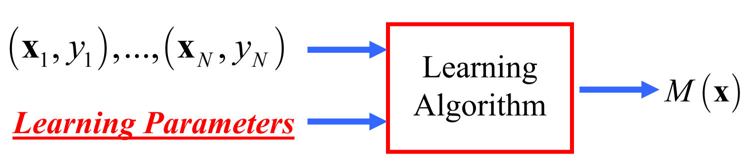
Model Selection: The Frequentist Approach

Greg Grudic

Building Supervised Learning Models



Model is used to make predictions! $\hat{y} = M(\mathbf{x})$

Learning Parameters

- These dictate how the learning algorithm will build a model
- Changing the learning parameters changes how good the model is
- <u>Goal:</u> Choose the learning parameters that produce the best model

Learning Parameters

- What are the <u>learning parameters</u> for the linear perceptron algorithm?
  - Learning rate.
- What are the <u>learning parameters</u> for the non-linear perceptron algorithm?
  - Learning rate and choice pf nonlinear basis functions.
- What are the <u>learning parameters</u> for linear C – SVM classification?
  - C
- What are the <u>learning parameters</u> for nonlinear C – SVM classification?
  - C, kernel choice, kernel parameter values

Which Model Is Best?

- OR – which learning parameters should I use?
- The ones that produce a model that gives the best accuracy results on data that was NOT used to build the model (sometimes called future data or **test data**).
- Test data does not appear in the learning set!
- So how do I pick the pick model parameters if I don't know what the test data is?
- Answer: create a *fake test set* called a validation set.

Measuring Model Accuracy: Regression

- Assume a set of data $(\mathbf{x}_1, y_1),\ldots,(\mathbf{x}_K, y_K)$
- Regression accuracy
  - Two commonly used metrics
    - Mean Square Error

$$error_{M(\mathbf{x})} = \frac{1}{K}\sum_{i=1}^{K}\left(y_i - M(\mathbf{x}_i)\right)^2 = \frac{1}{K}\sum_{i=1}^{K}\left(y_i - \hat{y}_i\right)^2$$

    - Relative Error

$$error_{M(\mathbf{x})} = \frac{\sum_{i=1}^{K}\left(y_i - M(\mathbf{x}_i)\right)^2}{\sum_{i=1}^{K}\left(y_i - \bar{y}\right)^2}$$

Measuring Model Accuracy: Classification

- Assume a set of data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_K, y_K)$
- Classification accuracy

$$error_{M(\mathbf{x})} = \frac{1}{K}\sum_{i=1}^{K} c\left(\mathbf{x}_i, y_i, M(\mathbf{x}_i)\right)$$

Where $c\left(\mathbf{x}_i, y_i, M(\mathbf{x}_i)\right) = \begin{cases} 0 & \text{if } y_i = M(\mathbf{x}_i) \\ 1 & \text{otherwise} \end{cases}$

Picking the Best Learning Parameters

- Partition learning data into disjoint sets
  - Training Set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$
    - Used to build the model
  - Validation Set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_V, y_V)$
    - Used to evaluate model
- Pick the Learning Parameters that give the lowest error on the Validation Set

$$error_{M(\mathbf{x})} = \frac{1}{V}\sum_{i=1}^{V} c\left(\mathbf{x}_i, y_i, M(\mathbf{x}_i)\right)$$

How Big Should the Training and Validation Sets Be?

- It Depends…
- If you have Lots of data for learning
  - Randomly putting half the data into each set is often sufficient
- If you only have a Small data set for learning
  - Usually do N-Fold Cross Validation

N-Fold Cross-Validation

- Partition the data $D_0 = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_M, y_M)\}$ into N disjoint sets $T_1, \ldots, T_N$
- For i from 1 to N, do
  - Use T_i for validation and the remaining S_i for training
    - Training Set: $S_i = \{D_0 - T_i\}$
    - Error on validation T_i : $error_{T_i}$
- Return the average error on validation sets

$$error_{M(\mathbf{x})} = \frac{1}{N} \sum_{i=1}^{N} error_{T_i}$$

Pick the learning parameters that minimize this error!

Does My Cross Validation Error Reflect the True Error of My Model?

- ***No***!!!!!!!!!!!!!!!!!!
- More on predicting the error rate of a learning algorithm on a specific set of data later….

Next Class

- Dimensionality Reduction
  - Linear PCA and Kernel PCA
- Project
  - I will ask each of you to briefly discuss (2 to 5 min) what kind of project might interest you