Nearest Neighbor-III

M. R. Hasan
Readings

• Bishop: 2.5
• Murphy: 1.4.1, 1.4.2, 1.4.3
• Alpaydin: 8.1, 8.2, 8.3, 8.4
What We Will Cover

• Deciding Number of Nearest Neighbors in K-NN
• Model Selection
• Cross-Validation
• Hyperparameter Tuning
• Kernel Function
• K-NN: Non-parametric Classifier
• K-NN: Instance or Memory-based Learning
K-Nearest Neighbors Model

Practical Issues

- Distance metric (Euclidean, Manhattan, etc.)

- How many neighbors?

- Variance of the features are significantly different

- Non-zero covariance among some features

- Data is high-dimensional

- Complexity of K-NN algorithm & efficient solutions
K-Nearest Neighbors: Practical Considerations

• Distance Metric
• Number of Nearest Neighbors
• Weighing the Nearest Neighbors

These are hyperparameters of the K-NN model.

While a model’s parameters are estimated from data automatically, its hyperparameters are set manually for estimating model parameters.
Practical Issues

- Distance metric (Euclidean, Manhattan, etc.)
- How many neighbors?
- Variance of the features are significantly different
- Non-zero covariance among some features
- Data is high-dimensional
- Complexity of K-NN algorithm & efficient solutions

K-Nearest Neighbors Model
How do we choose an Optimal Value for K?
K-Nearest Neighbors: Choosing K

• Can we use the entire data set as our neighbors (K = N)?

• **Not a good idea!**

• Why not?

Predict the class of test points \( a \) and \( b \) by setting \( K = \) all data points

Both the test points \( a \) and \( b \) are predicted to belong to the **same class (red circle)**!
K-Nearest Neighbors: Choosing K

• For a very large $K$, **all classifications will become the same**!

• Which is to simply assign each novel $x$ to the **most numerous class** in the train data.

For an *empirical understanding* of this issue, see my Github notebook:

K-Nearest Neighbors: Choosing K

• How should we choose the optimal value for K?

• Our choice should be based on the best generalization performance.
K-Nearest Neighbors: Choosing K

• For \( k = 1 \) (left plot), there are no misclassified training examples.

• However, the decision boundary is jagged and almost certainly “overtrained” to fit the training data.

• It means that its predictions probably won’t generalize well to new observations.
K-Nearest Neighbors: Choosing K

• For $k = 25$ (right plot), some training examples are misclassified.

• But the decision boundary is relatively smooth and seems more likely to produce reasonable predictions for new data.
K-Nearest Neighbors: Choosing K

• This suggests that there is an optimal intermediate setting of $K$ which gives the best generalization performance.

• In other words, by varying $K$ we can control the complexity of the model (simple vs. intricate decision boundary).
K-Nearest Neighbors: Choosing K

• The process of choosing the optimal parameter value for the model (e.g., K) is known as model selection.

Model Selection: From the 3 K-NN models (K = 1, 5, 25), we have to select the model that gives the best generalization performance.
K-NN: Model Selection
K-Nearest Neighbors: Choosing K

- We plot various choices of the nearest neighbors K (x-axis) and corresponding performance measures (y-axis).

From this graph we find the optimum no. of nearest neighbors.

How do we do Model Selection?
K-Nearest Neighbors: Choosing K

• Before performing model selection, we usually take the available data and **partition** it into:
  - A training set
  - A validation set (hold-out set)
  - A test set
• Typical ratio is: 80% - 10% - 10%

The **training & validation** set are used for model selection.
K-Nearest Neighbors: Choosing K

- Training set: used for two purposes:
  - To train a **range of models** (e.g., Logistic Regression, KNN)
  - To train a **given model** with a range of values for its complexity parameters (e.g., various K values in K-NN)
K-Nearest Neighbors: Choosing K

- Validation set:
- To select the optimum K

It also provides an unbiased evaluation of a model fit on the training dataset during training.
K-Nearest Neighbors: Choosing K

- Test set:
- Provides the gold standard used to evaluate the model.
- It is only used once a model is completely trained (using the train and validation sets).
- For example, we select the best model using the validation set and then test it using the test set.
K-Nearest Neighbors: Choosing K

• In many applications, however, the supply of data for training and testing will be **limited**.
• In order to build good models, we wish to use **as much of the available data as possible for training**.
• But it leaves small amount data for validation.
K-Nearest Neighbors: Choosing K

- However, if the validation set is small, it will give a relatively noisy estimate of predictive performance.

