Machine Learning
Baruch College
Lecture 3

Miguel A. Castro

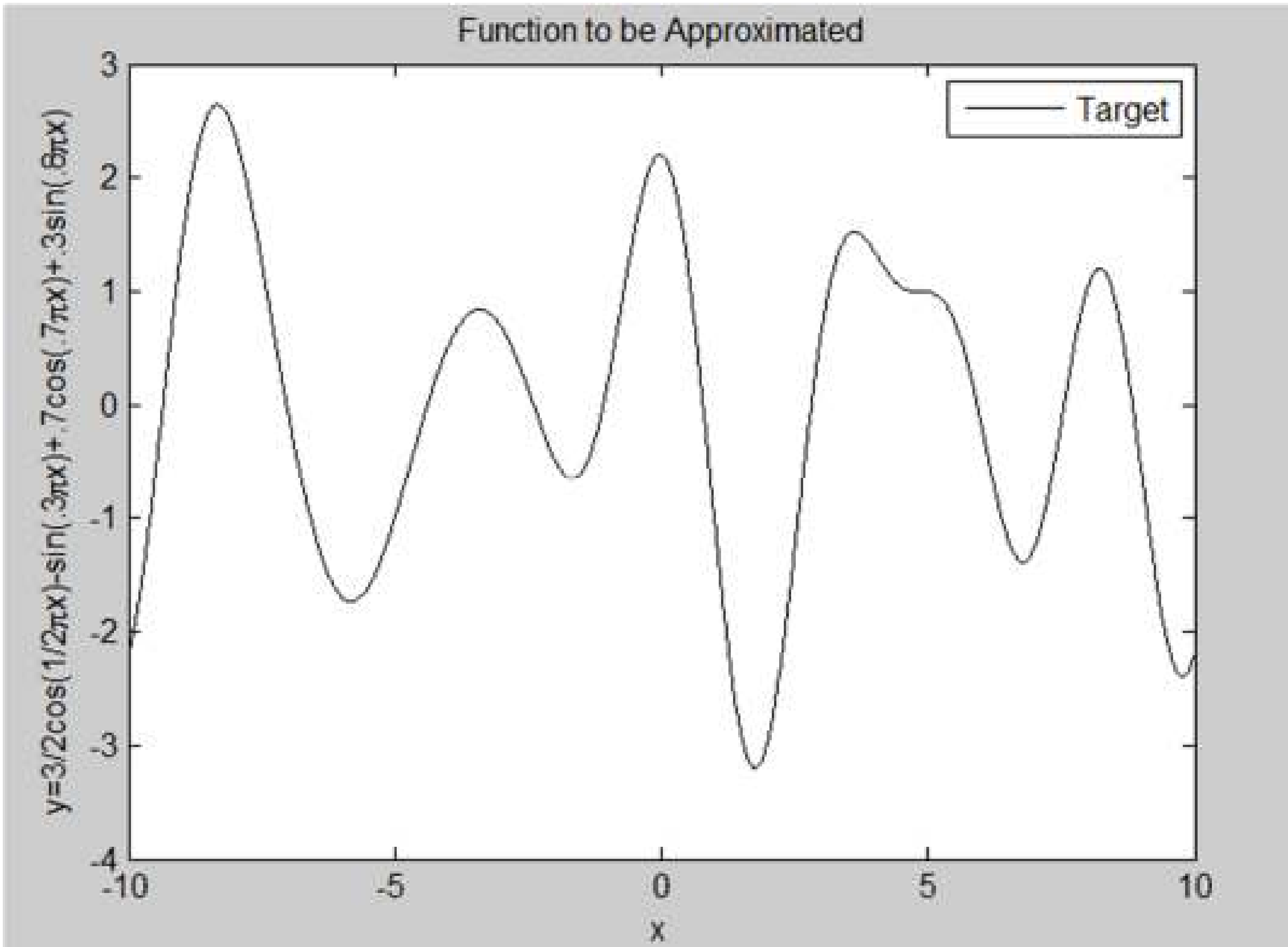
Today We'll Cover:

- Class Schedule: Propose to move Exam to October 6
- Review of Last Lecture
- Continue ROC Curves
- Association Rules

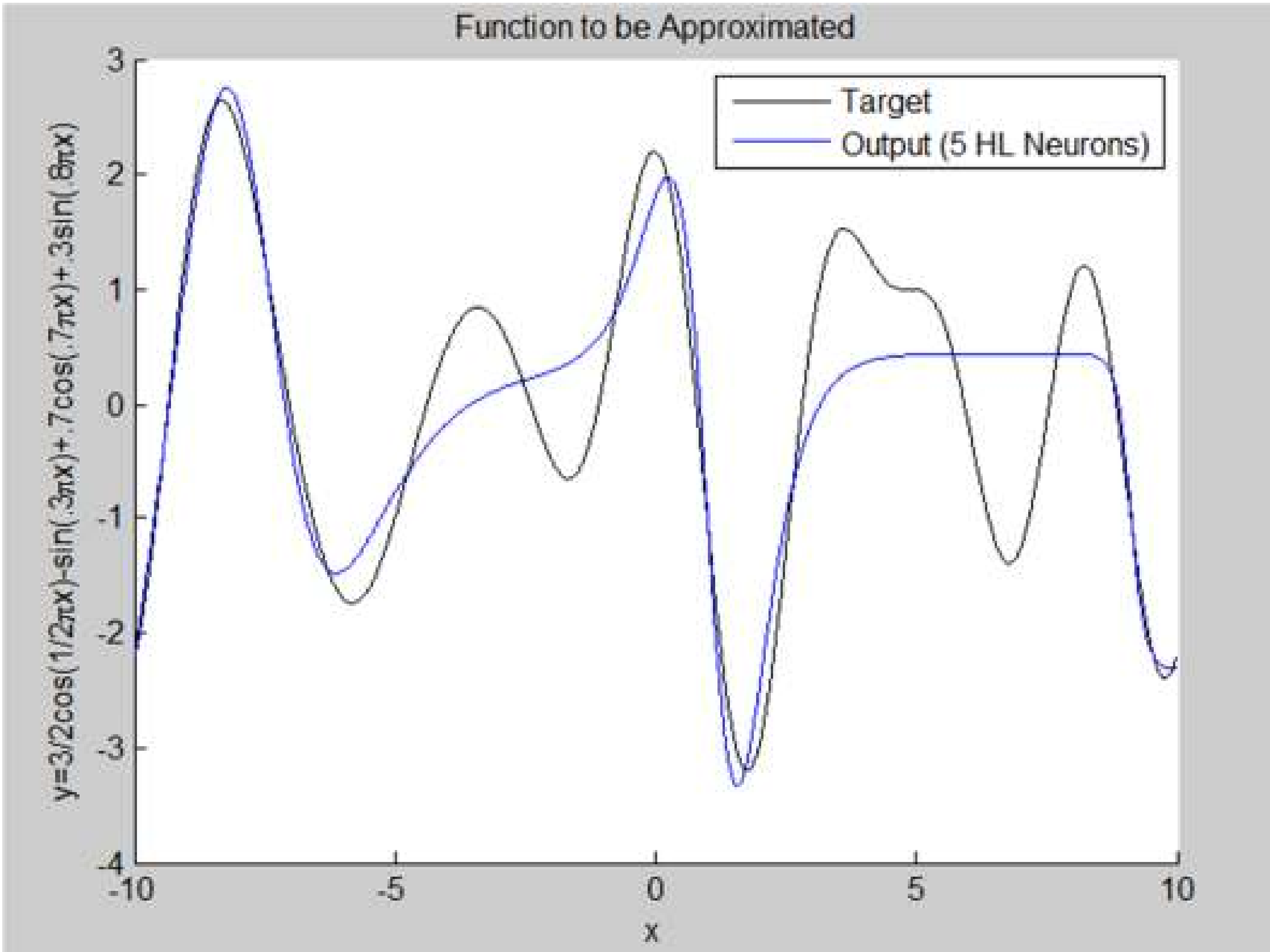
Last Time…

- <u>Multi-layer</u>, Feedforward Neural Networks with <u>*Non-linear Activations*</u> are Universal Approximators
  - Able to discover input/output mappings arbitrarily well
  - Example: Classification
  - Example: Function Approximation
  - Classification is, in a way, a subset of Function Approximation, but treated and evaluated differently.
- In order for "learning" to be automated we require the Non-linear Activations to be <u>*Differentiable*</u> → Automated Learning Rules such as <u>*Backpropagation*</u>.

