) = P(A)P(B)$$

$$P(A|B) = P(A), \qquad P(B|A) = P(B)$$

- What if you have 100 variables? How can you count all combinations?
- Fully modeling dependencies between many variables (more than 3 or 4) is challenging and requires a lot of data

Naive Bayes

Probabilistic model

$$y^* = \underset{y}{\operatorname{argmax}}\, P(y|x)$$

Or equivalently…

$$y^* = \underset{y}{\operatorname{argmax}}\, P(x|y)P(y)$$

$$\underset{y}{\operatorname{argmax}}\, P(y|x) = \underset{y}{\operatorname{argmax}}\, P(y|x)P(x) = \underset{y}{\operatorname{argmax}}\, P(y,x) = \underset{y}{\operatorname{argmax}} P(x|y)P(y)$$

Naive Bayes

Notation

- x_i is the ith feature variable
  - i indicates the feature index
- x_n is the nth feature vector
  - n indicates the sample index
  - y_n is the nth label
- x_{ni} is the ith feature of the nth sample
- $\delta(x_{ni} = v)$ returns 1 if $x_{ni} = v$; 0 otherwise
  - v indicates a feature value
  - δ is an indicator function, mapping from true/false to 1/0

Naive Bayes

Naïve Bayes Model

Assume features $x_1..x_m$ are independent given the label y:

$$P(\boldsymbol{x}|y) = \prod_i P(x_i|y)$$

Then

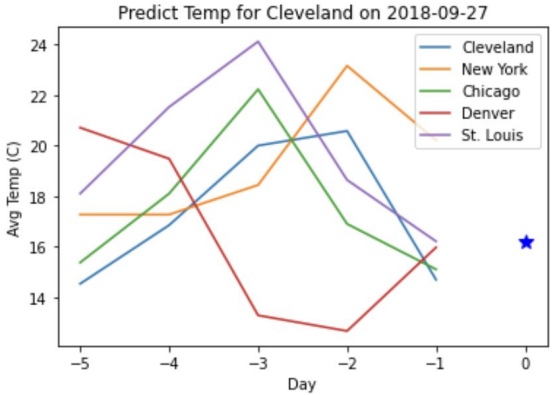
$$y^* = \underset{y}{\text{argmax}} \prod_i P(x_i|y)P(y)$$

Naive Bayes

Examples

- Digit classification: choose the label that maximizes the product of likelihoods of each pixel intensity

- Temperature prediction: each feature predicts y with some offset and variance ($y - xi$ is univariate Gaussian)



Naive Bayes

Naïve Bayes Algorithm

- **Training**
  - Estimate parameters for $P(xi\ |y)$ for each i
  - Estimate parameters for $P(y)$

- **Prediction**
  - Solve for y that maximizes $P(x, y)$

$$y^* = \underset{y}{\text{argmax}} \prod_i P(x_i|y)P(y)$$

Naive Bayes

Q: How to estimate $P(xi \mid y)$ from data?

- **Basic principles of fitting likelihood parameters from data**
  - MLE (maximum likelihood estimation): Choose the parameter that maximizes the likelihood of the data
  - MAP (maximum a priori): Choose the parameter that maximizes the data likelihood and its own prior
    - As Warren Buffet says, it's not just about maximizing expected return – it's about making sure there are no zeros.

Bernoulli (x is binary; y is discrete)

$$P(x_i|y = k) = \theta_{ki}^{x_i}(1 - \theta_{ki})^{1-x_i}$$

Categorical (x is has multiple discrete values, y is discrete)

Naive Bayes

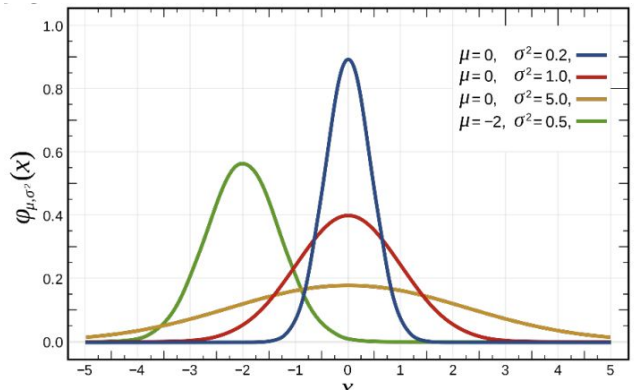
Q: How to estimate $P(xi \mid y)$ from data?

- **Basic principles of fitting likelihood parameters from data**
  - MLE (maximum likelihood estimation): Choose the parameter that maximizes the likelihood of the data
  - MAP (maximum a priori): Choose the parameter that maximizes the data likelihood and its own prior
    - As Warren Buffet says, it's not just about maximizing expected return – it's about making sure there are no zeros.

Bernoulli (x is binary; y is discrete)

$$P(x_i|y=k) = \theta_{ki}^{x_i}(1-\theta_{ki})^{1-x_i}$$

Categorical (x is has multiple discrete values, y is discrete)



Naive Bayes

Q: How to estimate $P(xi \mid y)$ from data?

- xi is Gaussian (aka Normal), y is discrete
- ($y - x$i) is Gaussian
- xi and y are jointly Gaussian
    - $N(.)$ stands for normal distribution with given value, mean, and (co-)variance
- *xi* is continuous (non-Gaussian), *y* is discrete
    - First turn *x* into discrete (e.g. if values range [0, 1), assign x = floor(x*10)
    - Now can estimate as categorical
- If *x* is text, e.g. “blue”, “orange”, “green”
    - Map each possible text value into an integer and solve as categorical

Naive Bayes

Q: How to estimate $P(y)$?

- **Three options:**
  - Assume that y is “uniform” (every value is equally likely) and ignore
  - If y is discrete, count
  - If y is continuous, model as Gaussian or convert to discrete and count

Naive Bayes

Simple Naive Bayes example

- **Suppose I want to classify a fruit based on description**
  - Features: weight, color, shape, whether it's hard
  - E.g.
    - 0.5 lb, "red", "round", yes
    - 15 lb, "green", "oval", yes
    - 0.01 lb, "purple", "round", no

Q1: What are these three fruit?

Q2: How might you model P(xi |fruit) for each of these four features?

Naive Bayes

Simple Naive Bayes example

- **Suppose I want to classify a fruit based on description**
  - Features: weight, color, shape, whether it's hard
  - E.g.
    - 0.5 lb, "red", "round", yes — Apple
    - 15 lb, "green", "oval", yes — Watermelon
    - 0.01 lb, "purple", "round", no — Grape
  - Model P(weight | fruit) as a Gaussian
  - Model P(color | fruit) as a discrete distribution (multinomial)
  - Model P(shape | fruit) as a categorical
  - Model P(is_hard | fruit) as a Bernoulli (binary)

Naive Bayes

Q: How to predict y from x?

- **If *y* is discrete: 1. 2.**
  - Compute $P(x, y)$ for each value of y
  - Choose value with maximum likelihood

Turning product into sum of logs is an important frequently used trick for argmax/argmin!

Naive Bayes

Q: How to predict y from x when $(y - x)$ is Gaussian

If y is continuous,

$$\frac{\partial}{\partial y} \sum_i \log P(x_i|y) + \log P(y) = 0$$

General formulation (set partial derivative wrt y of $\log P(x, y)$ to 0)

$$\frac{\partial}{\partial y} \sum_i -\frac{1}{2}\frac{(y - x_i - \mu_i)^2}{\sigma_i^2} - \frac{1}{2}\frac{(y - \mu_y)^2}{\sigma_y^2} = 0$$

Example of Temperature regression: $y - x_i$ is Gaussian

$$\frac{\partial}{\partial y} \sum_i \frac{1}{2}\frac{-y^2}{\sigma_i^2} + \frac{yx_i}{\sigma_i^2} + \frac{y\mu_i}{\sigma_i^2} - \frac{1}{2}\frac{y^2}{\sigma_y^2} + \frac{y\mu_y}{\sigma_y^2} = 0$$

$$\sum_i \left(\frac{-y}{\sigma_i^2} + \frac{x}{\sigma_i^2} + \frac{\mu_i}{\sigma_i^2}\right) - (y - \mu_y)/\sigma_y^2 = 0$$

$$y\left(\sum_i \frac{1}{\sigma_i^2} + \frac{1}{\sigma_y^2}\right) = \sum_i \frac{x + \mu_i}{\sigma_i^2} + \frac{\mu_y}{\sigma_y^2}$$

$$y = \frac{1}{\sum_i \frac{1}{\sigma_i^2} + \frac{1}{\sigma_y^2}} \cdot \left[\sum_i \frac{x + \mu_i}{\sigma_i^2} + \frac{\mu_y}{\sigma_y^2}\right]$$

$$y = \frac{1}{\sum_i w_i + w_y} \cdot \left[\sum_i (x + \mu_i) w_i + \mu_y \cdot w_y\right]$$

Prediction is weighted average of means, where weights are inverse variance

$$P(x_i|y) \sim N(y - x_i, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{1}{2}\frac{(y - x_i - \mu_i)^2}{\sigma_i^2}\right)$$

Naive Bayes

Using priors

- Priors on the likelihood parameters prevent a single feature from having zero or extremely low likelihood due to insufficient training data
- Discrete: initialize counts with α (e.g. $\alpha = 1$)