- One solution to this dilemma is to use cross-validation.
K-Nearest Neighbors: Choosing K

- We will use a technique called **S-fold cross-validation** (CV) [also known as **k-fold** CV].
- A **test set** should still be held out for **final evaluation**.
- But the **validation set** is no longer needed when doing CV.
K-Nearest Neighbors: Choosing K

- First we randomly divide the data into training and testing set (e.g., 80% & 20%).
- Then the **S-fold cross-validation** is performed on the **training set**.

![S-fold Cross Validation Diagram]
K-Nearest Neighbors: Choosing K

- In **S-fold cross-validation** we split the training set into S-folds (in the figure $S = 4$).
- We create a model (e.g., $K = 1$) and “train” ($S - 1$) folds.
- Then, we evaluate the model using the remaining fold (i.e., validation fold).
K-Nearest Neighbors: Choosing K

• For the model (e.g., K = 1), this procedure is then repeated for all S possible choices, indicated here by the red blocks.
• The performance scores from the S runs are then averaged.
K-Nearest Neighbors: Choosing K

- In the example below we used “accuracy” as the performance measure.
- But there are other measures as well (e.g., misclassification error, precision, recall, etc., discussed later).
- Here, the average accuracy provides the performance measure of the model (e.g., K = 1).
K-Nearest Neighbors: Choosing K

- Next we create another model (e.g., K = 3).
- Then, run cross-validation to obtain the **average performance measure** of this model.
K-Nearest Neighbors: Choosing K

• Finally, we **plot various choices** of the nearest neighbor K (x-axis) and corresponding performance measures (y-axis).

From this graph we find the **optimum no. of nearest neighbors**.

This process is known as **hyperparameter tuning**!
Choosing **Optimal K**: S-Fold Cross Validation
K-Nearest Neighbors: Choosing K

• In addition to use it for model selection (choosing the optimal hyperparameter values), we use cross-validation for evaluating a model’s performance.
K-Nearest Neighbors: Choosing K

- Cross-validation allows us to get:
  - An estimate of the performance of our model.
  - A measure of how precise this estimate is (i.e., its standard deviation).

We would not have this information if we just used one validation set.
K-Nearest Neighbors: Choosing K

• But cross-validation comes at the cost of training the model several times, so it is not always possible.
K-Nearest Neighbors Model

**Practical Issues**
- Distance metric (Euclidean, Manhattan, etc.)?
- How many neighbors?

**Model Selection:**
- Distance metric function
- Value of $k$

**Hyperparameter Tuning**
K-Nearest Neighbors: Effect of Varying K
K-Nearest Neighbors: Example

- Handwritten Digit Example.
- Consider **two classes of handwritten digits**, zeros and ones.
- Each digit contains $28 \times 28 = 784$ pixels.
- The train data consists of 300 zeros, and 300 ones.
- A subset are plotted in Figures a and b.
K-Nearest Neighbors: Example

• To test the performance of the nearest neighbor method (based on Euclidean distance, $K = 1$) we use an independent test set containing a further 600 digits.

• The nearest neighbor method, applied to this data, correctly predicts the class label of all 600 test points.

• The reason for the high success rate is that examples of zeros and ones are sufficiently different that they can be easily distinguished.
K-Nearest Neighbors: Example

- A **more difficult task** is to distinguish between ones and sevens.
- We repeat the above experiment, now using 300 training examples of ones, and 300 training examples of sevens.
K-Nearest Neighbors: Example

• Again, 600 new test examples (containing 300 ones and 300 sevens) were used to assess the performance.
• This time, **18 errors are found** using nearest neighbor classification (3% error rate).
K-Nearest Neighbors: Example

- The **18 test points** on which the nearest neighbor method makes errors are shown below.

