Principal Component Analysis - PCA

Principal Component Analysis - PCA

- Noise filtering
- Visualization
- Feature Extraction
- Stock market predictions
- Gene data analysis

Principal Component Analysis - PCA

- Identify patterns in data
- Detect the correlation between variables

Principal Component Analysis - PCA

Reduce the dimensions of a d-dimensional dataset by projecting it onto a (k)-dimensional subspace (where k<d)

Principal Component Analysis - PCA

- Standardize the data.
- Obtain the Eigenvectors and Eigenvalues from the covariance matrix or correlation matrix, or perform Singular Vector Decomposition.
- Sort eigenvalues in descending order and choose the k eigenvectors that correspond to the k largest eigenvalues where k is the number of dimensions of the new feature subspace ($k \leq d$)/.
- Construct the projection matrix $\mathbf{W}$ from the selected k eigenvectors.
- Transform the original dataset $\mathbf{X}$ via $\mathbf{W}$ to obtain a k-dimensional feature subspace $\mathbf{Y}$

https://plot.ly/ipython-notebooks/principal-component-analysis/

Linear Discriminant Analysis - LDA

Linear Discriminant Analysis - LDA

Linear Discriminant Analysis - LDA

- Used as a dimensionality reduction technique
- Used in the pre-processing step for pattern classification
- Has the goal to project a dataset onto a lower-dimensional space

Linear Discriminant Analysis - LDA

- Sounds similar to PCA right?

LDA differs because in addition to finding the component axises with LDA we Are interested in the axes that maximize the separation between multiple classes.

Linear Discriminant Analysis- LDA

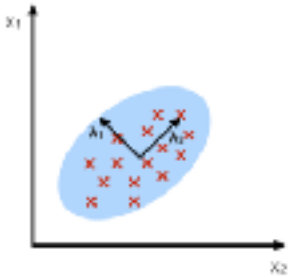
Breaking it down further:

The goal of LDA is to project a feature space (a dataset n-dimensional samples) onto a small subspace subspace k(where $k \leq n-1$) while maintaining the class-discriminatory information.

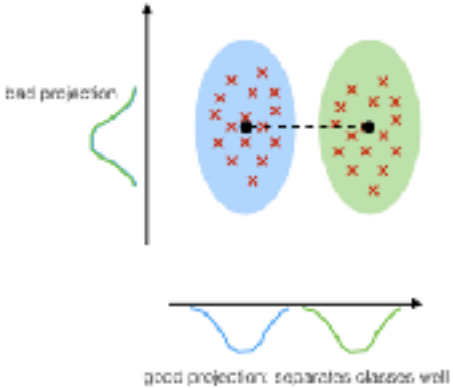
Both PCA and LDA are linear transformation techniques used for dimensional reduction. PCA is described as unsupervised but LDA is supervised because of the relation to the dependent variable.

Linear Discriminant Analysis - LDA

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



LDA: https://sebastianraschka.com/Articles/2014_python_lda.html

Linear Discriminant Analysis - LDA

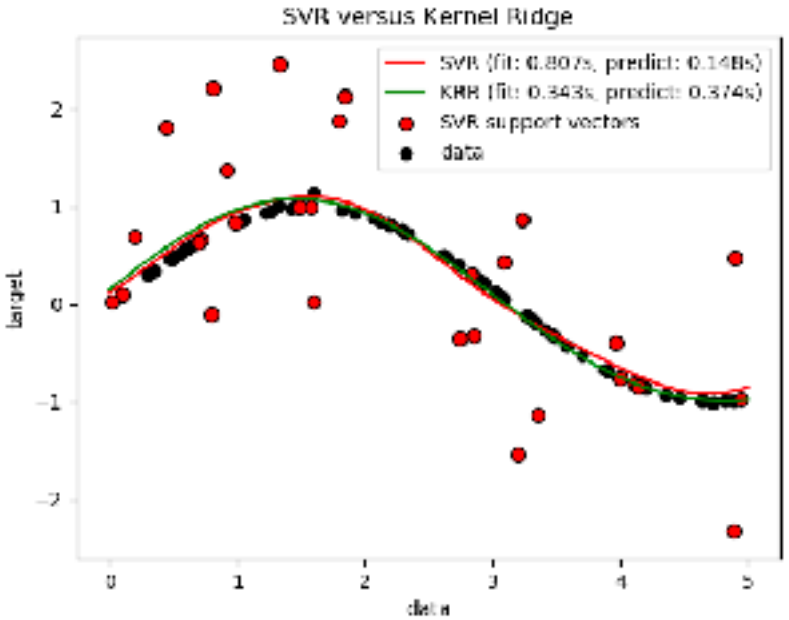
1. Compute the d-dimensional mean vectors for the different classes from the dataset.
2. Compute the scatter matrices (in-between-class and within-class scatter matrix).
3. Compute the eigenvectors $(\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_d)$ and corresponding eigenvalues $(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_d)$ for the scatter matrices.
4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $d \times k$ dimensional matrix $\boldsymbol{W}$ (where every column represents an eigenvector).
5. Use this $d \times k$ eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the matrix multiplication: $\boldsymbol{Y} = \boldsymbol{X} \times \boldsymbol{W}$ (where $\boldsymbol{X}$ is a $n \times d$-dimensional matrix representing the n samples, and $\boldsymbol{y}$ are the transformed $n \times k$-dimensional samples in the new subspace).