  $P(x_i=v|y=k) = (\alpha + \text{count}(x_i=v, y=k)) / \text{sum}_v[\alpha + \text{count}(x_i=v, y=k)]$

```
theta_kiv[k,i,v] = (np.sum((X[:,i]==v) & (y==k))+alpha) / (np.sum(y==k)+alpha*num_v)
```

- Continuous: add some ϵ to the variance (e.g. $\epsilon = 0.1/N$)
  - For multivariate, add to diagonal of covariance

```
std[i] = np.std(y-X[:,i], axis=0)+np.sqrt(0.1/len(X))
```

Naive Bayes

MLE and MAP estimates of binary variable likelihoods

- MLE (maximize data likelihood)

$$P(x = 1|y = 1) = \frac{\sum_n \delta(x_n = 1, y_n = 1)}{\sum_n \delta(x_n = 0, y_n = 1) + \sum_n \delta(x_n = 1, y_n = 1)}$$

- MAP (maximum a posteriori) with prior α

$$P(x = 1|y = 1) = \frac{\alpha + \sum_n \delta(x_n = 1, y_n = 1)}{(\alpha + \sum_n \delta(x_n = 0, y_n = 1)) + (\alpha + \sum_n \delta(x_n = 1, y_n = 1))}$$

- This is a Bayesian prior that implies $P(x = 0 \mid y) \approx P(x = 1|y)$, unless data tells us differently
- Similar concept to regularization that we saw in linear regression and classification
- Important because it avoids zeros that could dominate the overall likelihood and provides a more stable estimate with limited data
- With more data, the prior has less effect

Naive Bayes

Example: estimate joint probability under Naïve Bayes assumption

#	x1	x2	y
1	1	1	1
2	0	1	1
3	1	0	0
4	0	1	0
5	1	1	1
6	1	0	0
7	1	0	1
8	0	1	0

$P(x1|y)$

x1	$y=0$	$y=1$
0		
1		

$P(x2|y)$

x2	$y=0$	$y=1$
0		
1		

	$y=0$	$y=1$
$P(y)$		

$P(y=0, x1=1, x2=1) = ?$

$P(y=1, x1=1, x2=1) = ?$

$P(y=0|x1=1, x2=1) = ?$

Naive Bayes

Example: estimate joint probability under Naïve Bayes assumption

#	x1	x2	y
1	1	1	1
2	0	1	1
3	1	0	0
4	0	1	0
5	1	1	1
6	1	0	0
7	1	0	1
8	0	1	0

$P(x1|y)$

x1	$y=0$	$y=1$
0	2/4	1/4
1	2/4	3/4

$P(x2|y)$

x2	$y=0$	$y=1$
0	2/4	1/4
1	2/4	3/4

	$y=0$	$y=1$
$P(y)$	2/4	2/4

$$P(y=0, x1=1, x2=1) = 1/8$$

$$P(y=1, x1=1, x2=1) = 9/32$$

$$P(y=0|x1=1, x2=1) = 4/13$$

Naive Bayes

Prior over parameters: initialize each count with α

#	x1	x2	y
1	1	1	1
2	0	1	1
3	1	0	0
4	0	1	0
5	1	1	1
6	1	0	0
7	1	0	1
8	0	1	0

$\alpha = 1$

$P(x1|y)$

x1	$y=0$	$y=1$
0	2/4	1/4
1	2/4	3/4

→

x1	$y=0$	$y=1$
0	3/6	2/6
1	3/6	4/6

$P(x2|y)$

x2	$y=0$	$y=1$
0	2/4	1/4
1	2/4	3/4

→

x2	$y=0$	$y=1$
0	3/6	2/6
1	3/6	4/6

	$y=0$	$y=1$
$P(y)$	2/4	2/4

→

$y=0$	$y=1$
2/4	2/4

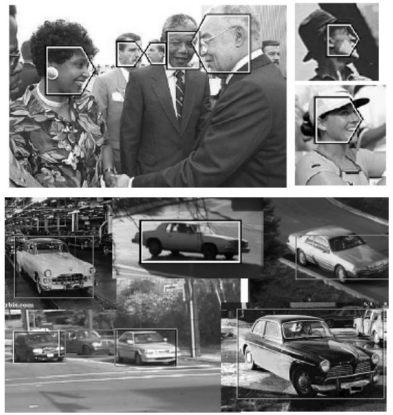
Naive Bayes

Use case: “Semi-naïve Bayes” object detection

- Best performing face/car detector in 2000-2005
- Model probabilities of small groups of features (wavelet coefficients)
- Search for groupings, discretize features, estimate parameters

A Statistical Method for 3D Object Detection Applied to Faces and Cars

Henry Schneiderman and Takeo Kanade



$$\frac{\prod_{x,y \in \text{region}} \prod_{k=1}^{17} P_k(pattern_k(x,y), x, y | \text{object})}{\prod_{x,y \in \text{region}} \prod_{k=1}^{17} P_k(pattern_k(x,y), x, y | \text{non-object})} > \lambda$$

https://www.cs.cmu.edu/afs/cs.cmu.edu/user/hws/www/CVPR00.pdf

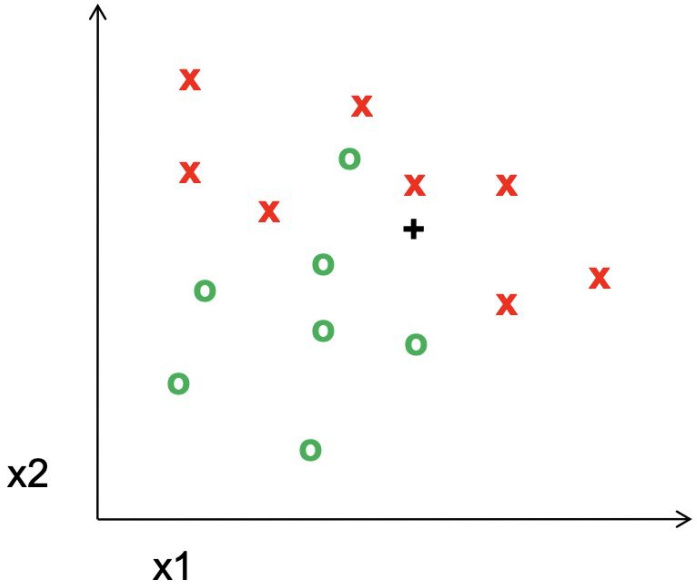
Naive Bayes

Naïve Bayes Summary

- **Pros**
  - Easy and fast to train
  - Fast inference
  - Can be used with continuous, discrete, or mixed features
- **Cons**
  - Does not account for feature interactions
  - Does not provide good confidence estimate
- **Notes**
  - Best when used with discrete variables, variables that are well fit by Gaussian, or kernel density estimation

K-Nearest Neighbors

Q: What class do you think the '+' belongs to?



K-Nearest Neighbors

Key principle of machine learning

- Given feature/target pairs in machine learning $(X_1, y_1), \ldots, (X_n, y_n)$:

  If X_i is similar to X_j , then y_i is probably similar to y_j

  With variations on how you define similarity and make predictions based on multiple similar examples, this principle underlies virtually all ML algorithms.

K-Nearest Neighbors

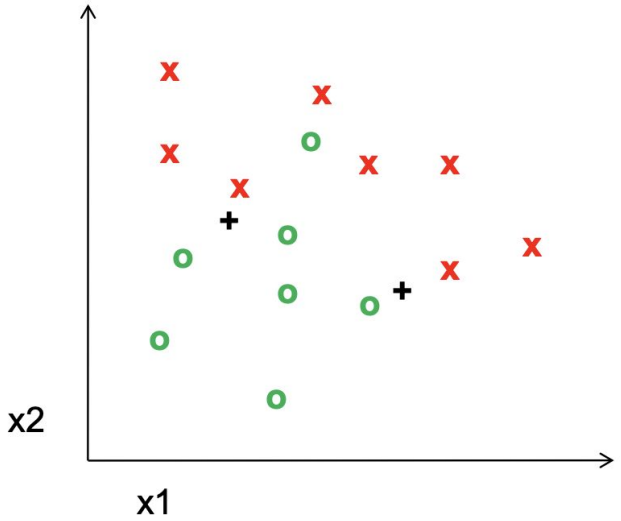
Nearest neighbor algorithm

- For given test features, assign the label / target value of the most similar training features

  1. $i^* = \underset{i}{\text{argmin}}\, distance(X_{train}[i], X_{test})$
  2. $y_{test} = y_{train}[i^*]$

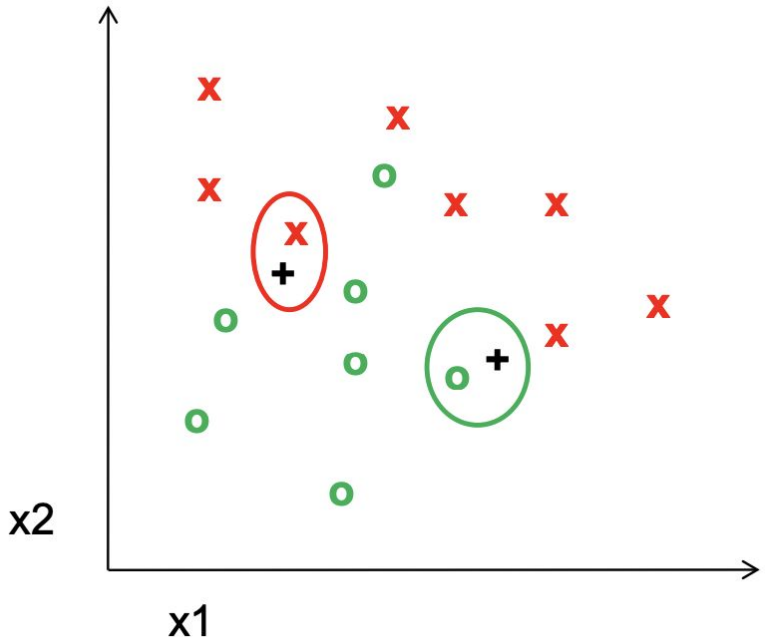
  Distance function is up to designer. Simplest is L2 distance.