Function Approximation (Review)

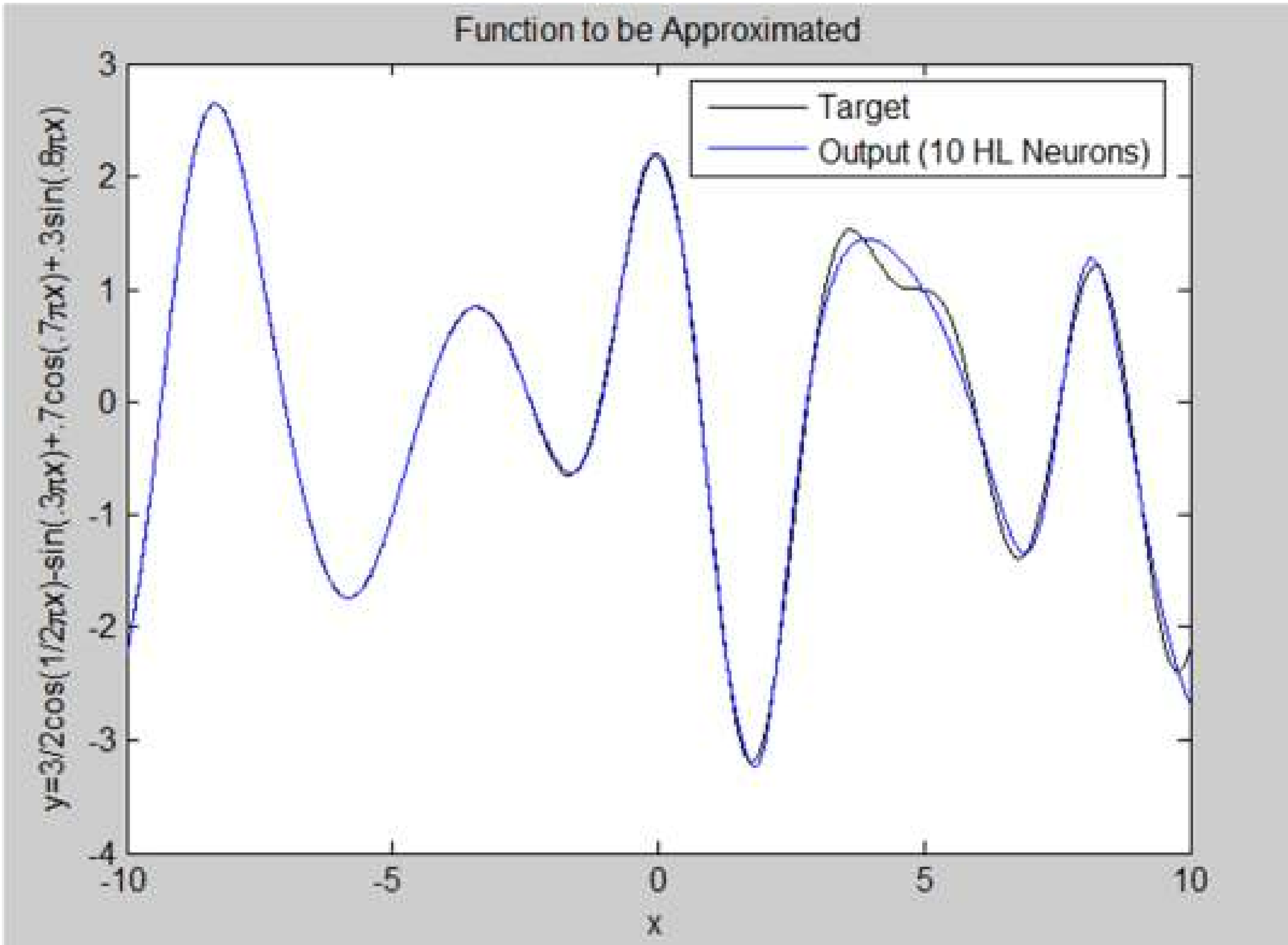


Function Approximation, FNN, 5 HN (Review)



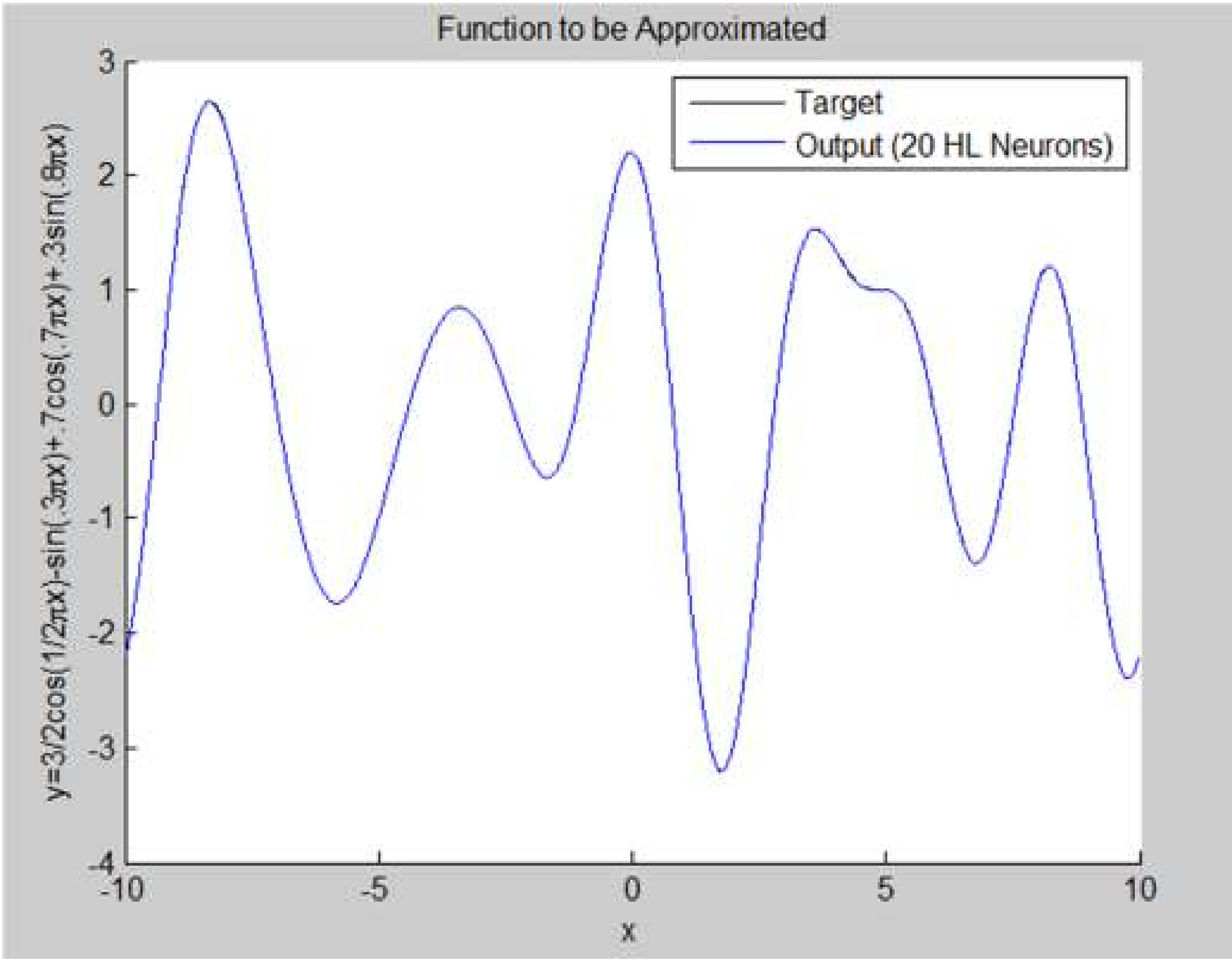
- Pretty Good

Function Approximation, FNN, 10 HN (Review)