https://sebastianraschka.com/Articles/2014_python_lda.html

Linear Discriminant Analysis - LDA

Support Vector Regression - SVR

How does it work?



Support Vector Regression - SVR

Building a SVR

1. Collect a training set $\tau = \{\vec{X}, \vec{Y}\}$

2. Choose a kernel and it's parameters as well as any regularization needed.

3. Form the correlation matrix, $\vec{K}$

4. Train your machine, exactly or approximately, to get contraction coefficients $\vec{\alpha} = \{\alpha_i\}$

5. Use those coefficients, create your estimator $f(\vec{X}, \vec{\alpha}, x^\star) = y^\star$

Support Vector Regression - SVR

Next step is to choose a kernel

- Gaussian

Regularization

- Noise

Support Vector Regression - SVR

Correlation Matrix

$$K_{i,j} = \exp\left(\sum_k \theta_k \left|x_k^i - x_k^j\right|^2\right) + \epsilon\delta_{i,j}$$

Support Vector Regression - SVR

The main part of the algorithm $\bar{K}\vec{\alpha} = \vec{y}$

$\vec{y}$ is the vector of values corresponding to your training set,

$\bar{K}$ is your correlatiōn matrix

$\vec{\alpha}$ is a set of unknowns we need to solve for.

$$\vec{\alpha} = \bar{K}^{-1}\vec{y}$$

Support Vector Regression - SVR

- Once $\vec{\alpha}$ parameters known - form the estimator
- we use the coefficients we found during the optimization step and the kernel we started off with.

- To estimate the value y* for a test point, $\vec{x^*}$ - compute the correlation vector $\vec{k}$,

- $y^* = \vec{a} \cdot \vec{k}$

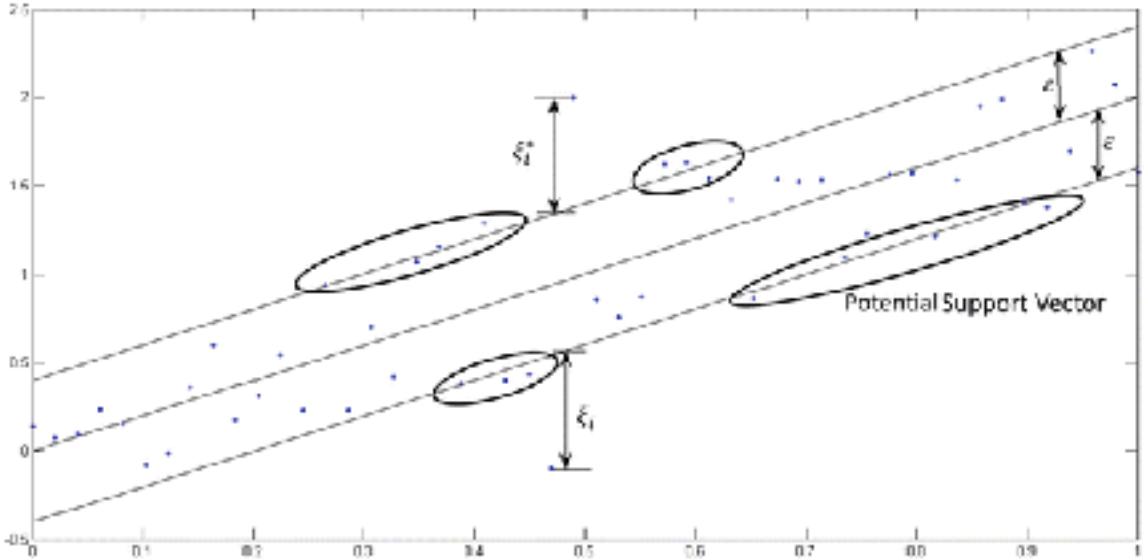
$$k_i = \exp\left(\sum_k \theta_k |x_k^i - x_k^\star|^2 \right)$$