- K-nearest neighbor: predict based on K closest training samples



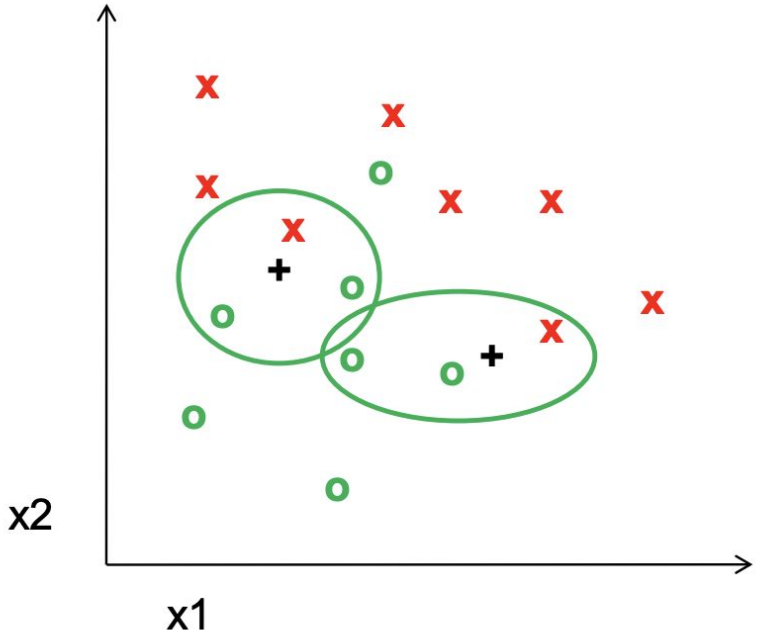
K-Nearest Neighbors

1-nearest neighbor



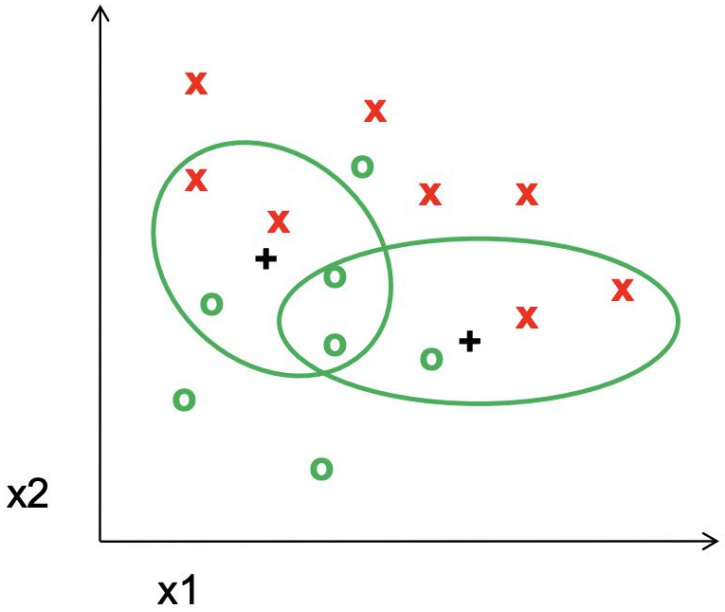
K-Nearest Neighbors

3-nearest neighbor



K-Nearest Neighbors

5-nearest neighbor



K-Nearest Neighbors

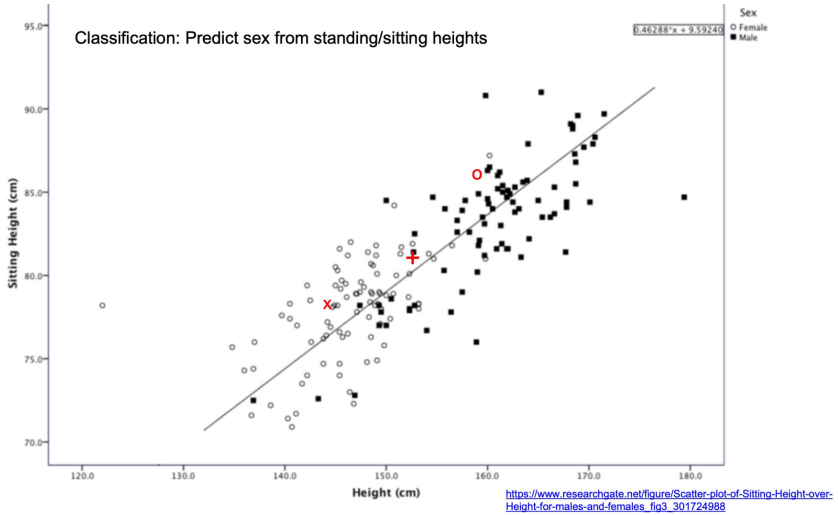
KNN Distance Function

- Euclidean or L2 norm: $\|\boldsymbol{x} - \boldsymbol{t}\|_2 = \sqrt{\sum_k (x_k - t_k)^2}$
  - Assumes all dimensions are equally scaled
  - Dominated by biggest differences
- Citi Block or L1 norm: $\|\boldsymbol{x} - \boldsymbol{t}\|_1 = \sum_k |x_k - t_k|$
  - Assumes all dimensions are equally scaled
  - Less sensitive to very large differences along one dimension
- Mahalanobis distance: $d_M(\boldsymbol{x}, \boldsymbol{t}) = \sqrt{(\boldsymbol{x} - \boldsymbol{t})^T \sum^{-1} (\boldsymbol{x} - \boldsymbol{t})}$
  - Normalized by inverse feature covariance matrix: “whitening”
  - When diagonal covariance is assumed, this is equivalent to scaling each dimension by $1/\sigma_k$

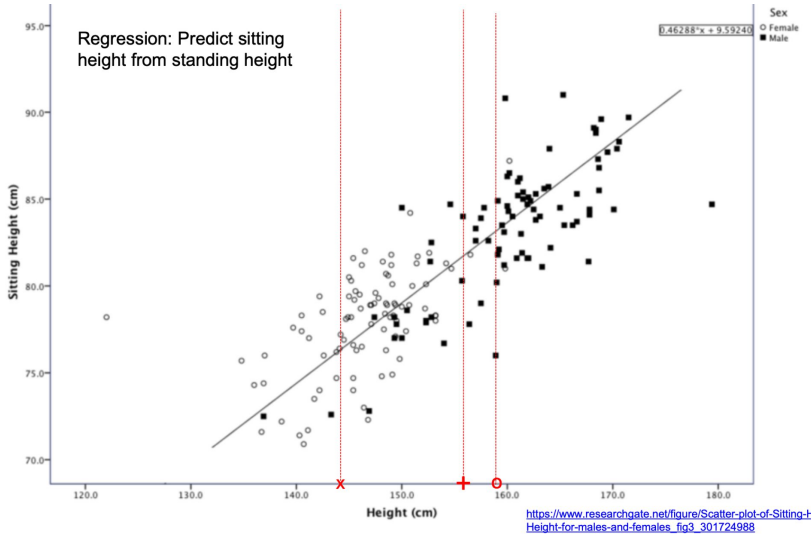
K-Nearest Neighbors

KNN Classification vs Regression

- For classification, prediction is usually the mode or most common class of the returned labels.
- For regression, prediction is usually the arithmetic mean (average, informally) of the returned values.



https://www.researchgate.net/figure/Scatter-plot-of-Sitting-Height-over-Height-for-males-and-females_fig3_301724988

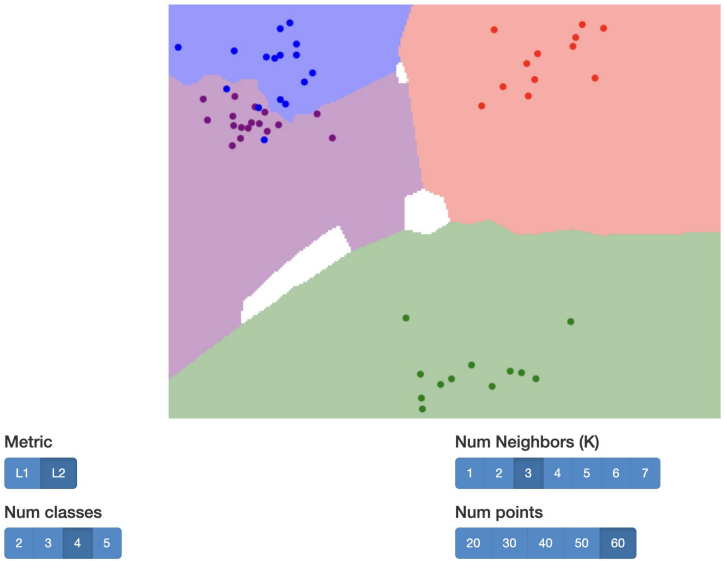


https://www.researchgate.net/figure/Scatter-plot-of-Sitting-Height-over-Height-for-males-and-females_fig3_301724988