- Better

Function Approximation, FNN, 20 HN (Review)



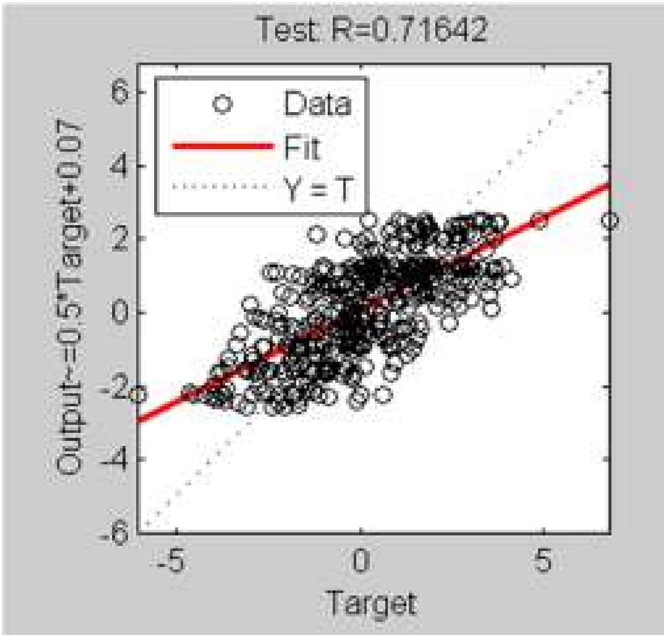
- Spot on!

Last Time…

- Neural Networks can be thought of or treated as being *Non-parametric* to the extent that there is no model specification (“black box”).
- They are *Parametric* in the sense that they have internal *Parameters* (connection strengths or “weights”).
  - They suffer from the usual need for model *Parsimony* to avoid…
- The Problem of Noise:
  - Overfitting vs. Generalization
  - **Generalization**: the ability to capture input/output relations that *persist* when presented with new data.
  - **Overfitting**: capturing random or noisy input/output relations that *will not persist* when presented with new data because the captured patterns arise from noise, and are thus not predictable.
- All else being equal, Generalization is helped by
  - More data.
  - Fewer parameters (parsimony).

Last Time…

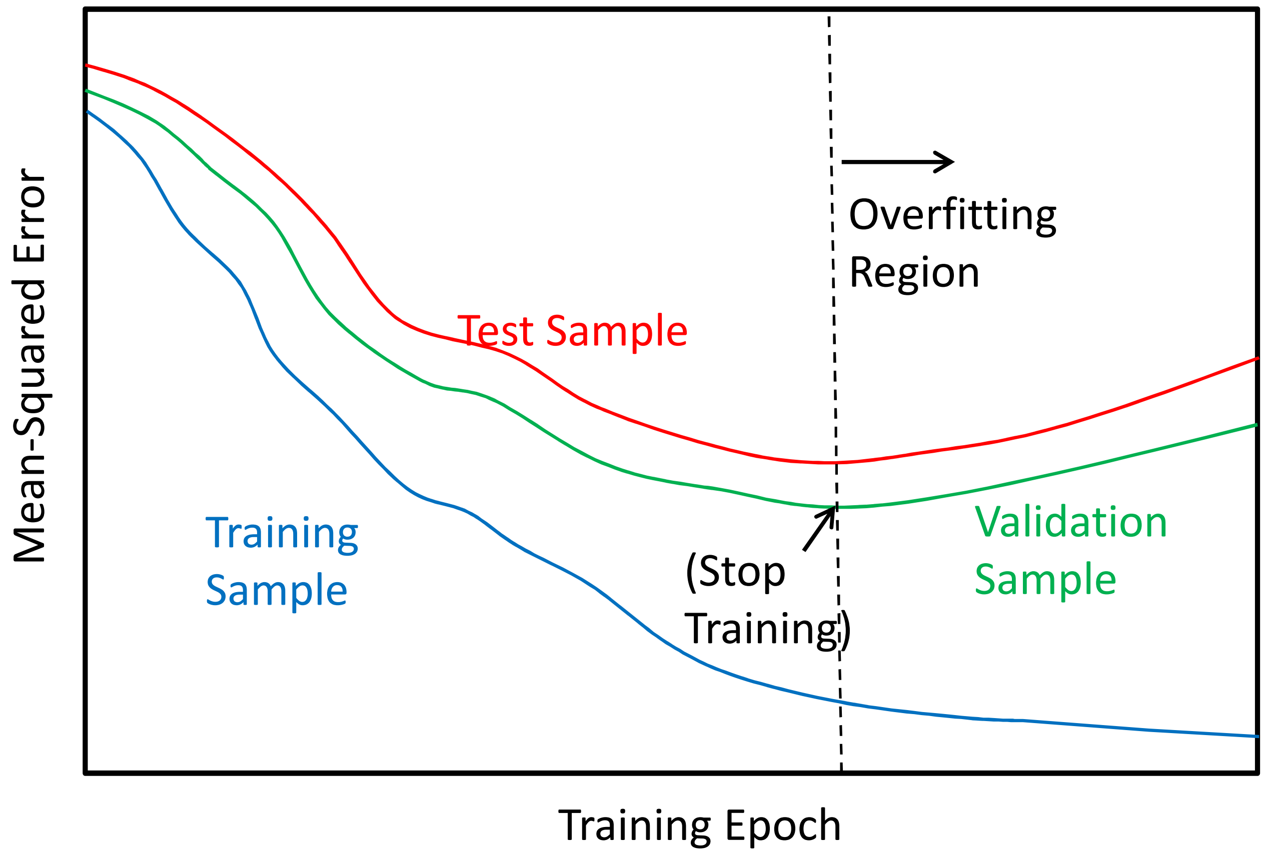
- One way to test Function Approximation *<u>Performance</u>* is to look at the regression of Actual Output vs. Target (on the *Test* set):



- We can then look at performance stats such as R^2.

Last Time…

- Techniques to avoid overfitting:
  - Early Stopping or Cross Validation: Split your data set into Training, Validation, and Test (or Hold Out) sets:
    - Train on the Training Set.
    - Stop when Validation Error flattens out and starts to increase.
    - Test your model's predictive power using the Test (Hold Out) set.



Last Time…

- Techniques to avoid overfitting (continued):
  - Regularization:
    - Pruning by zeroing out small weights (If $|w_{ij}| < \delta$ then set $w_{ij} \triangleq 0$;).
    - Pruning by zeroing out weights that have little impact on the output (If $\left|\frac{\partial output}{\partial w_{ij}}\right| < \delta$ then set $w_{ij} \triangleq 0$;).
    - Test your model's predictive power using the Test (Hold Out) set.
  - Pruning methods work well but rely on "small" parameter δ and are computationally expensive.
  - Regularization by adding a term in the Penalty (Error) function which effectively penalizes large weights:

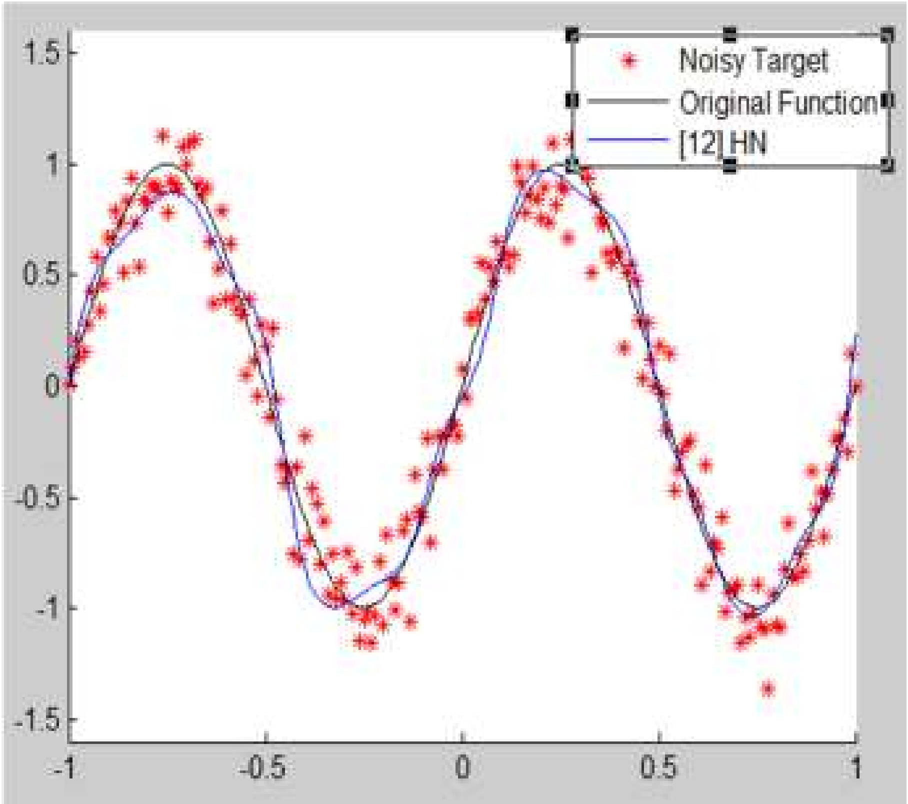
$$\mathcal{F}_W = (1-\lambda)\frac{1}{N}\sum_i \tfrac{1}{2}(t_i - a_i)^2 + \lambda \sum_{ij} \tfrac{1}{2}{w_{ij}}^2 .$$

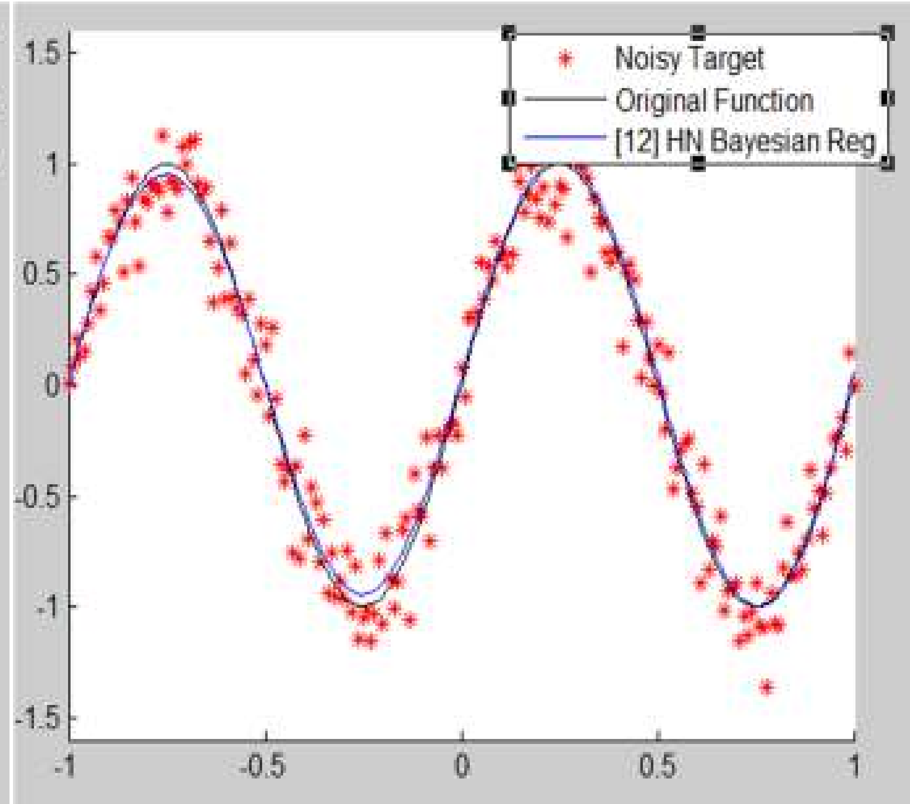
Last Time…

- Techniques to avoid overfitting (continued):
  - Annealing Methods:
    - Add noise to the learning process to prevent it from getting stuck in local minima and slowly reduce the noise to eventually settle on the global minimum.
    - Works, but it’s computationally expensive and relies on *ad hoc* parameters like the “temperature” (similar to GA).
  - Bayesian Regularization:
    - Introduces objective criteria for regularizing parameters;
    - Introduces objective criteria for comparing among models (including non-neural network approaches);
    - Does not decrease the data size;
    - Bonus: Obtains the *Effective Number of Degrees of Freedom* (weights).
    - This can be used to re-train a new network with a smaller architecture that matches the Effective Number of Weights found above, or to check if a larger network leads to the same Effective Number of Weights, meaning it’s unnecessary to go larger.

Last Time…

- Bayesian Regularization produces smoother fit:





- Number of Effective Parameters went from 37 to 9.
- R^2s comparable (around the max) but a bit higher for BR.
- But fit is smoother with Bayesian Regularization (better Generalization).

Last Time…

- Strictly speaking, only a single hidden layer is needed for function approximation capability.
- However, we saw in the last lecture some examples where networks with similar numbers of degrees of freedom (weights) but with more hidden layers appeared to generalize better.
- This does not constitute proof, but is a potentially interesting finding
- A more exhaustive empirical study (perhaps a Monte Carlo simulation under various conditions) may offer support for this hypothesis.

Aside:

- Suppose we are confronted with a noisy signal (as in HFT).
- We build a reasonable (parsimonious and regularized) model.
- We extract a signal (a prediction) using this model.
- We measure the model's R^2 (using the Test set).
- If we assume that our model
  - 1. Has not been overfitted, and
  - 2. Has captured the predictive part of the signal reasonably well (*i.e.*, its R^2 is close to the optimal one), then:
  - We can make inferences about the signal-to-noise ratio $\left({}^{1}/_{n}\right)$ and hence about the model's risk. For example,

$$R^2 \simeq maxR^2 = 1 - \frac{n^2}{1+n^2} \;\; \Rightarrow \;\; n^2 \simeq \frac{1-R^2}{R^2}.$$

  - If we assume that the model's expected return is proportional to the (unobserved) target signal's volatility:

$$r \simeq \lambda \sigma_Y,$$

  - Then we can see that the *Sharpe Ratio* of a (single-security) strategy employing our model is:

$$SR \simeq \frac{\lambda}{\sqrt{1+n^2}} \simeq \lambda R.$$

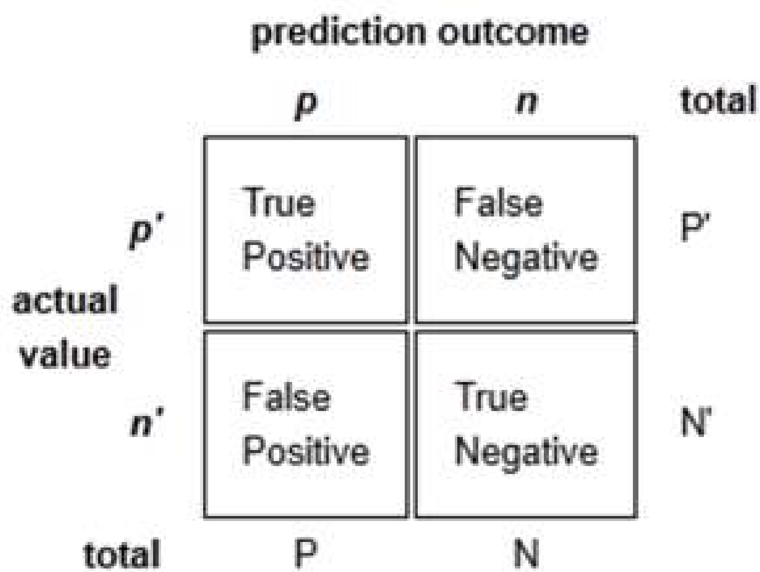
- An interesting question would be to extend this analysis to two or more securities.