Support Vector Regression - SVR

Let’s try to simplify everything:

SVR has a different regression goal compared to linear regression. In linear regression we are trying to minimize the error between the prediction and data. In SVR our goal is to make sure that errors do not exceed the threshold.

Support Vector Regression - SVR



https://link.springer.com/chapter/10.1007/978-1-4302-5990-9_4

Support Vector Regression - SVR

Notes: SVR is very useful for:

Since you are able to generate training points you then know what the “right” is. But it may be expensive to compute the answer for every new point needed.

- SVR and in particular gaussian processes are a very good method to use for efficient computations, by pre-computing a training set and then using a SVR machine to interpolate the results.

Support Vector Regression - SVR

In a classification problem, the vectors $\vec{X}$ are used to define
a hyperplane that separates the two different classes in your solution.

These vectors are used to perform linear regression. The vectors closest to the test point are referred to as support vectors. We can evaluate our function anywhere so any vectors could be closest to our test evaluation location.

Gaussian processes are a particular form of SVM. The difference between the two lies in choice of kernel and loss function. The functional form of the kernel determines which vectors in your training set most strongly influence the regression and the form of your estimator. The loss function choice determines the coefficients used in regression. Together these two pieces totally determine the form and accuracy of your estimator. Although this makes it sound like the two are totally different, the spirit of the two are identical.

Support Vector Regression - SVR

In a classification problem, the vectors $\vec{X}$ are used to define
a hyperplane that separates the two different classes in your solution.

These vectors are used to perform linear regression. The vectors closest to the test point are referred to as support vectors. We can evaluate our function anywhere so any vectors could be closest to our test evaluation location.

Gaussian processes are a particular form of SVM. The difference between the two lies in choice of kernel and loss function. The functional form of the kernel determines which vectors in your training set most strongly influence the regression and the form of your estimator. The loss function choice determines the coefficients used in regression. Together these two pieces totally determine the form and accuracy of your estimator. Although this makes it sound like the two are totally different, the spirit of the two are identical.

Kernel Principal Component Analysis

Kernel PCA

PCA Recap

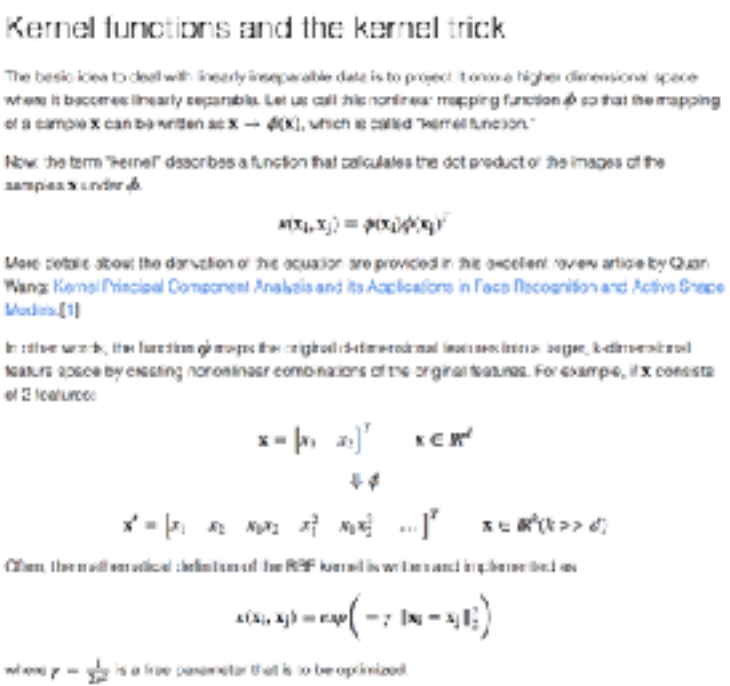
- PCA finds the directions of maximal variance in the data.
- The goal of PCA is to look to identify patterns in the data by finding the directions of maximal variance and high dimensional data and project it into a smaller dimensional subspace.