K-Nearest Neighbors

KNN Classification Demo

- http://vision.stanford.edu/teaching/cs231n-demos/knn/



K-Nearest Neighbors

Comments on K-NN

- Simple: an excellent baseline and sometimes hard to beat
  - Naturally scales with data: it may be the only choice when you have one example per class, and is still often achieves good performance when you have many
  - Higher K gives smoother functions

- Can be slow… but there are tricks to speed it up, e.g.
  - $\underset{i}{\text{argmin}} \|x_i - x_t\|_2 = \underset{i}{\text{argmin}} \left(x_i^T x_i - 2x_i x_t + x_t^T x_t\right) = \underset{i}{\text{argmin}} \left(x_i^T x_i - 2x_i x_t\right)$ can be precomputed
  - FAISS
  - Approximate search like FLANN or LSH

- No training time (unless you learn a distance function)

- With infinite examples, 1-NN provably has error that is at most twice Bayes optimal error (but we never have infinite examples)

K-Nearest Neighbors

KNN Usage Example: Deep Face

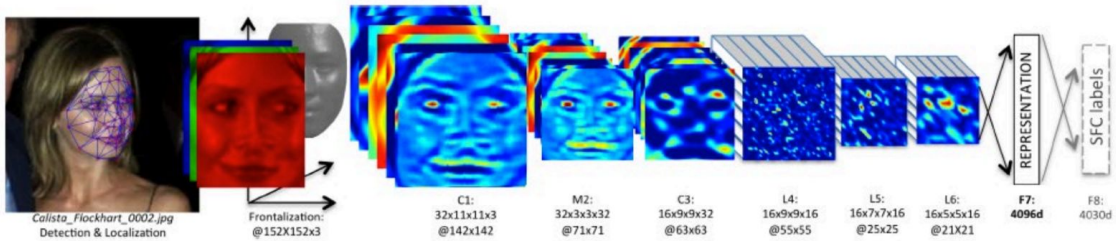
DeepFace: Closing the Gap to Human-Level Performance in Face Verification

Yaniv Taigman Ming Yang Marc'Aurelio Ranzato Lior Wolf

Facebook AI Research
Menlo Park, CA, USA
{yaniv, mingyang, ranzato}@fb.com

Tel Aviv University
Tel Aviv, Israel
wolf@cs.tau.ac.il

CVPR 2014



1. Detect facial features
2. Align faces to be frontal
3. Extract features using deep network while training classifier to label image into person (dataset based on employee faces)
4. In testing, extract features from deep network and use nearest neighbor classifier to assign identity

- Performs similarly to humans in the LFW dataset (labeled faces in the wild)
- Can be used to organize photo albums, identifying celebrities, or alert user when someone posts an image of them
- If this is used in a commercial deployment, what might be some unintended consequences?
- This algorithm is used by Facebook (though with expanded training data)

K-Nearest Neighbors

KNN Summary

- **Key Assumptions**
  - Samples with similar input features will have similar output prediction
  - Depending on distance measure, may assume all dimensions are equally important
- **Model Parameters**
  - Features and predictions of the training set
- **Designs**
  - K (number of nearest neighbors to use for prediction)
  - How to combine multiple predictions if K > 1
  - Feature design (selection, transformations)
  - Distance function (e.g. L2, L1, Mahalanobis)

K-Nearest Neighbors

KNN Summary

- **When to Use**
  - Few examples per class, many classes
  - Features are all roughly equally important
  - Training data available for prediction changes frequently
  - Can be applied to classification or regression, with discrete or continuous feature
  - Most powerful when combined with feature learning
- **When Not to Use**
  - Many examples are available per class (feature learning with linear classifier may be better)
  - Limited storage (cannot store many training examples)
  - Limited computation (linear model may be faster to evaluate)

K-Nearest Neighbors

T/F: Why?

- 1-NN will never have higher training error than 3- NN for classification and regression.
- For 1-NN classification, you cannot remove a training sample without affecting at least some portion of the decision boundary.

Classification And Regression Trees (CART)

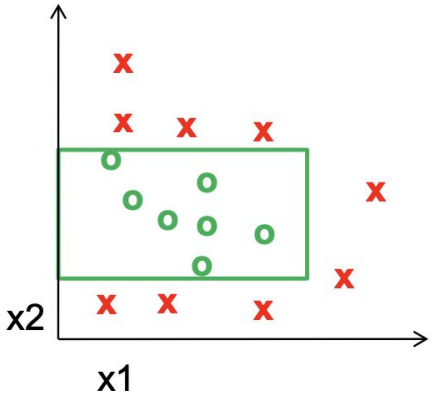
Recap of approaches we've seen so far

- Nearest neighbor is widely used
  - Super-powers: can instantly learn new classes and predict from one or many examples
- Naïve Bayes represents a common assumption as part of density estimation, more typical as part of an approach rather than the final predictor
  - Super-powers: Fast estimation from lots of data; not terrible estimation from limited data
- Logistic Regression is widely used
  - Super-powers: Effective prediction from high-dimensional features
- Linear Regression is widely used
  - Super-powers: Can extrapolate, explain relationships, and predict continuous values from many variables
- Almost all algorithms involve nearest neighbor, logistic regression, or linear regression
  - The main learning challenge is typically **feature learning**

Classification And Regression Trees (CART)

Recap of approaches we've seen so far

- So far, we've seen two main choices for how to use features
  - Nearest neighbor uses all the features jointly to find similar examples
  - Linear models make predictions out of weighted sums of the features
- If you wanted to give someone a rule to split the 'o' from the 'x', what other idea might you try?



If x2 < 0.6 and x2 > 0.2 and x2 < 0.7, 'o' Else 'x'

Q: Can we learn these kinds of rules automatically?

Classification And Regression Trees (CART)

Decision trees

- **Training:** Iteratively choose the attribute and split value that best separates the classes for the data in the current node
- Combines feature selection/modeling with prediction

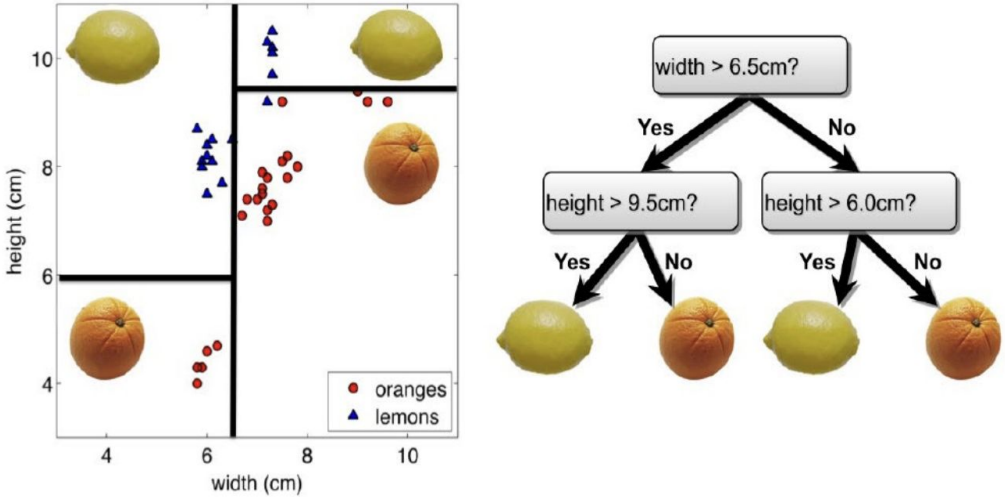


Fig Credit: Zemel, Urtasun, Fidler

Classification And Regression Trees (CART)

Decision trees

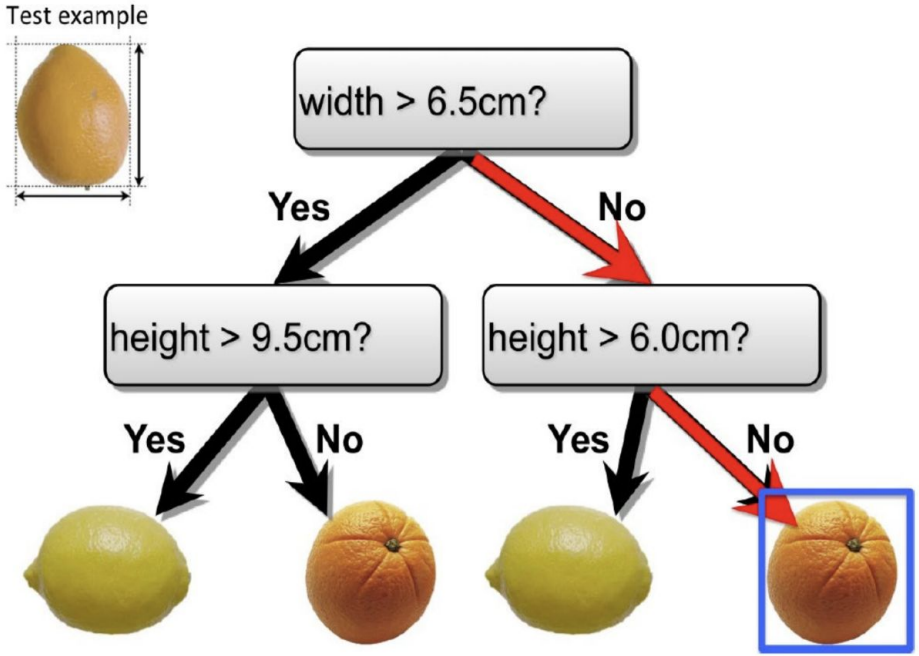


Fig Credit: Zemel, Urtasun, Fidler

Classification And Regression Trees (CART)