Last Time…

- Two-class classifiers (*e.g.* 0/1 or True/False) can make Two Types of Mistakes:
  - Type I Error: Assumes False when True;
  - Type II Error: Assumes True when False.
- Usually Type I Errors come at the expense of Type II Errors and vice-versa.
- Tools for evaluating Classifier Performance:
  - Confusion Matrix
  - ROC Curve

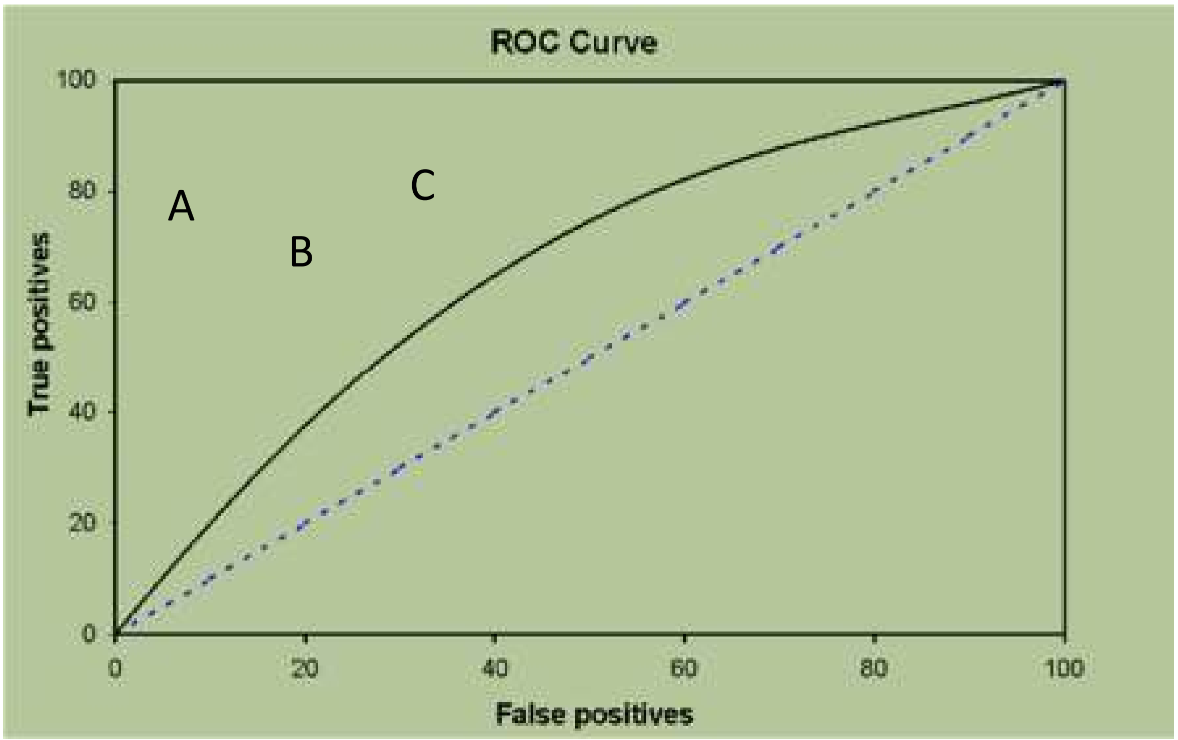
Last Time…

- *Confusion Matrix* Quantifies predicted vs. actual classes.
- Quantifies misclassification error rates and correct classification rates.



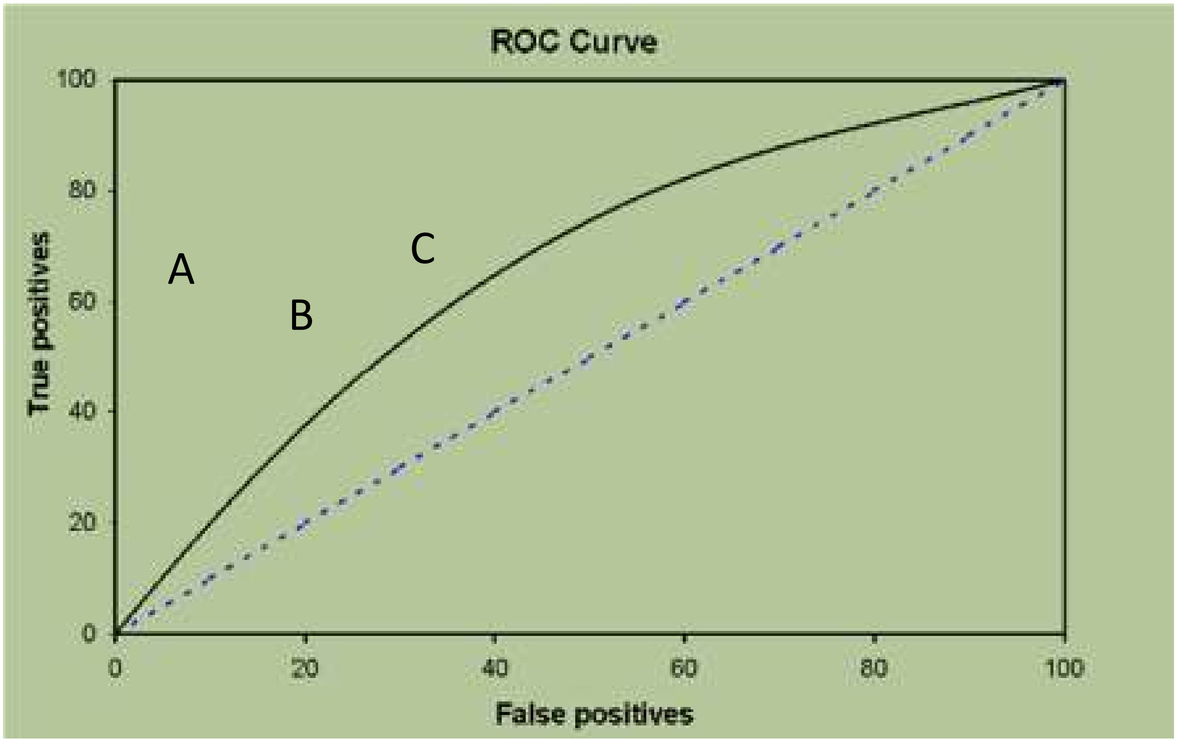
Last Time…

- *ROC Curves*:



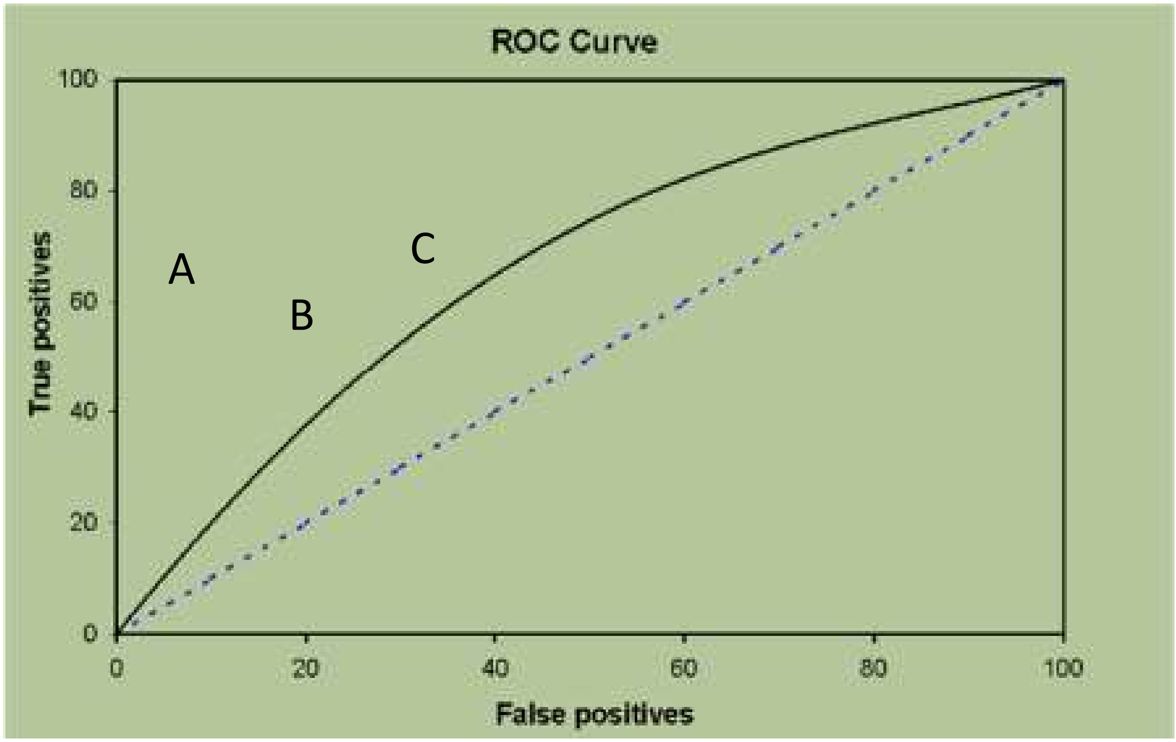
- Dashed line is “Random” classifier.
- The more “NorthWest” a classifier is, the better.
- Classifier A is objectively better than B or C, but it’s not clear that B is better than C.
- Can incorporate costs of errors into ROC curves to make them more relevant.

