Mathematical techniques in data science

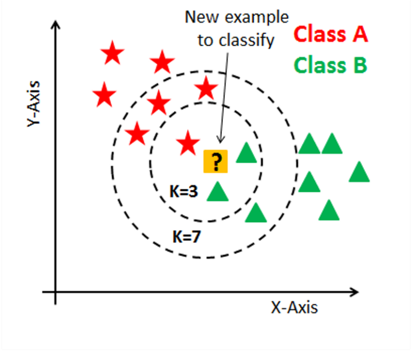
Lecture 5: Nearest neighbors (cont.)

Last lecture: Nearest neighbors

- Very simple idea: Make predictions based on labels of the nearest training examples
- Applicable to both classification and regression

K-nearest neighbor (K-NN) for classification

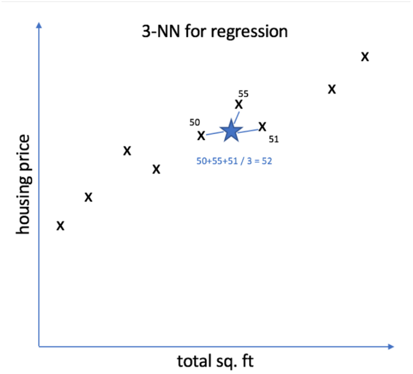
- Learning: Store all training examples
- Predict label of x:
  - Find the nearest K training examples to x
  - Assign the most frequent label to x



(Source: kdnuggets.com)

K-nearest neighbor (K-NN) for regression

- Learning: Store all training examples
- Predict label of x:
  - Find the nearest K training examples to x
  - Assign the average of the K nearest labels to x

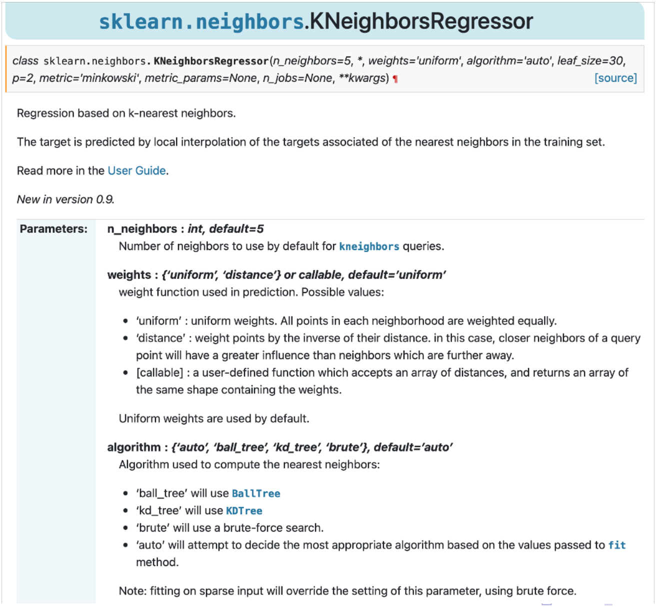


(Source: Jeremy Jordan)

Algorithmic details

- Number of neighbors
- How to define "near-ness"?
- How to find the nearest neighbors?
- Non-uniform weights

Nearest Neighbors on scikit-learn



sklearn.neighbors.KNeighborsRegressor

class `sklearn.neighbors.`**`KNeighborsRegressor`**(*n_neighbors=5*, ***, *weights='uniform'*, *algorithm='auto'*, *leaf_size=30*, *p=2*, *metric='minkowski'*, *metric_params=None*, *n_jobs=None*, ***kwargs*) ¶ [source]

Regression based on k-nearest neighbors.

The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set.

Read more in the User Guide.

New in version 0.9.

Parameters:

n_neighbors : *int, default=5*
Number of neighbors to use by default for `kneighbors` queries.

weights : *{'uniform', 'distance'} or callable, default='uniform'*
weight function used in prediction. Possible values:

- 'uniform' : uniform weights. All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

Uniform weights are used by default.

algorithm : *{'auto', 'ball_tree', 'kd_tree', 'brute'}, default='auto'*
Algorithm used to compute the nearest neighbors:

- 'ball_tree' will use `BallTree`
- 'kd_tree' will use `KDTree`
- 'brute' will use a brute-force search.
- 'auto' will attempt to decide the most appropriate algorithm based on the values passed to `fit` method.

Note: fitting on sparse input will override the setting of this parameter, using brute force.

Nearest Neighbors Demo

General steps to build ML models

- Get and pre-process data
- Visualize the data (optional)
- Create a model
- Train the model; i.e. call model.fit()
- Predict on test data
- Compute evaluation metrics (accuracy, mean squared error, etc.)
- Visualize the trained model (optional)

Review: Probability/Statistics

- Parameter estimation
- Bias-variance decomposition
- Overfitting and underfitting

Parameter estimation

- Model: A family of distributions/functions indexed by a vector of parameters θ
- Parameter estimation/tuning: given data $(Z_1, Z_2, \ldots, Z_n)$, find $\hat{\theta}$ that best "fits" (explain) the data

$$\begin{array}{ccccc} \text{parameter} & \Longrightarrow & \textit{sample} & \Longrightarrow & \textit{estimator} \\ \theta & \Longrightarrow & Z_1, Z_2, \ldots, Z_n & \Longrightarrow & \hat{\theta} \end{array}$$