Example with discrete inputs

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Attributes:

1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: whether we are hungry.
5. Patrons: how many people are in the restaurant (values are None, Some, and Full).
6. Price: the restaurant's price range ($, $$, $$$).
7. Raining: whether it is raining outside.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (French, Italian, Thai or Burger).
10. WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

Slide Credit: Zemel, Urtasun, Fidler

Classification And Regression Trees (CART)

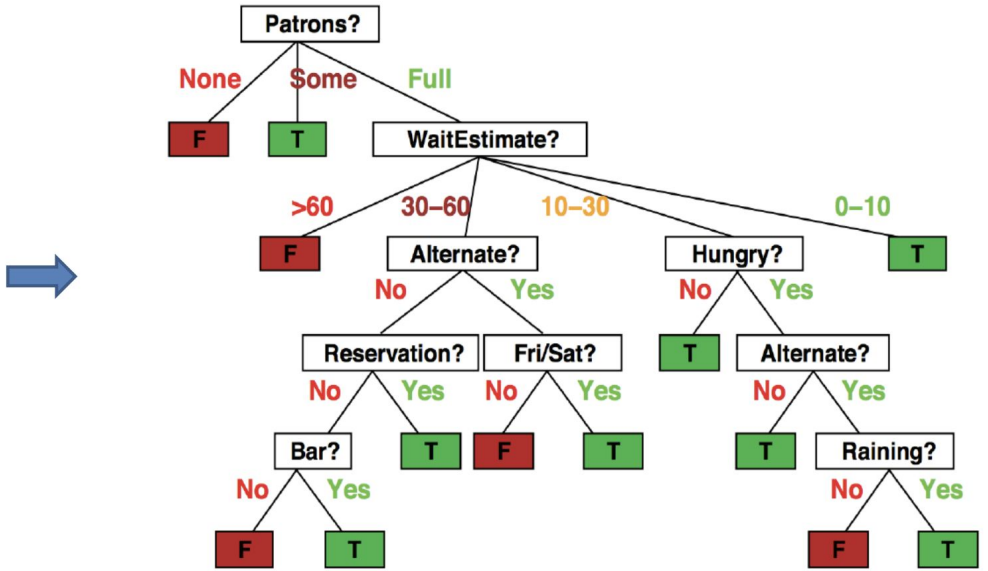
Example with discrete inputs

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Attributes:

1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: whether we are hungry.
5. Patrons: how many people are in the restaurant (values are None, Some, and Full).
6. Price: the restaurant's price range ($, $$, $$$).
7. Raining: whether it is raining outside.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (French, Italian, Thai or Burger).
10. WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

- The tree to decide whether to wait (T) or not (F)



Slide Credit: Zemel, Urtasun, Fidler

Classification And Regression Trees (CART)

Example with discrete inputs

- Internal nodes test attributes
- Branching is determined by attribute value
- Leaf nodes are outputs (class assignments)

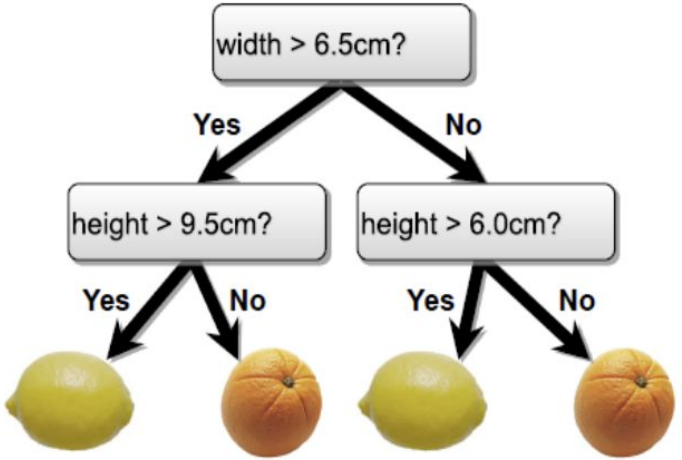
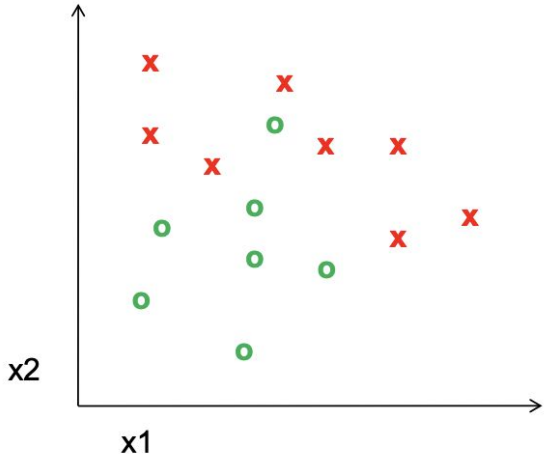


Figure Source: Zemel, Urtasun, Fidler

Classification And Regression Trees (CART)

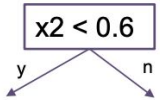
Decision tree algorithm

- **Training**
  - Recursively, for each node in tree:
    - If labels in the node are mixed:
      a. Choose attribute and split values based on data that reaches each node
      b. Branch and create 2 (or more) nodes
    - Return

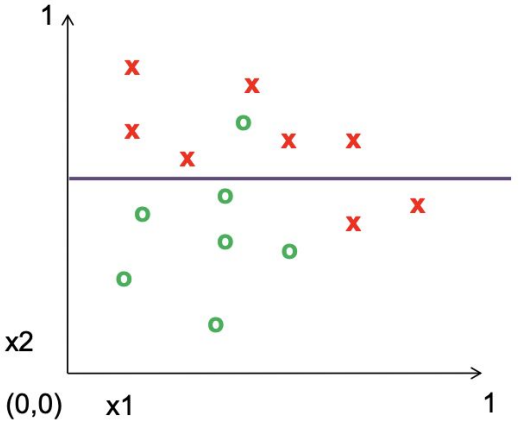


Classification And Regression Trees (CART)

Decision tree algorithm



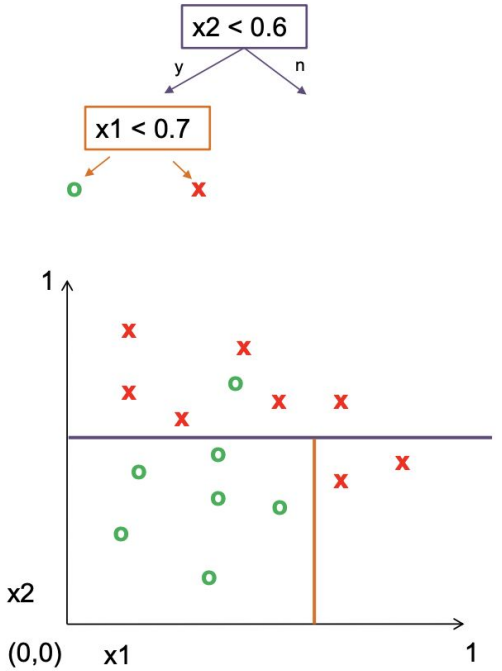
- **Training**
  - Recursively, for each node in tree:
    - If labels in the node are mixed:
      a. Choose attribute and split values based on data that reaches each node
      b. Branch and create 2 (or more) nodes
    - Return



Classification And Regression Trees (CART)

Decision tree algorithm

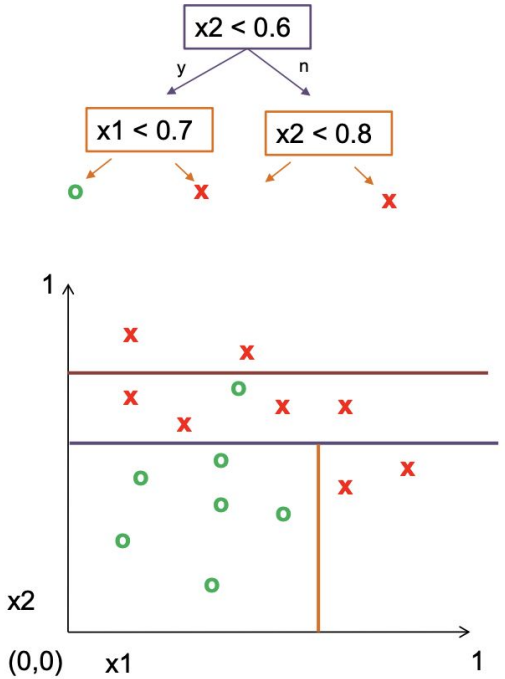
- **Training**
  - Recursively, for each node in tree:
    - If labels in the node are mixed:
      a. Choose attribute and split values based on data that reaches each node
      b. Branch and create 2 (or more) nodes
    - Return



