Machine Learning Baruch College Lecture 3

Miguel A. Castro

Today We’ll Cover:

- Questions
- Review of Last Lecture
- ROC Curves and Classification
- A Case Study: European Companies
- Start on Association Rules

Universal Approximation

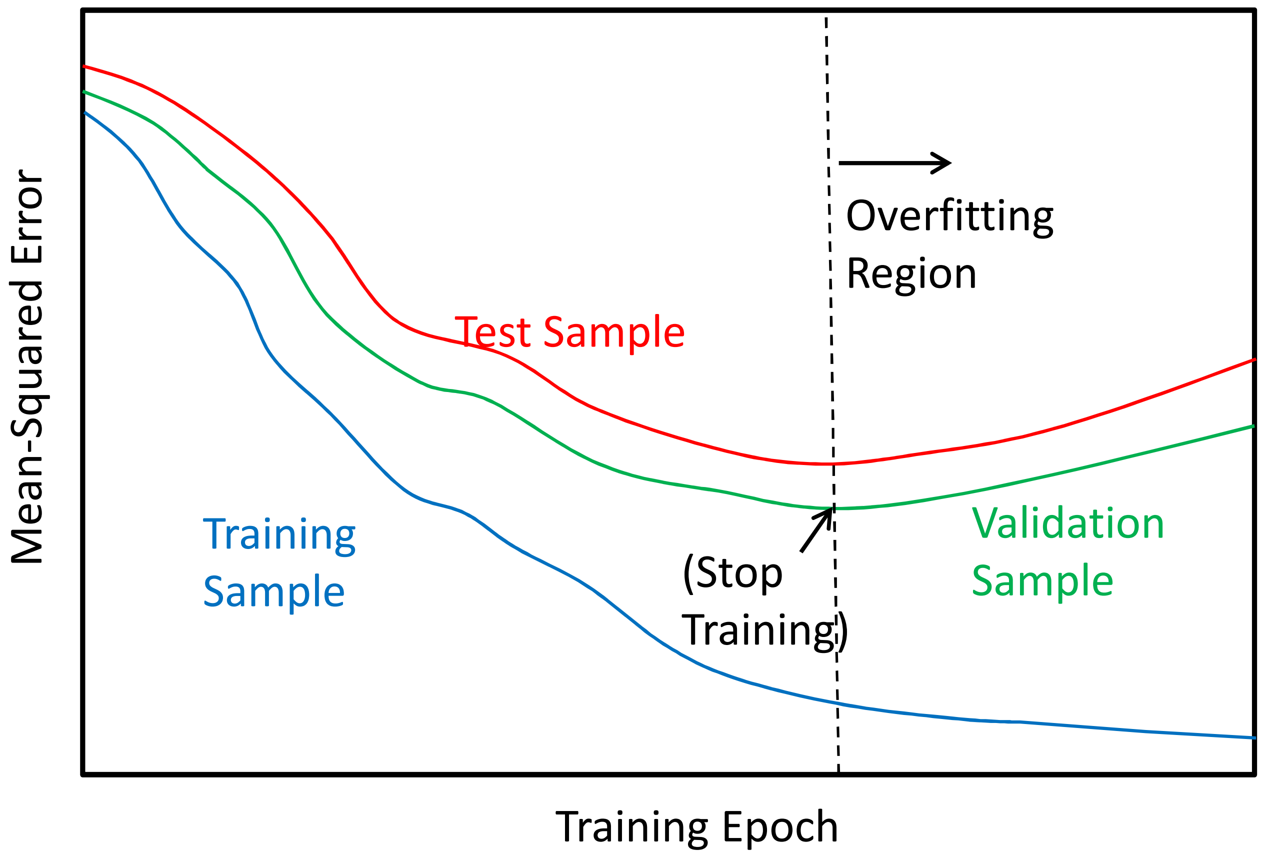
- Multi-layer, Feedforward Neural Networks with ***Non-linear Activations*** are Universal Approximators
  - Able to discover input/output mappings arbitrarily well
  - Example: Classification
  - Example: Function Approximation
  - Classification is, in a way, a subset of Function Approximation, but treated and evaluated differently.
- There are 2 minimum criteria for universal approximation:
  1. At least one Hidden Layer, and
  2. Non-linear Activations in the Hidden Layer.
- In order for "learning" to be automated we require *Differentiable Activations* ➔ Automated Learning Rules such as *Backpropagation.*

Overfitting vs. Generalization

- 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).

Early Stopping to Avoid Overfitting

- 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.



