# COMPSCI 389
# Introduction to Machine Learning

**Days:** Tu/Th. **Time:** 2:30 – 3:45 **Building:** Morrill 2 **Room:** 222

**Topic 5.5: Validation Sets (and Review of NN Variants and Model Evaluation)**

Prof. Philip S. Thomas (pthomas@cs.umass.edu)

# Nearest Neighbors (NN)

- Given a query, find the label associated with the closest point in the training data.
- "Closest" is determined using Euclidean distance:

$$\text{dist}(x, x') = \sqrt{\sum_{i=1}^{n} \left(x_i - x_i'\right)^2}$$

- If multiple points equally near, break ties arbitrarily.
  - For example, return the label of just one of the nearest neighbors.

# K-Nearest Neighbors (k-NN)

- NN was unreasonable when many points are equally close but have different labels.
- NN is also unreasonable when many points are close to the query (but not precisely equal in distance).
- k-NN improves upon NN by returning the average of the labels of the k nearest neighbors.
  - Data structures like KD-Trees and Ball-Trees can be used to efficiently find the k nearest neighbors to a query point.

# Weighted k-Nearest Neighbors (Weighted k-NN)

- The k-NN algorithm does not distinguish between the cases:
  - All k neighbors are roughly the same distance from the query.
  - Some of the k neighbors are much closer than others.
- Instead, it gives the same weight to all k neighbors.
- Weighted k-NN weights each of the k points based on their distance:

$$\hat{y} = \sum_{i=1}^{k} \frac{w_i}{\sum_{j=1}^{k} w_j} y_i^{NN}$$

$i^{\text{th}}$ nearest neighbor's label

# Weighted k-NN (cont.)

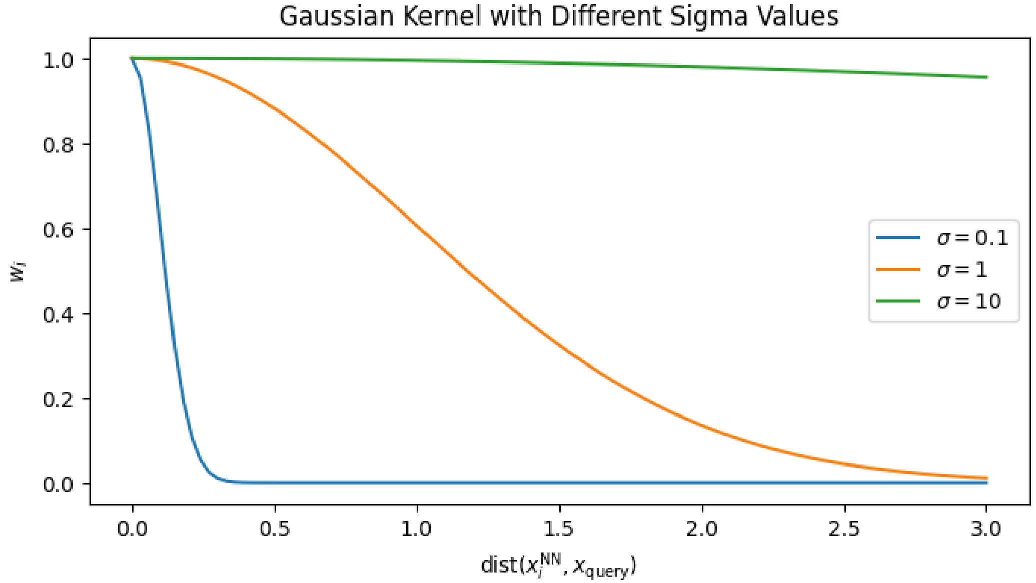
- Many choices of weighting functions.
- One common choice is the Gaussian kernel:

$i^{\text{th}}$ nearest neighbor's features

Query features

$$w_i = e^{-\frac{\text{dist}\left(x_i^{NN}, x_{\text{query}}\right)^2}{2\sigma^2}}$$

- Sigma scales how quickly weights decrease with distance.



# Tuning Hyperparameters

- **Hyperparameter**: a variable, like *k*, that changes the behavior of the algorithm, and which is often set by the data scientist applying the algorithm.
- **Grid Search**: Specify possible values for each hyperparameter (often equally spaced), train models using all possible combinations of hyperparameter settings, and select the ones that result in the best fit.

# Classification with NN-Variants

- NN: No changes needed!
- k-NN: The predicted label comes from a majority vote of the k nearest neighbors.
- Weighted k-NN: Each neighbor's vote is weighted in the vote.

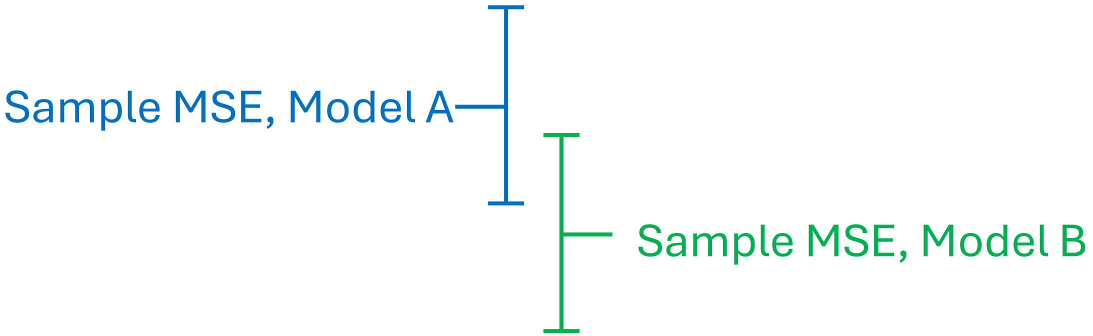
- **Note**: We will focus on regression for a while, and then return to classification after a few lectures.