- If we use **$K = 3$ nearest neighbors**, the classification error **reduces to 14**, which is a slight improvement.
K-Nearest Neighbors: Example

• Note that the best machine learning methods classify real world digits (over all 10 classes) to an error of less than 1 per cent.
• It is better than the performance of an “average” human.
• This simple example on K-NN suggests that varying K we can improve the performance.
K-Nearest Neighbors: Varying K

- Another example that shows what happens when we vary K using.
- Observe that K controls the degree of smoothing (complexity).
- Small K produces many small regions of each class, whereas large K leads to fewer larger regions.
K-Nearest Neighbors: Varying K

- An **interesting property** of the nearest-neighbor ($K = 1$) classifier is:

- In the limit $N \to \infty$, the error rate is never more than twice the minimum achievable error rate of an optimal classifier (Cover & Hart, 1967).

- It means that K-NN is **pretty good tool to have in your ML toolbox**!
K-Nearest Neighbors: Practical Considerations

• Distance Metric
• Number of Nearest Neighbors
• Weighing the Nearest Neighbors

These are **hyperparameters** of the K-NN model.

While a model’s parameters are estimated from data automatically, its hyperparameters are set **manually** for estimating model parameters.
Weighted/Kernel-NN
So far, we considered a **democratic** K-NN: each nearest neighbor’s vote have **equal weight**.

However, it might be useful to turn to “**oligarchy**” based K-NN!
Weighted/Kernel NN

- In the “democratic” K-NN model, for a given value of $K$, we considered all neighbors with the same importance or weight.
- However, not all nearest neighbors are the same (even within the same vicinity).
- Thus, it makes sense to put more weight on the closest neighbors.

This is done by using a weight function or kernel function.
Weighted/Kernel NN

• What is a kernel?

• A kernel is a **weighting function** used in non-parametric estimation techniques.

• It **measures the “distance”** between the sample points by giving **more weight** to the “nearest” points.
Weighted/Kernel NN

- Note that there are different kernel based methods used in Machine Learning.
- For the purpose of this discussion, kernels are mostly used as a device for localization.
- Kernels are used in kernel density estimation to estimate random variables’ density functions.
Weighted/Kernel NN

• What is **good choice** for \( k(.) \)?
• We want it to **vary smoothly**.
• In other words, we want \( k(.) \) to **decay with the distance** \( |x - x_n| \).
• Possible choices: \( 1/d, 1/d^2 \), etc.

\[
k(\tilde{x}, \tilde{x}_n) = \frac{1}{d^2(\tilde{x}, \tilde{x}_n)}
\]
Weighted/Kernel NN

• One **mathematically convenient** choice is following.

\[
k(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}
\]

So, our **kernel function** becomes:

\[
k(x, \hat{x}_n) = \frac{1}{\sqrt{2\pi h^2}} e^{-\frac{d^2(x, \hat{x}_n)}{2h^2}}
\]

\[
k(\hat{x}, \hat{x}_n) = \frac{1}{\sqrt{2\pi h^2}} e^{-\frac{||x - \hat{x}_n||^2}{2h^2}}
\]

Here \( h \) is the **bandwidth** of the kernel function.
The intuition here is that by varying $h$ we can increase our “trust” circle!

- $h \to 0$: include just the closest one.
- $h \to \infty$: include all neighbors.

$$k(\hat{x}, \hat{x}_n) = \frac{1}{\sqrt{2\pi}h^2} e^{-\frac{d^2(\hat{x}, \hat{x}_n)}{2h^2}}$$

Here $h$ represents the standard deviation of the Gaussian components.

This is called the Gaussian kernel function.
Weighted/Kernel NN

- We can show that K-NN is a **special case** of a general approach called **Kernel Density Estimation**.
- We will not discuss Kernel Density Estimation.