Kernel PCA

What is Kernel PCA and what does it do?

- Kernel PCA just performs PCA but in a new space.
- It uses Kernel trick to find principal components in a different space (higher dimensional space)
- PCA finds new directions based on covariance matrix of original variables. It can extract maximum P (number of features) eigen values.
- KPCA finds new directions based on kernel matrix. It can extract n (number of observations) eigenvalues.
- Kernel PCA does take more time compared to PCA

Kernel PCA



Kernel functions and the kernel trick

The basic idea to deal with linearly inseparable data is to project it onto a higher dimensional space where it becomes linearly separable. Let us call this nonlinear mapping function ϕ so that the mapping of a sample $\mathbf{x}$ can be written as $\mathbf{x} \rightarrow \phi(\mathbf{x})$, which is called "kernel function."

Now, the term "kernel" describes a function that calculates the dot product of the images of the samples $\mathbf{x}$ under ϕ.

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)^T$$

More details about the derivation of the equation are provided in the excellent review article by Quan Wang: Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models.[1]

In other words, the function ϕ maps the original d-dimensional features into a larger, k-dimensional feature space by creating nonlinear combinations of the original features. For example, if $\mathbf{x}$ consists of 2 features:

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \qquad \mathbf{x} \in \mathbb{R}^d$$

$$\Downarrow \phi$$

$$\mathbf{x}' = \begin{bmatrix} x_1 & x_2 & x_1 x_2 & x_1^2 & x_1 x_2^2 & \dots \end{bmatrix}^T \qquad \mathbf{x} \in \mathbb{R}^k (k >> d)$$

Often, the mathematical definition of the RBF kernel is written and implemented as

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \, \| \mathbf{x}_i - \mathbf{x}_j \|_2^2 \right)$$

where $\gamma = \frac{1}{2\sigma^2}$ is a free parameter that is to be optimized.

http://sebastianraschka.com/Articles/2014_kernel_pca.html#kernel-functions-and-the-kernel-trick

Kernel PCA

Pick a kernel

Construct the normalized kernel matrix of the data (dimension m x m):

$$\tilde{\mathrm{K}} = \mathrm{K} - 21_{1/n}\mathrm{K} + 1_{1/n}\mathrm{K}1_{1/n}$$

Solve an eigenvalue problem:

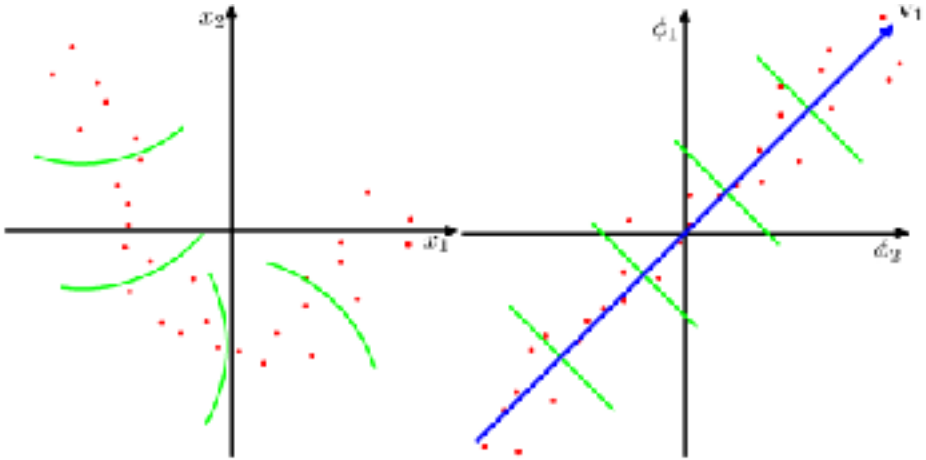
$$\tilde{\mathrm{K}}\alpha_i = \lambda_i\alpha_i$$