Estimate vs estimator

$$\begin{aligned} &\textit{sample} &&\Longrightarrow \textit{estimator} \\ &Z_1, Z_2, \ldots, Z_n &&\Longrightarrow \hat{\theta} \end{aligned}$$

$$\begin{aligned} &\textit{observed data} &&\Longrightarrow \textit{estimate} \\ &z_1, z_2, \ldots, z_n &&\Longrightarrow \hat{\theta} \end{aligned}$$

Mean Squared Error

- Measuring error of estimation

$$|\hat{\theta} - \theta| \qquad \text{or} \qquad (\hat{\theta} - \theta)^2$$

- The error of estimation is random

Definition

The mean squared error of an estimator $\hat{\theta}$ is

$$E[(\hat{\theta} - \theta)^2]$$

Bias-variance decomposition

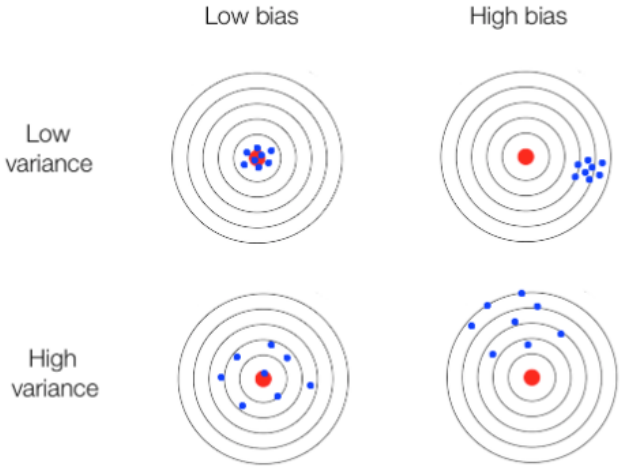
Theorem

$$MSE(\hat{\theta}) = E[(\hat{\theta} - \theta)^2] = V(\hat{\theta}) + \left(E(\hat{\theta}) - \theta\right)^2$$

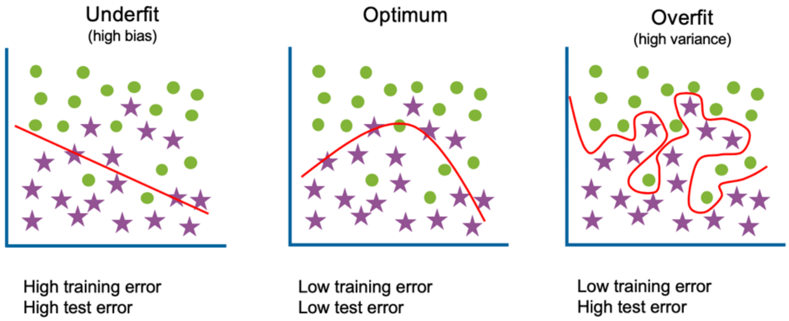
Bias-variance decomposition

Mean squared error $=$ variance of estimator $+$ $(bias)^2$

Bias-variance decomposition

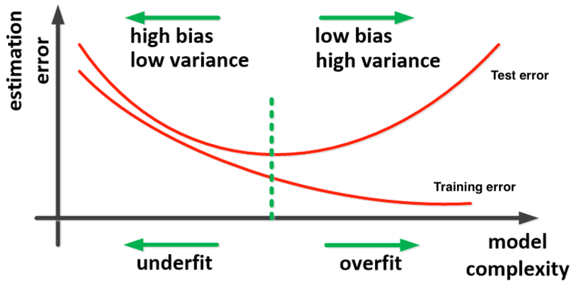


Underfiting/Overfitting

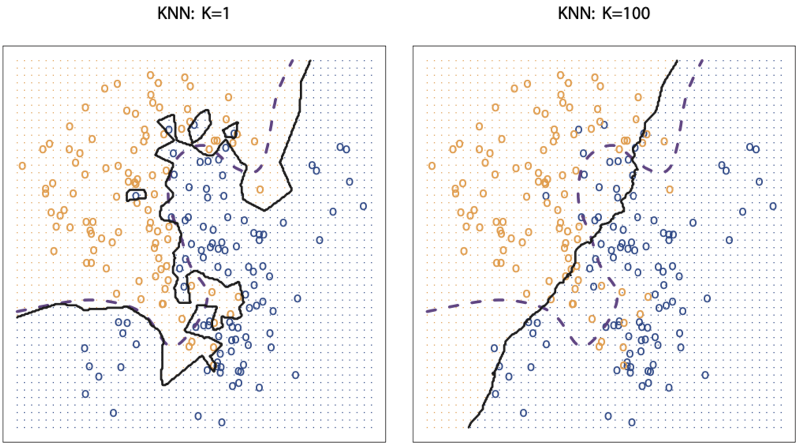


(Source: IBM)

Underfiting/Overfitting



Underfiting/Overfitting



Nearest neighbors: pros and cons

Pros:

- Simple algorithm
- Easy to implement, no training required
- Can learn complex target function

Cons:

- Prediction is slow
- Don't work well with high-dimensional inputs (e.g., more than 20 features)