Classification And Regression Trees (CART)

Decision tree algorithm

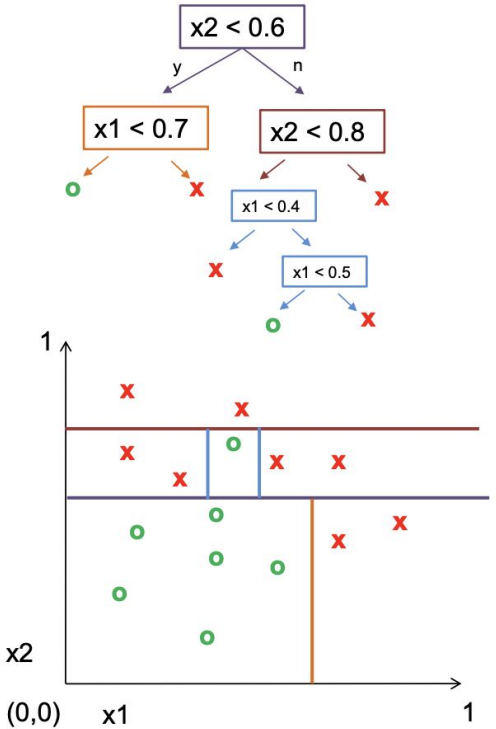
- **Training**
  - Recursively, for each node in tree:
    - If labels in the node are mixed:
      a. Choose attribute and split values based on data that reaches each node
      b. Branch and create 2 (or more) nodes
    - Return



Classification And Regression Trees (CART)

Decision tree algorithm

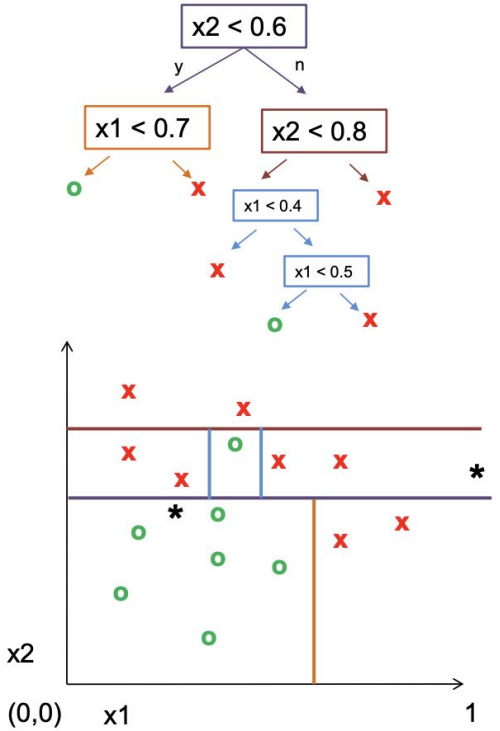
- **Training**
  - Recursively, for each node in tree:
    - If labels in the node are mixed:
      a. Choose attribute and split values based on data that reaches each node
      b. Branch and create 2 (or more) nodes
    - Return



Classification And Regression Trees (CART)

Decision tree algorithm

- **Training**
  - Recursively, for each node in tree:
    - If labels in the node are mixed:
      a. Choose attribute and split values based on data that reaches each node
      b. Branch and create 2 (or more) nodes
    - Return



Classification And Regression Trees (CART)

Q: How do you choose what/where to split?

- **What attribute is better to split on X1 or X2?**

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

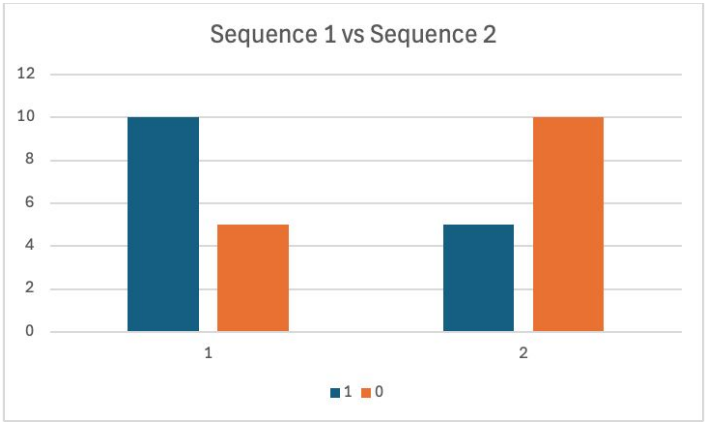
Idea: Use counts at leaves to define probability distributions, so we can measure uncertainty.

Slide Source: Zemel, Urtasun, Fidler

Classification And Regression Trees (CART)

Quantifying Uncertainty: Coin Flip Example

- **Sequence 1:** 1 1 1 1 0 1 0 1 0 1 1 1 0 0 1 ……
- **Sequence 2:** 0 1 0 0 0 0 0 0 1 1 1 0 0 0 1 ……

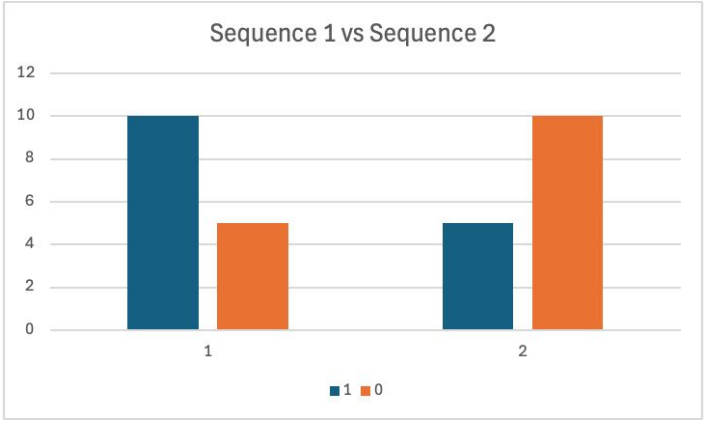


Classification And Regression Trees (CART)

Quantifying Uncertainty: Coin Flip Example

Entropy H:

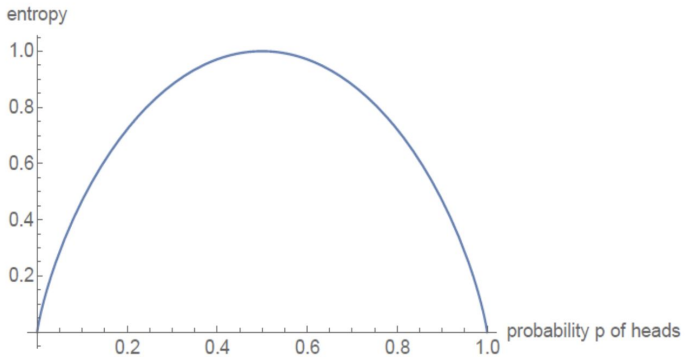
$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$



Classification And Regression Trees (CART)

Quantifying Uncertainty: Coin Flip Example

Entropy: $H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$



Classification And Regression Trees (CART)

Entropy of a Joint Distribution

Example: $X = \{\text{Raining, Not raining}\}$, $Y = \{\text{Cloudy, Not cloudy}\}$

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

$$
\begin{aligned}
H(X,Y) &= -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_2 p(x,y) \\
&= -\frac{24}{100}\log_2\frac{24}{100} - \frac{1}{100}\log_2\frac{1}{100} - \frac{25}{100}\log_2\frac{25}{100} - \frac{50}{100}\log_2\frac{50}{100} \\
&\approx 1.56\text{bits}
\end{aligned}
$$

Classification And Regression Trees (CART)

Specific Conditional Entropy

Example: $X = \{\text{Raining, Not raining}\}$, $Y = \{\text{Cloudy, Not cloudy}\}$

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

What is the entropy of cloudiness Y, **given that it is raining**?

$$\begin{aligned} H(Y|X=x) &= -\sum_{y\in Y} p(y|x)\log_2 p(y|x) \\ &= -\frac{24}{25}\log_2\frac{24}{25} - \frac{1}{25}\log_2\frac{1}{25} \\ &\approx 0.24\text{bits} \end{aligned}$$

We used: $p(y|x) = \frac{p(x,y)}{p(x)}$, and $p(x) = \sum_y p(x,y)$ (sum in a row)

Classification And Regression Trees (CART)

Conditional Entropy

Example: $X = \{\text{Raining, Not raining}\}$, $Y = \{\text{Cloudy, Not cloudy}\}$

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

The expected conditional entropy:

$$\begin{aligned} H(Y|X) &= \sum_{x \in X} p(x) H(Y|X = x) \\ &= -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(y|x) \end{aligned}$$

Classification And Regression Trees (CART)

Conditional Entropy

Example: $X = \{\text{Raining, Not raining}\}$, $Y = \{\text{Cloudy, Not cloudy}\}$

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

What is the entropy of cloudiness, given the knowledge of whether or not it is raining?

$$\begin{aligned} H(Y|X) &= \sum_{x \in X} p(x) H(Y|X = x) \\ &= \frac{1}{4} H(\text{cloudy}|\text{is raining}) + \frac{3}{4} H(\text{cloudy}|\text{not raining}) \\ &\approx 0.75 \text{ bits} \end{aligned}$$

Classification And Regression Trees (CART)

Conditional Entropy

- **Some useful properties:**
  - H is always non-negative
  - Chain rule: $H(X, Y) = H(X|Y) + H(Y) = H(Y|X) + H(X)$
  - If X and Y independent, then X doesn't tell us anything about Y: $H(Y|X) = H(Y)$
  - But Y tells us everything about Y: $H(Y|Y) = 0$
  - By knowing X, we can only decrease uncertainty about Y: $H(Y|X) <= H(Y)$

Classification And Regression Trees (CART)

Information Gain

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

- How much information about cloudiness do we get by discovering whether it is raining?

$$\begin{aligned} IG(Y|X) &= H(Y) - H(Y|X) \\ &\approx 0.25 \text{ bits} \end{aligned}$$

- If X is completely uninformative about Y: IG(Y|X) = 0
- If X is completely informative about Y: IG(Y|X) = H(Y)
- How can we use this to construct our decision tree?