For any data point (new or old), we can represent it as

$$y_j = \sum_{i=1}^{n} \alpha_{ji} K(x, x_i), \quad \mathrm{j} = 1,..,d$$

http://www.cs.haifa.ac.il/~rita/uml_course/lectures/KPCA.pdf

Kernel PCA

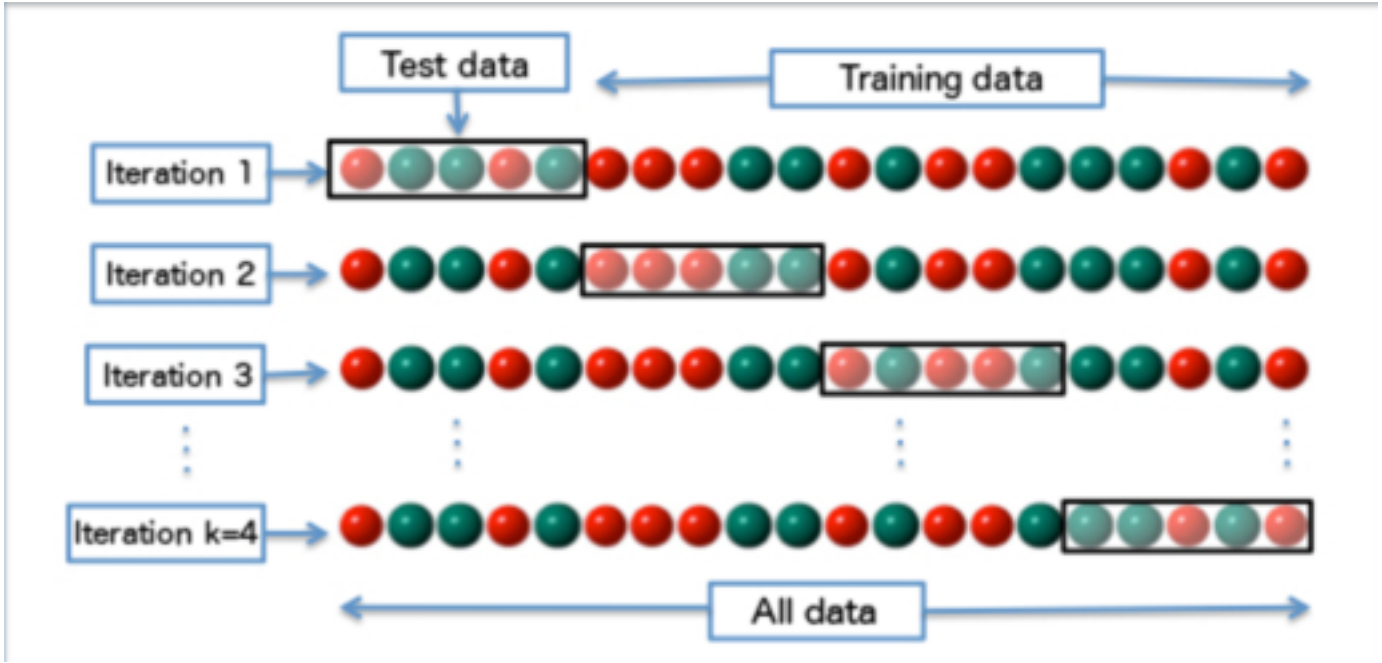


K-FOLD CROSS-VALIDATION

K-Fold Cross-Validation

- How to split training and testing data?
- Maximize the learning to obtain the best validation
- Cross-Validation
- K-Fold

K-Fold Cross-Validation



https://upload.wikimedia.org/wikipedia/commons/1/1c/K-fold_cross_validation_EN.jpg

K-Fold Cross-Validation

- The advantage of this method over repeated sub-sampling is that all observations are used for both training and validation
- Each observation is used for validation exactly once
- 10-fold is commonly used

K-Fold Cross-Validation

- K = 2
- Two sets d0 and d1
- Train on d0 and validate on d1
- Train on d1 and validate on d0
- When K = n (number of observations), the k-fold cross validation is exactly the leave one out cross-validation.

https://en.wikipedia.org/wiki/Cross-validation_(statistics)

K-Fold Cross-Validation

- Randomly partitioned into K equal sized subsamples
- Of the K subsamples, a single subsample is retained as the validation data for testing
- The remaining K - 1 subsamples are used as training data
- Cross-validation is then repeated K times (folds)
- The K results from the folds are then averaged for a single estimation

Grid Search

Grid Search

Cross Validation:

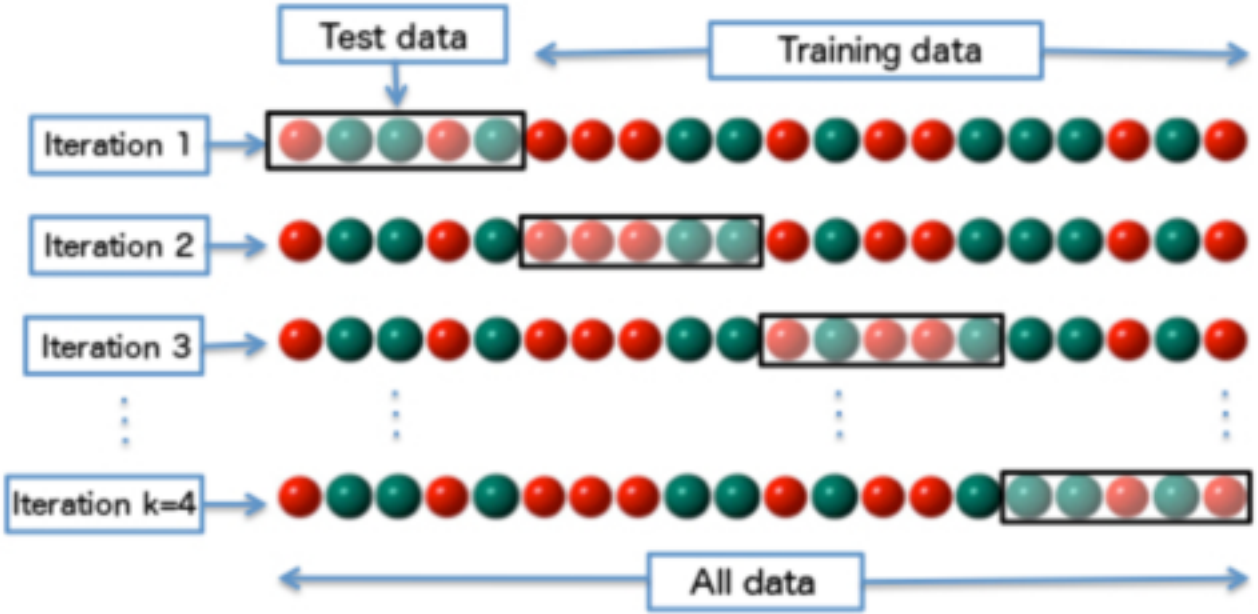
- Reserve part of your data to use in evaluating the model
- K-Fold
- Take 80% of your data and use it for training and then you take the remaining 20% of the data and use it to evaluate the performance of the model.

Grid Search

Cross Validation:

- Reserve part of your data to use in evaluating the model
- K-Fold
- Take 80% of your data and use it for training and then you take the remaining 20% of the data and use it to evaluate the performance of the model.

Grid Search



https://en.wikipedia.org/wiki/Cross-validation_(statistics)

Grid Search

- Grid search means that you have a set of models (that each have differing values and lie on a grid).
- You take each of the models to train and evaluate it using cross-validation.
- Then you select the one that performs the best

Grid Search

Example:

- You have a grid with the following values for (gamma, C): (1, 1), (0.1, 1), (1, 10), (0.1, 10).
- It's a grid because it's like a product of [1, 0.1] for gamma and [1, 10] for C.
- Grid-search would basically train for each of these four pair of (gamma, C) values.
- Then evaluate it using cross-validation, and select the one that did best.

https://stackoverflow.com/questions/19335165/cross-validation-and-grid-search

Grid Search

Recap:

- Applied across machine learning to calculate the best parameters to use for any given model.
- Can be extremely computationally expensive
- Grid Search will build a model on each parameter combination possible.
- It iterates through every parameter combination and stores a model for each combination.

XGBoost

XGBoost

XGBoost stands for extreme Gradient Boosting

- Awesome tool and algorithm used to push the limits of computation resources for boosted tree algorithms.
- Belongs to a broad collection of tools under Distributed Machine Learning Community (DLMC) and XGBoost was created by Tianqi Chen.
- Supports many languages
- Command Line Interface
- It can run on Hadoop

XGBoost

- Supervised Learning
- An ensemble method that seeks to create a strong classifier (model) based on “weak” classifiers.
- XGBoost implements this algorithm for decision tree boosting with an additional custom regularization term in the objective function.