# Confidence Intervals

- We shouldn't always trust the sample performance metrics.
  - Sample MSE: $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
    - This is a statistic or sample statistic.
  - It could be quite different from the true MSE: $\mathbf{E}\left[\left(Y - \hat{Y}\right)^2\right]$.
    - This is a parameter or population statistic.
- We can compute *confidence intervals* for sample statistics.
  - If the sample statistic is an average (of normally distributed values) then $\pm\, 1.96 \times SE$ is a 95% confidence interval.
    - SE is the *standard error*: $SE = \frac{\sigma}{\sqrt{n}}$, where $\sigma$ is the sample standard deviation with Bessel's correction.
- We often report performance metrics with $\pm X$, where X is standard error, 1.96 times standard error, a 95% or 90% confidence interval, or standard deviation.

# Model Evaluation

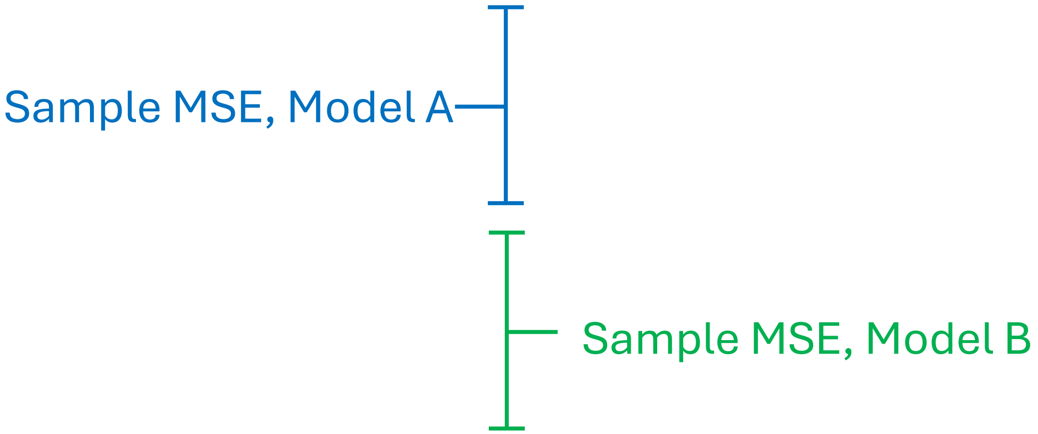
- We can evaluate models trained on the training data by:
  - Compute the sample MSE (or metric of interest) on the test set.
  - Compute a confidence interval (or related quantity) for the sample MSE.
  - Check whether one model's high-confidence lower-bound is larger than another model's high-confidence upper bound.




**Cannot** conclude that Model A is better than Model B with sufficient confidence.

# Model Evaluation

- We can evaluate models trained on the training data by:
  - Compute the sample MSE (or metric of interest) on the test set.
  - Compute a confidence interval (or related quantity) for the sample MSE.
  - Check whether one model's high-confidence lower-bound is larger than another model's high-confidence upper bound.




**Can** conclude that Model A is better than Model B with sufficient confidence.

# Algorithm Evaluation

- Consider the following:
  - Train model A using one algorithm.
  - Train model B using another algorithm.
  - Evaluate models A and B using confidence intervals.
- This does **not** fully evaluate the two algorithms.
- It fails to capture how much the learned models vary with different training sets.

# K-Fold Cross-Validation

- Split data D into k equal-sized subsets (folds), F1, F2, ..., Fk
- For i from 1 to k:
  - Set aside fold Fi as the validation set, and combine the remaining k-1 folds to form a training set.
  - Train the model M on the k-1 training folds using the ML algorithm being evaluated.
  - Evaluate the performance of model M on the validation fold Fi. Store the performance metric Pi
- Calculate the average performance metric: mean(P1,P2, ..., Pk).
- Optionally, calculate other statistics (like standard error) of the performance metrics across the folds.

# Train/Validation/Test Sets (**New Material**!)

- Validation sets are often used to automatically tune hyperparameters.
- The data is split into three sets: train, evaluation, and test. The following procedure is then used:
  - For each hyperparameter setting:
    - Train a model using the training data.
    - Evaluate the model using the validation data.
  - Select the hyperparameter settings that achieve the best evaluation on the validation set.
  - Train a model using all the training and validation data and the hyperparameters that achieved the best evaluation.
  - Evaluate the model using the testing set.

# Nested Cross-Validation: Train/Test/Validation + k-Fold Cross-Validation

- Train/Test/Validation does not account for the variance that results from the selection of the training and validation sets.
  - It evaluates the performance of the one model learned from a specific pair of training and validation sets.
- The use of train/validation/test sets can be combined with k-fold cross-validation to account for this additional variance.
  - This method is called **nested cross-validation**.
  - While principled, this method is computationally intensive.
  - For this introductory course you should understand the general idea behind nested cross-validation, but need not study the algorithmic details.

# End