Classification And Regression Trees (CART)

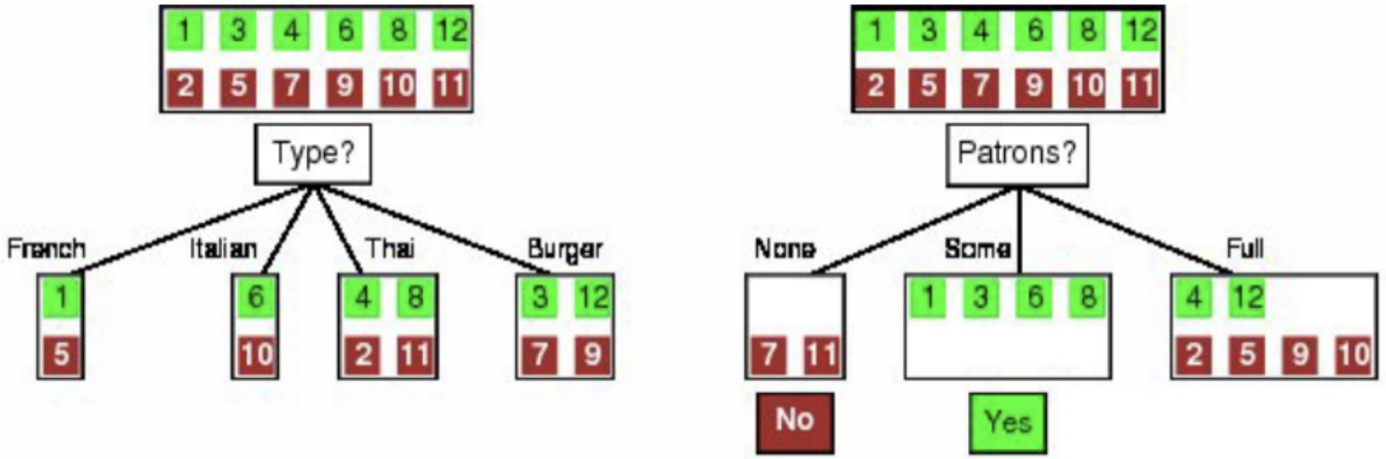
Constructing decision tree

- **Training**
  - Recursively, for each node in tree:
    - If labels in the node are mixed:
      a. **Choose attribute and split values based on data that reaches each node**
      b. Branch and create 2 (or more) nodes
    - Return

1. Measure information gain
   - For each discrete attribute: compute information gain of split
   - For each continuous attribute: select most informative threshold and compute its information gain. Can be done efficiently based on sorted values.
2. Select attribute / threshold with highest information gain

Classification And Regression Trees (CART)

Constructing decision tree

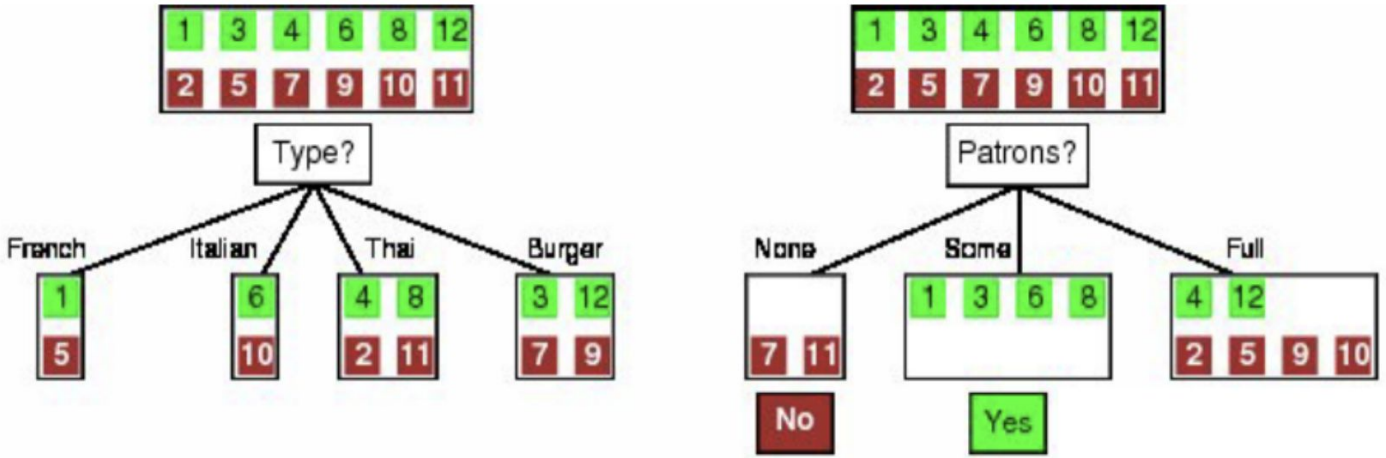


Q: Is it better to split based on type or patrons?

Slide Source: Zemel, Urtasun, Fidler

Classification And Regression Trees (CART)

Constructing decision tree



$$IG(Y) = H(Y) - H(Y|X)$$

$$IG(type) = 1 - \left[\frac{2}{12}H(Y|\text{Fr.}) + \frac{2}{12}H(Y|\text{It.}) + \frac{4}{12}H(Y|\text{Thai}) + \frac{4}{12}H(Y|\text{Bur.})\right] = 0$$

$$IG(Patrons) = 1 - \left[\frac{2}{12}H(0,1) + \frac{4}{12}H(1,0) + \frac{6}{12}H(\frac{2}{6},\frac{4}{6})\right] \approx 0.541$$

Slide Source: Zemel, Urtasun, Fidler

Classification And Regression Trees (CART)

Q: What if you need to predict a continuous value?

- **Regression Tree**
  - Same idea, but choose splits to minimize sum squared error $\sum_{n \in node}(f_{node}(x_n) - y_n)^2$
  - $f_{node}(x_n)$ typically returns the mean prediction value of data points in the leaf node containing x_n
  - What are we minimizing?

Classification And Regression Trees (CART)

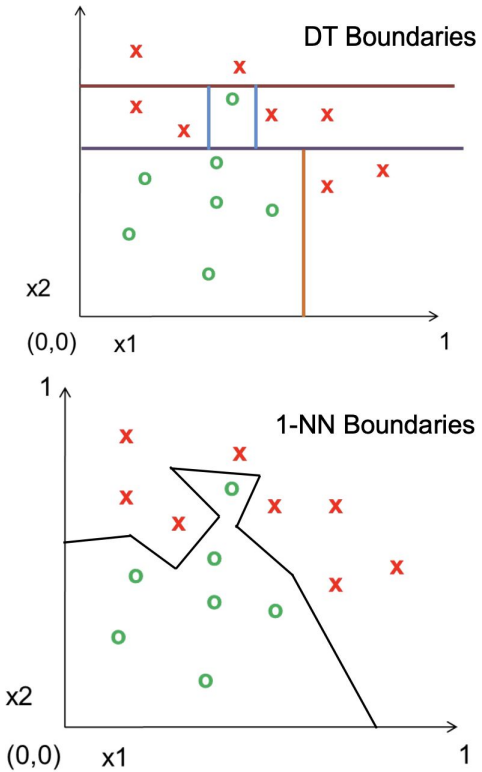
Variants

- Different splitting criteria, e.g. Gini index: $1 - \sum_i p_i^2$ (very similar result, a little faster to compute)
- Most commonly, split on one attribute at a time
  - In case of continuous vector data, can also split on linear projections of features
- Can stop early
  - when leaf node contains fewer than N_{min} points
  - when max tree depth is reached
- Can also predict multiple continuous values or multiple classes

Classification And Regression Trees (CART)