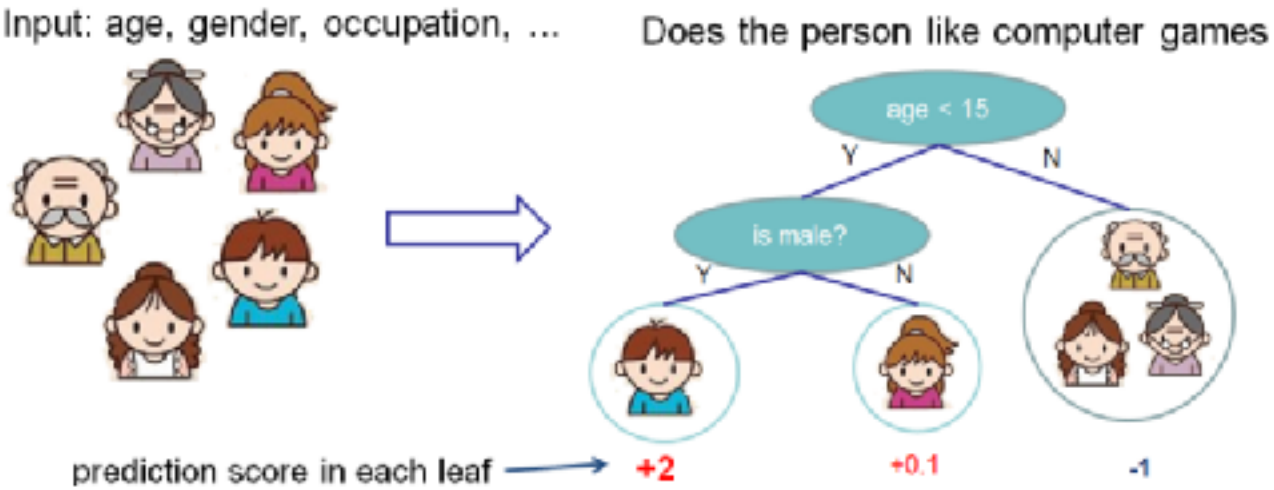
XGBoost

- Adds predictors and corrects previous models
- Fits the new model to new residuals of the previous prediction
- Regression and Classification

XGBoost

- Engineered to exploit every bit of memory and hardware resources for tree boosting algorithms.
- XGBoost offers several advanced features for model tuning, computing environments and algorithm enhancement.
- Capable of performing the three main forms of gradient boosting (Gradient Boosting (GB), Stochastic GB and Regularized GB) and it is robust enough to support fine tuning and addition of regularization parameters.

XGBoost

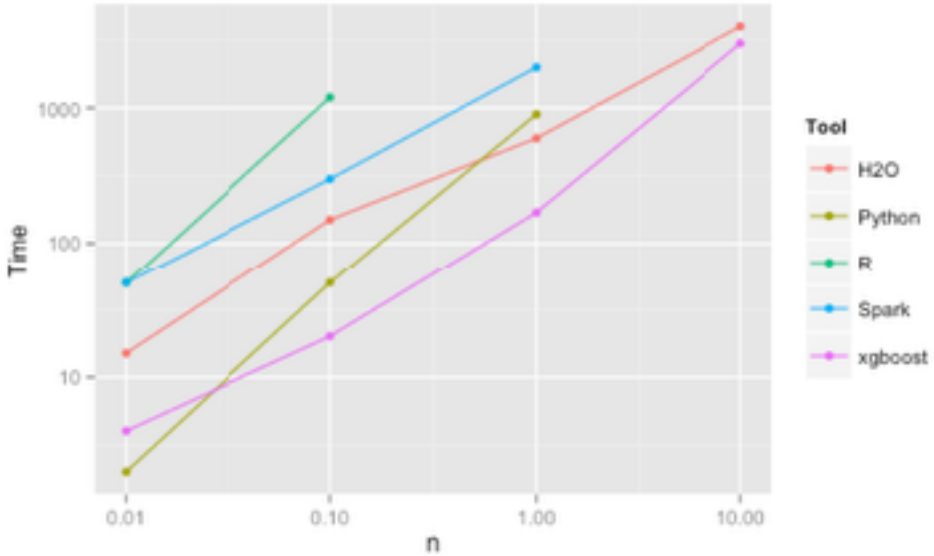


https://xgboost.readthedocs.io/en/latest/model.html

XGBoost

Why Choose To Use XGBoost?

Execution Speed
Model Performance



https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html