ROC Curves



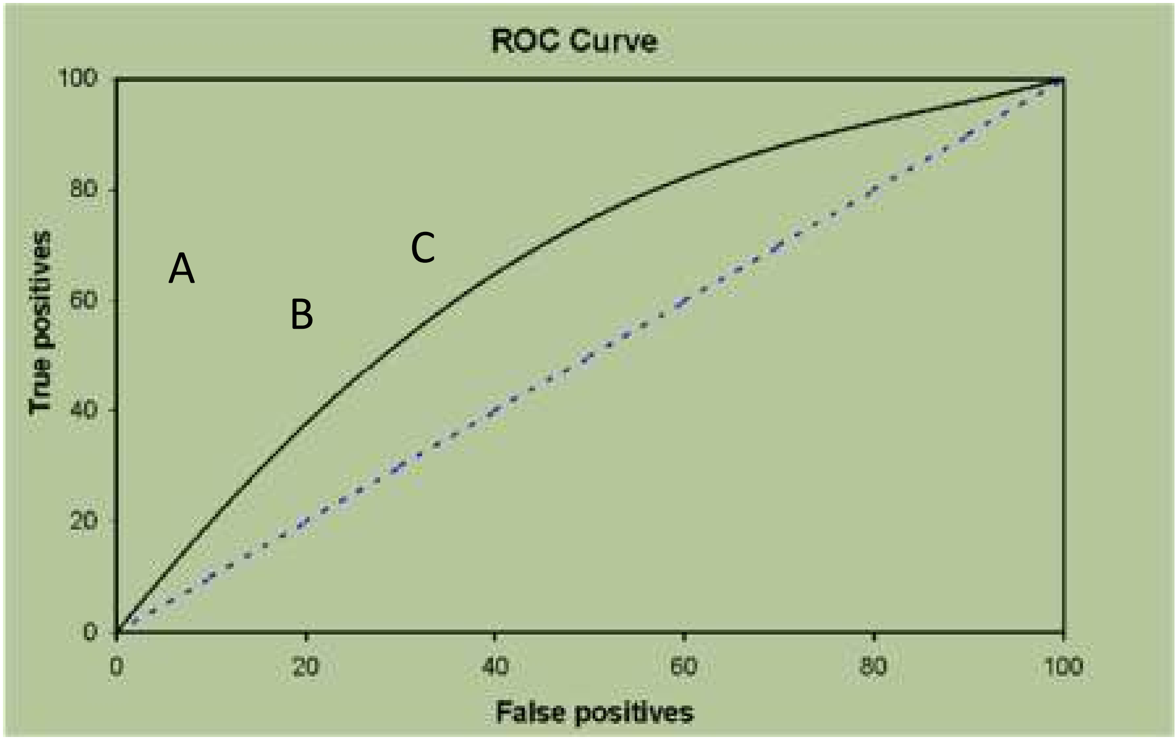
- Consider a Neural Network trained to classify classes 0 and 1.
- Its output is a number between 0 and 1 (*e.g.*, if we use a logistic output activation).
- To turn the output into a 0/1 class we compare the output to a number θ. If the output is greater than or equal to θ then we assign a class of 1, otherwise we assign a class of 0.
- Every value of θ is a unique classifier (a single point in the ROC plot above).
- As we sweep θ we trace something like the black curve in the figure.
- The family of classifiers traced by this process (the black curve above) is called the ROC curve of the neural network.

ROC Curves



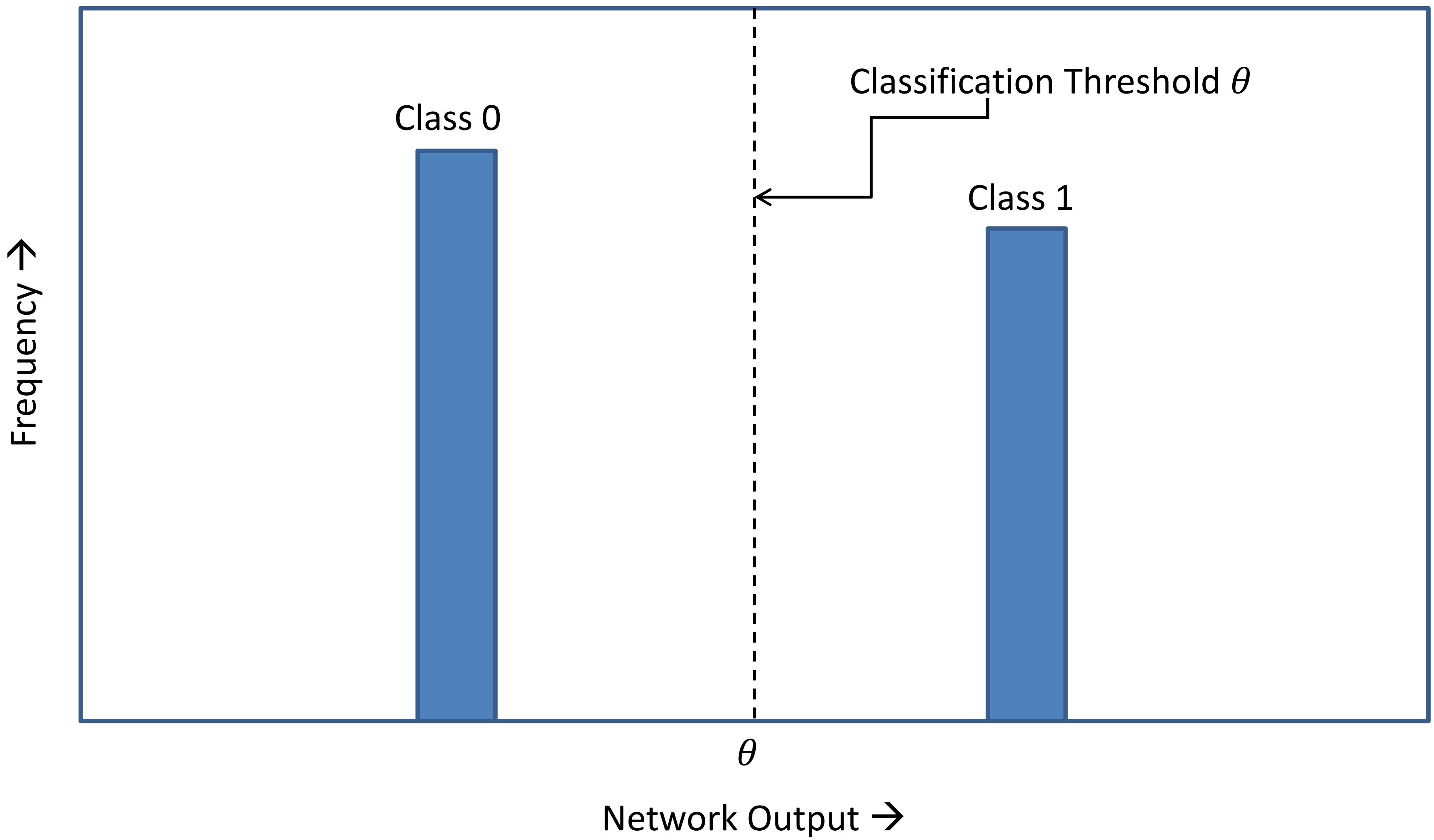
- To compare classifiers (or strictly speaking to compare the family of classifiers associated with a given model as above) we can compare their ROC curves.
- A standard procedure is to use the ROC curve's AUC (Area Under the Curve) to distinctly rank classifiers.
- Typically, the larger the AUC, the better the classifier.