The kernel based technique allows us to present a **probabilistic framework to reformulate** the nearest neighbor model.
K-NN: Non-Parametric Model
Nearest Neighbor Model

• The nearest neighbor is a non-parametric model.

• Let’s explain this by comparing it with the parametric model (e.g., Linear Regression).

• We will use an example of a regression problem in which the goal is to predict real number labels of data (e.g., house price based on number of bedrooms, yard size, etc.).
Parametric model:
Assume a form of the mapping function: \( y = f(x) \)

Linear Regression: \( y = w^*x \)

2\textsuperscript{nd} degree Polynomial Regression: \( y = w_1^*x + w_2^*x^2 \)

10\textsuperscript{th} degree Polynomial Regression: \( y = w_1^*x + w_2^*x^2 + \ldots + w_{10}^*x^{10} \)

The capacity (size of hypothesis class) grows with the size of training data.
KNN: Non-parametric model

The K-NN does not assume anything about the form of the mapping function other than patterns that are close are likely to have a similar output variable.

Doesn’t impose a hypothesis on the mapping, let the data speak!
Machine Learning: Learning Functions

- Machine learning can be summarized as **learning a function** \( f \) that maps input variables \( (X) \) to output variables \( (Y) \).
- \( Y = f(x) \)
- An algorithm **learns this target mapping function** from training data.
- The **form** of the function is **unknown**.
- We **evaluate different machine learning algorithms** and see which is better at approximating the underlying function.
Parametric Model

- A parametric model **simplifies** the function to a **known form** (e.g., linear, polynomial, etc.).
- It **summarizes data with a set of parameters of fixed size** (independent of the number of training examples).
- For example, a binary classification problem is solved by predicting the binary distribution (binomial) via its mean and variance.

Linear Regression: $y = w \times x$

2$^{\text{nd}}$ degree Polynomial Regression: $y = w_1 \times x + w_2 \times x^2$
Parametric Model

• No matter **how much data** we throw at a parametric model, it *won’t change its mind* about how many parameters it needs.

• Examples: Linear Regression, Logistic Regression, Naïve Bayes, Linear Support Vector Machine.

**Linear Regression**: $y = w \times x$

**2\textsuperscript{nd} degree Polynomial Regression**: $y = w_1 \times x + w_2 \times x^2$
Parametric Model

- In parametric models we always have some hypotheses about the data.
- For example, in a regression problem we assume a linear function or a lower degree polynomial.
- When dataset is small, for avoiding overfitting it makes sense to have a strong restriction on the allowable hypotheses.

Linear Regression: \( y = w \times x \)

2nd degree Polynomial Regression: \( y = w_1 \times x + w_2 \times x^2 \)
Parametric Model

- But when there are thousands or millions or billions of examples to learn from, it seems like a better idea to let the data speak for themselves rather than forcing them to speak through a tiny vector of parameters.
- If the data say that the correct answer is a very wiggly function, we shouldn’t restrict ourselves to linear or slightly wiggly functions.

In a non-parametric model, we don’t impose a hypothesis on the mapping!
Non-Parametric Model

• Non-parametric models do not make strong assumptions about the form of the mapping function.

• By not making assumptions, they are free to learn any functional form from the training data.

• Example: K-NN, Decision Tree, RBF Kernelized Support Vector Machine.
Non-Parametric Model

• Non-parametric models are good when we have a lot of data and no prior knowledge.

• It is preferable when we don’t want to worry too much about choosing just the right features.
Nearest Neighbor: Non-Parametric Model

• The nearest neighbor is a non-parametric model.

• The K-NN does not assume anything about the form of the mapping function other than patterns that are close are likely have a similar output variable.
Nearest Neighbor: Non-Parametric Model

• How do we decide whether to use Parametric or Non-Parametric Models?

• We need to see whether the capacity (size of hypothesis class) grows with size of training data?
  - Yes = Non-Parametric Models
  - No = Parametric Models

But non-parametric models or memory-based learning suffers from some limitations!