Decision Tree vs. 1-NN

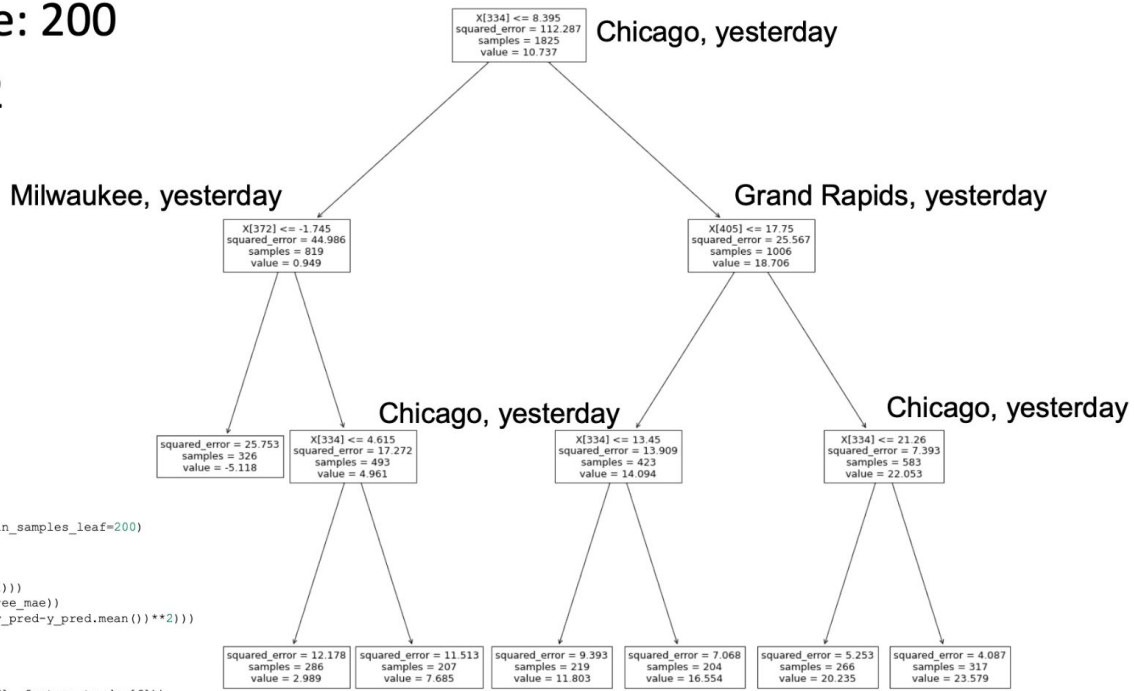
- Both have piecewise-linear decisions
- Decision tree is typically “axisaligned”
- Decision tree has ability for early stopping to improve generalization
- True power of decision trees arrives with ensembles (lots of small or randomized trees)



Classification And Regression Trees (CART)

Regression Tree for Temperature Prediction

- Min leaf size: 200
- RMSE= 3.42
- R^2=0.88




```
from sklearn import tree
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor(random_state=0, min_samples_leaf=200)
model.fit(x_train, y_train)
y_pred = model.predict(x_val)
tree_rmse = np.sqrt(np.mean((y_pred-y_val)**2))
tree_mae = np.sqrt(np.median(np.abs(y_pred-y_val)))
print('LR: RMSE={}, MAE={}'.format(tree_rmse, tree_mae))
print('R^2: {}'.format(1-tree_rmse**2/np.mean((y_pred-y_pred.mean())**2)))
plt.figure(figsize=(20,20))
tree.plot_tree(model)
plt.show()
for f in [334, 372, 405]:
  print('{}: {}, {}'.format(f, feature_to_city[f], feature_to_day[f]))
```

Grainger College of Engineering. (2024). CS 441 - Applied Machine Learning: Lecture 12 - Decision and Regression Trees.

Example: Algorithms in python

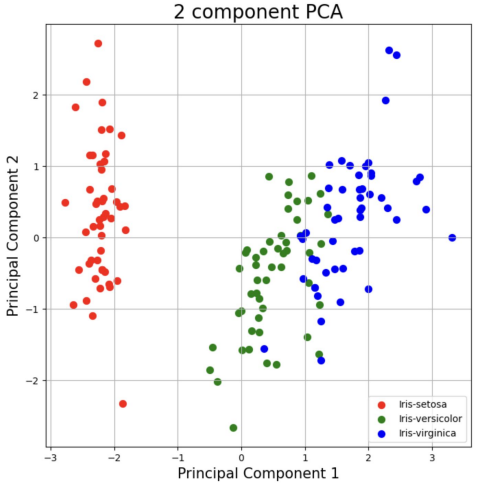
Principal Component Analysis（PCA）

- Iris flower data set
  - https://en.wikipedia.org/wiki/Iris_flower_data_set
  - **Description**:
    - The Iris dataset contains measurements of sepal and petal lengths for three species of iris flowers: **Setosa**, **Versicolor**, and **Virginica**.
    - Each sample has four features: Sepal Length, Sepal Width, Petal Length, and Petal Width.
    - It's a multi-class classification dataset with a total of 150 samples (50 samples per class).


Iris virginica

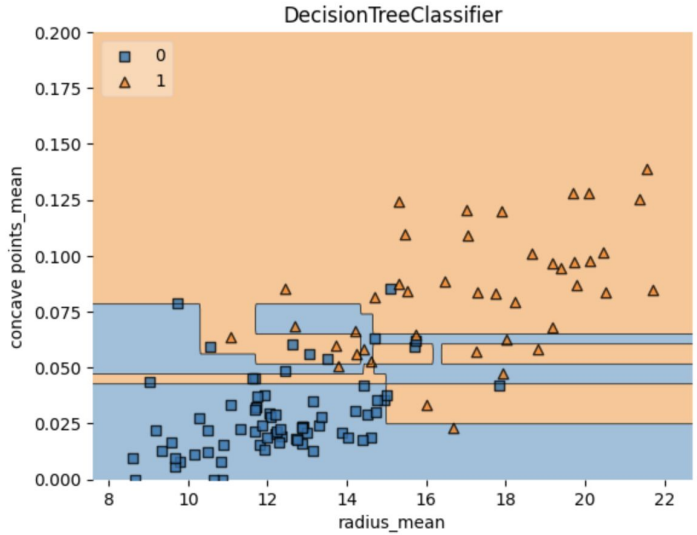

Iris versicolor

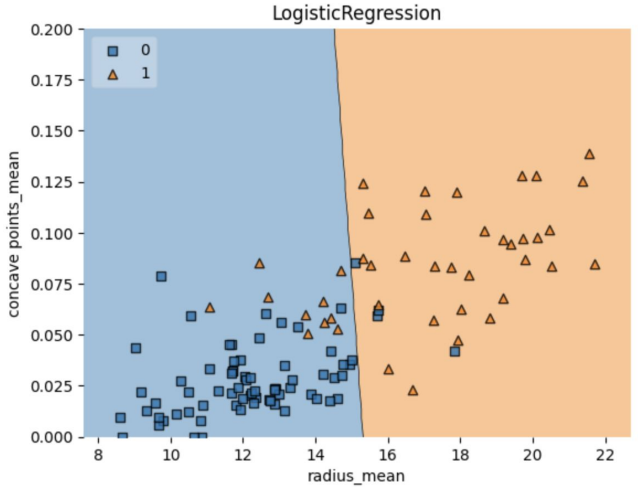

Iris setosa



Example: Algorithms in python

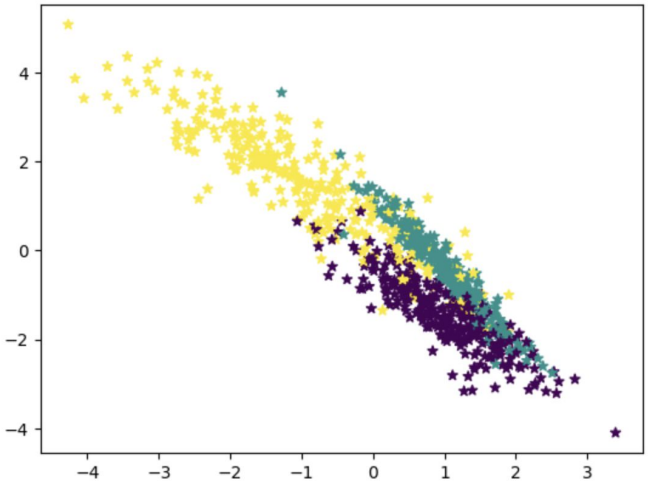
Classification And Regression Trees

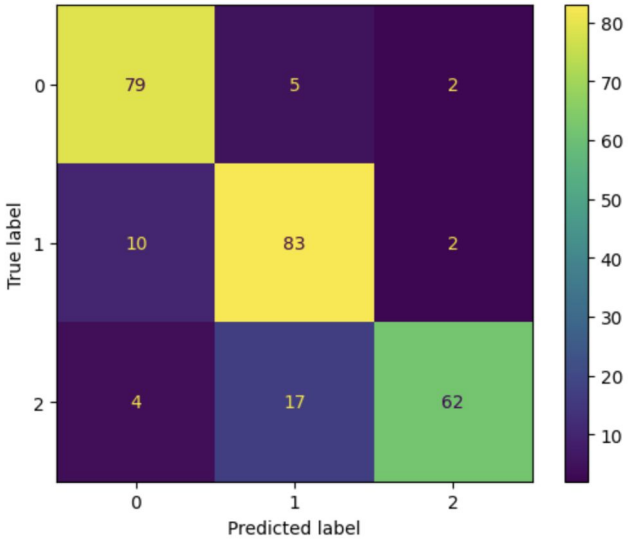




Example: Algorithms in python

Naive Bayes





Example: Algorithms in python

K-Nearest Neighbors