ROC Curves



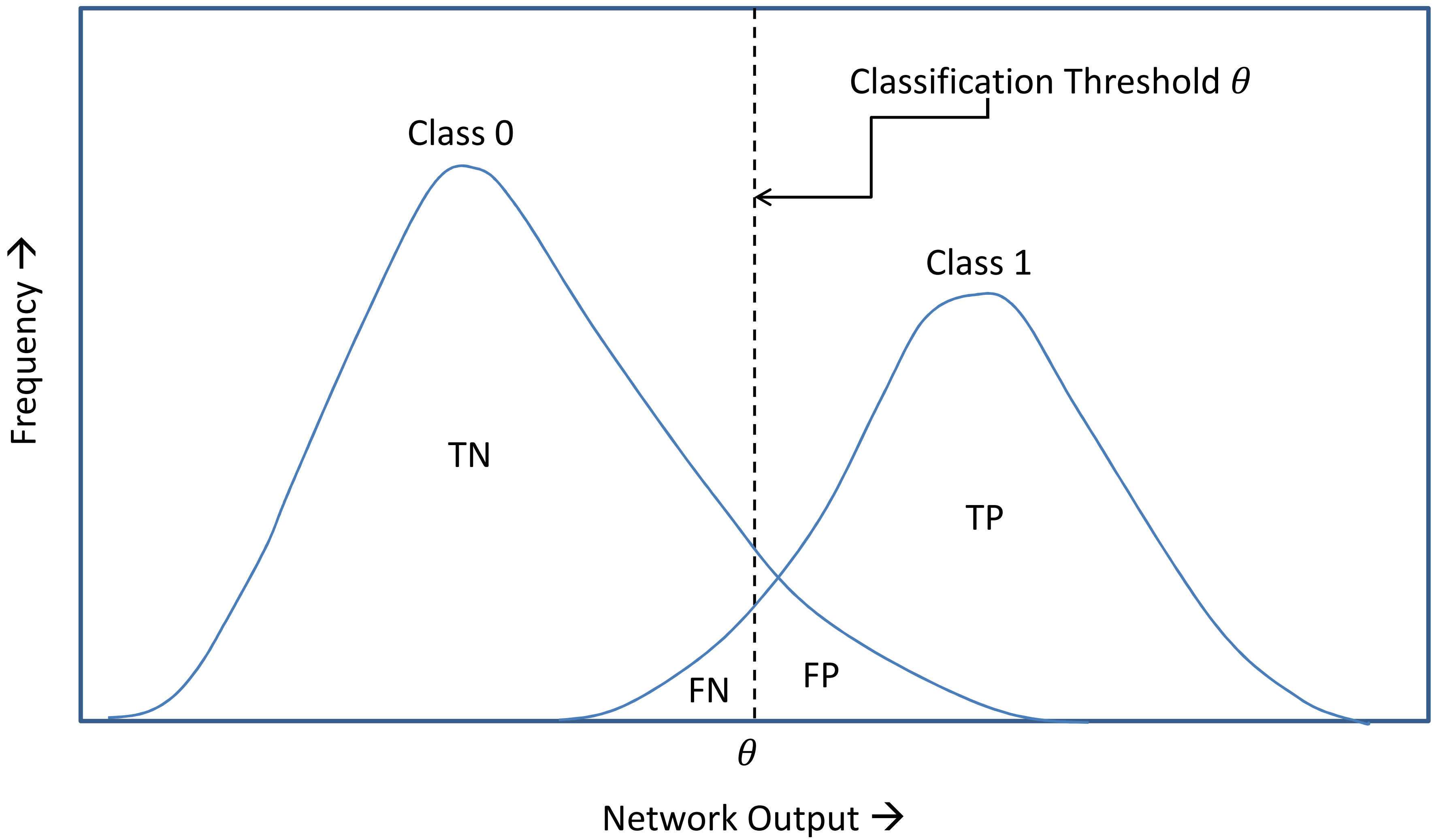
- To compare classifiers (or strictly speaking to compare the family of classifiers associated with a given model, as above) we can compare their ROC curves.
- A standard procedure is to use the ROC curve's AUC (Area Under the Curve) to uniquely rank classifiers.
- Ranking Criterion: the larger the AUC, the better the classifier.

Ideal Classifier



- The ideal classifier is able to separate the classes easily (remember the AND, OR, XOR examples earlier).

Noisy Classifier



- In the presence of noise, classes are blurred and there is an area of confusion (overlap). Notice tradeoff between FN and FP as θ is moved.

Some ROC Vocabulary

- Sensitivity: (TP) True Positive Rate
- Specificity: (TN) True Negative Rate
- (1 – Sensitivity): (FN) False Negative rate
- (1 – Specificity): (FP) False Positive rate

	Null hypothesis H_0 true	Null hypothesis H_0 false
H_0 rejected	FP (α)	TP ($1-\beta$)
H_0 accepted	TN	FN

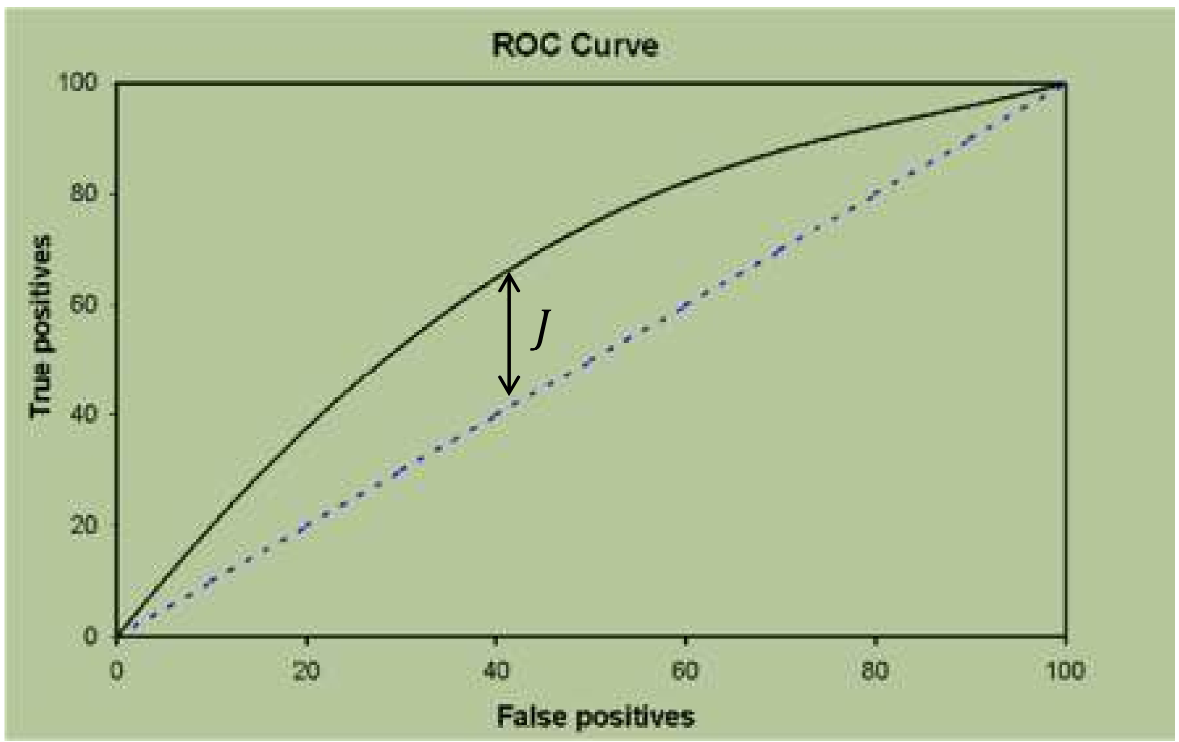
- α = Prob of Type I error (FP)
- β = Prob of Type II error (FN)

Optimizing Accuracy

- The Younden Index J is often used as the optimal criterion for choosing the threshold of a classifier:

$$J = Sensitivity + Specificity - 1 = TP - FP.$$

- Graphically, it represents the maximum vertical distance between the random classifier line and the ROC Curve:



Costs and ROC Analysis

- But the Younden Index doesn't take into account asymmetric error costs. At the other extreme, we can consider only costs (and not classifier performance) to find the classifier threshold θ.
- Suppose a trained classifier neural network's output, o, represents the probability that Class 1 is True. (**Aside*: an interesting Class Project would be to find out how a Classifier Neural Network with logistic output activation compares to Logistic Regression.)
- For example, Class 1 could mean "A Trade is *Profitable*," while Class 0 could represent "A Trade is *Unprofitable*."
- Generally, there are different costs/benefits associated with the different outcomes.
- Let P be the Profit associated with entering a Profitable Trade, and L be the Loss associated with entering an Unprofitable Trade.
- We should enter a Trade only when the expected profit is greater than the expected loss:

$$oP > (1 - o)L.$$

Costs and ROC Analysis

- Since our classification of a Class 1 (i.e. a Profitable Trade) is made whenever the network's output o exceeds the threshold θ:

$$o > \theta,$$

- This immediately suggests an expression for the threshold for entering profitable trades, independent of classifier performance:

$$\theta = \frac{L}{L + P}.$$

- In HFT usually $L > P$ due to transaction costs and other frictions (what condition does this impose on θ?).

Incorporating Classifier Performance: Misclassification Costs

- What about ***Misclassification Errors***?
- We can see that Profit will only result from entering a Profitable Trade:

$$Expected\ Profit = TP \cdot P. \quad (Why?)$$

- A Loss can occur either from entering an Unprofitable Trade, or from ***not*** entering a potentially Profitable Trade:

$$Expected\ Loss = FP \cdot L + FN \cdot P. \quad (Why?)$$

Incorporating Classifier Performance: Misclassification Costs

- The condition for profitability is that the Expected Profit exceed the Expected Loss:

$$TP \cdot P > FP \cdot L + FN \cdot P, or:$$

$$(TP - FN) \cdot P > FP \cdot L.$$

- This imposes a condition on the classifier's performance (and misclassification costs) given the trade's profit and loss:

$$\frac{(TP - FN)}{FP} > \frac{L}{P}.$$

New Topic: Association Rules

- ***Association Rules*** (also known as Market Basket Analysis) are an example of Unsupervised Learning.
- Association Rules are of the form "If X then Y."
- Example: A department store collects information on customer transactions. We wish to find Association Rules of the form: *"If jeans and t-shirts are purchased together, then belts are also purchased often."*
- Or, *"If a customer is female and she purchased vitamin supplements then calcium supplements are often also purchased."*
- The idea is to discover these association rules *Automatically.*
- Problem: The Number of possible items (occurrences) can be very large, and number of transactions can be huge.

Association Rules

- More formally, Let $\mathcal{I} = \{i_1, i_2, \dots, i_n\}$ be a set of literals called <u>*Generalized Items*</u>. These could be all the items sold at a store, along with customer demographic information, item category information, etc., or they could be the S&P500 stocks along with industry categories, macroeconomic states, etc.
- Let $\mathcal{D}$ be a <u>*Database*</u> of m transactions (or occurrences) within a specified period of interest: $\mathcal{D} = \{d_1, d_2, \dots, d_m\}$ is a set of m binary n-tuples $\{0,1\}^n$ where 0/1 represents the presence/absence or occurrence/non-occurrence of a Generalized Item.
- We call a subset X of $\mathcal{I}$ an <u>*Itemset*</u>.

Association Rules

- Suppose we have two disjoint itemsets $X \subset \mathcal{J}$ and $\mathrm{Y} \subset \mathcal{J}$ such that $X \cap Y = \emptyset$
- We say there is an *Association Rule* "$X \Rightarrow Y$" if both itemsets are frequently present together in the same transaction (or basket, or occurrence).
- For example, if $X = \{i_2, i_7\}$ and $Y = \{i_3\}$, the Rule $X \Rightarrow Y$ can be interpreted as saying: "*When items i_2 and i_7 are present in the same transaction (occur together), then item i_3 is often also present (often also occurs).*"
- Notice that the requirement that $X \cap Y = \emptyset$ ensures that we don't end up with trivial associations like if i_2 and i_3 are present (occur) then i_3 is also present (occurs).

The Association Rule Problem

- Statement of the *Association Rule Problem*: "*Find Interesting Association Rules from the Database of Transactions* $\mathcal{D}$*.*"
- The key here is to define what "interesting" means, and to figure out an automated way that this computationally intense problem can be tractably solved.
- The approach most often used today is the *Support-Confidence Framework*.

The Support-Confidence Framework

- The Support-Confidence Framework defines an Interesting Association as follows:
- An Interesting Association Rule $X \Rightarrow Y$ occurs when:
  1. X and Y have support s, and
  2. X and Y have confidence c.
- *Support* is defined as a lower bound on the percentage of transactions in $\mathcal{D}$ that contain both itemsets X and Y; *i.e.*, $P(X \cap Y) \geq s$.
- *Confidence* is defined as the lower bound on the percentage of those transactions containing X that also contain Y; *i.e.*, $P(Y|X) \geq c$.

The Support-Confidence Framework

- For example, suppose the transaction database $\mathcal{D}$ contains 1 million transactions and 10,000 of those contain both itemsets X and Y.
- In this case, the support of $X \Rightarrow Y$ is

$$s = \frac{10^4}{10^6} = 1\%.$$

- Likewise, if 50,000 transactions contain X and, out of those, 10,000 also contain Y, then $X \Rightarrow Y$ has a confidence of

$$c = \frac{10^4}{5 \times 10^4} = 20\%.$$

Tractability

- It is impossible to check all possible Itemset combinations for Interesting Association Rules.
- In fact, there are $\sum_{i=1}^{n} \binom{n}{i} = 2^n - 1$ such combinations, where n is the number of items, which could number in the thousands. For example, if $n = 1{,}000$ there would be over 10^{300} possibilities to check, which is prohibitive.
- However, we don’t have to check all possibilities…

Tractability

- Suppose we have found an Itemset that is supported. Then we know that all its subsets are supported (why?). For example, if $\{i_3, i_4, i_8\}$ is supported, then we know that $\{i_3\}$, $\{i_4\}$, $\{i_8\}$, $\{i_3, i_4\}$, $\{i_3, i_8\}$, etc… are all supported and we don't need to check them.
- Also, all supersets of an Itemset that is not supported are themselves not supported (why?), so we don't need to check them.
- Likewise, all supersets of a confident Itemset are confident, and all subsets of a non-confident Itemset are not confident. (why?)
- We can use these findings to prune the search space of Interesting Association Rules dramatically. We can start with 2-Itemsets and prune all supersets of those that are not supported, then explore 3-Itemsets etc. Then, we can prune all subsets of k-Itemsets that are not confident.

Throwing the Baby With the Bathwater

- In my opinion, Support-Confidence is not very useful at all!
  1. The reason is that we should define Association Rules to be Interesting only when they deviate from Random Chance; *i.e.*, we want *Non-Random Associations*, which the Support-Confidence Framework does ***not*** distinguish from Random ones.
  2. Suppose Milk occurs in 60% of all transactions at a grocery store, and Bread occurs in 50% of all transactions. This means that *By Random Chance Alone* we would expect Milk and Bread to occur together in 30% of all transactions. This might seem like a high confidence number, but it means nothing. It would be more interesting if they occurred together either significantly more frequently or significantly less frequently than the 30% expected by random chance alone.
  3. Support can ignore anti-correlated occurrences. For example, Coke and Pepsi may each occur in 50% of all baskets independently, (meaning we would expect them to occur together in 25% of all transactions by random chance alone). However, if they occur in 0.01% of all transactions together, this means they have a non-random effect on each other. Yet this might seem like a low support number, so we would be throwing away this information…
  4. We'll fix this in the next lecture.