Pruning and Regularization to Avoid Overfitting

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

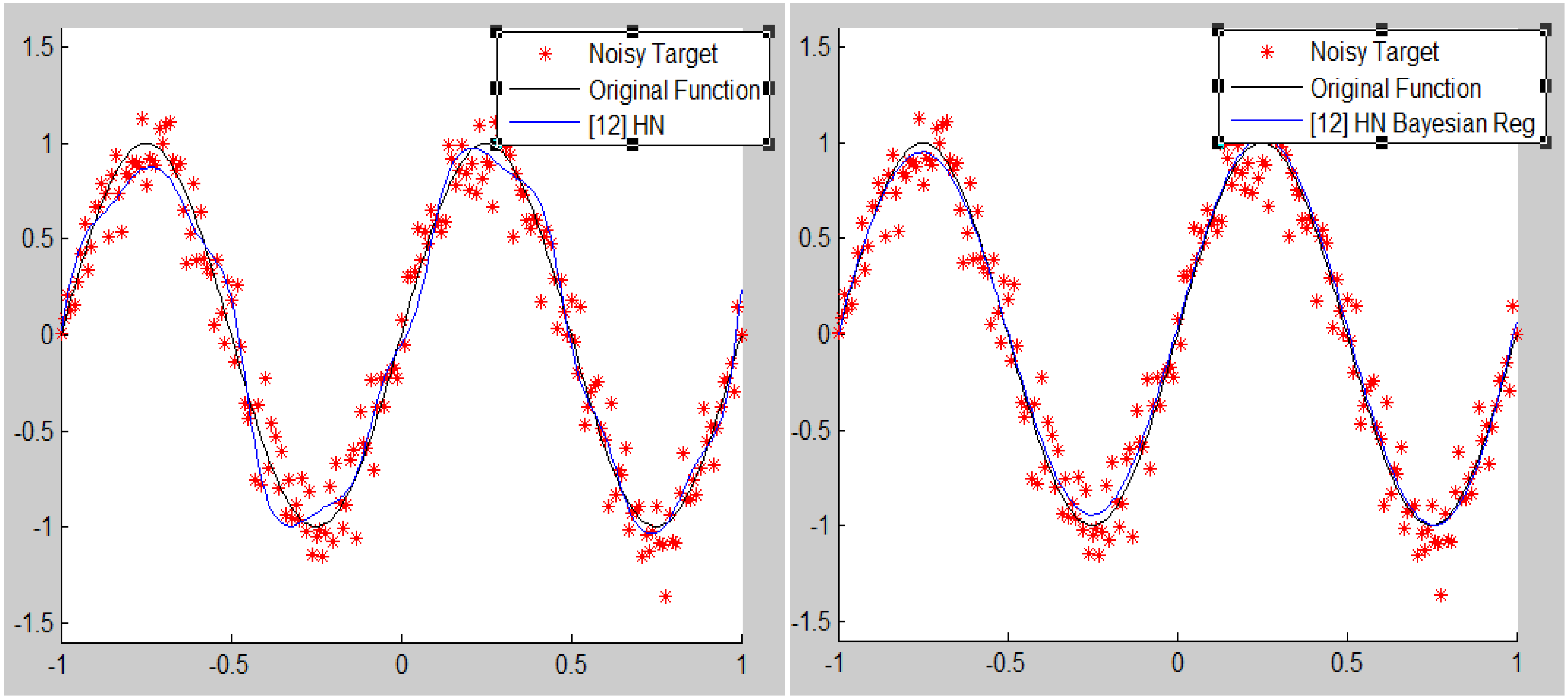
$$\mathcal{F}_W = (1-\lambda)\frac{1}{N}\sum_i \tfrac{1}{2}(t_i - a_i)^2 + \lambda \sum_{ij} \tfrac{1}{2}{w_{ij}}^2 \,.$$

Regularization

- 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 which uses "mutations" instead of "temperature" to inject noise in the learning process…).
  - Bayesian Regularization:
    - Introduces more objective criteria for regularizing parameters;
    - Introduces objective criteria for comparing among models, including non-neural network approaches: "*Effective Number of Parameters*";
    - Does not necessarily decrease the data size (although keeping a Hold-Out set is still a good idea);
    - Bonus: Obtains the <u>*Effective Number of Degrees of Freedom*</u> (weights).
    - This can be used to re-train a new network with a <u>*smaller architecture*</u> 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.

Overfitting

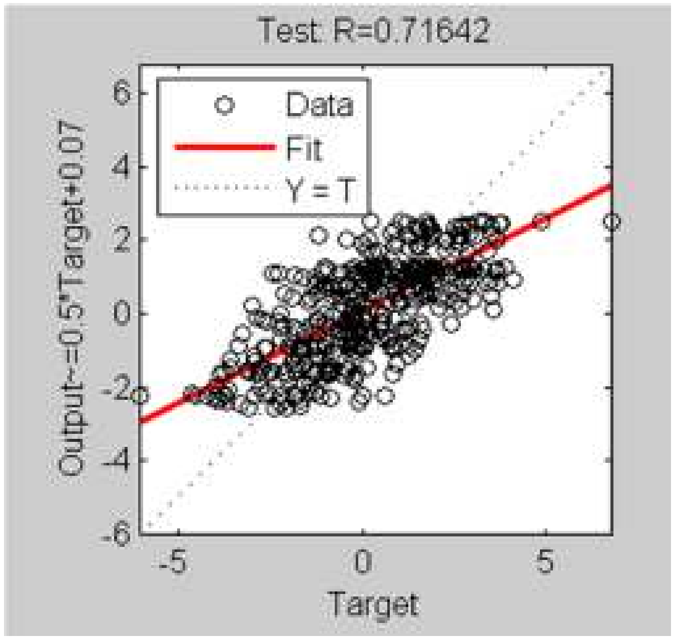
- Bayesian Regularization produces smoother fit:



- Both start out with 12 HN.
- Number of Effective Parameters went from 37 to 9 in this example (4 HN).
- R^2s comparable (around the max) but a bit higher for BR in this example.
- But fit is smoother with Bayesian Regularization (better Generalization).

Performance Testing

- One way to test Function Approximation *Performance* is to look at the regression of Actual Output vs. Target (on the *Test* or *Hold-Out* set):



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

Single vs. Multiple Hidden Layers

- 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, requiring fewer training sessions.
- This does not constitute proof that multiple hidden layers are better than one, but it is a potentially interesting finding.
- A more exhaustive empirical study (perhaps a Monte Carlo simulation under various conditions) may offer support for this hypothesis (Project idea?).

Aside: SR and Signal-to-Noise

- Suppose we are confronted with a noisy signal (as in HFT).
- We build a reasonable (parsimonious and regularized) predictive model.
- We extract a signal (a prediction) using this model.
- We measure the model's R^2 (using the Test or Hold-Out 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 optimal), then:
  - We can make inferences about the signal-to-noise ratio $\left({}^{1}/_{n}\right)$ and hence about the model's risk. For example,

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

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

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

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

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

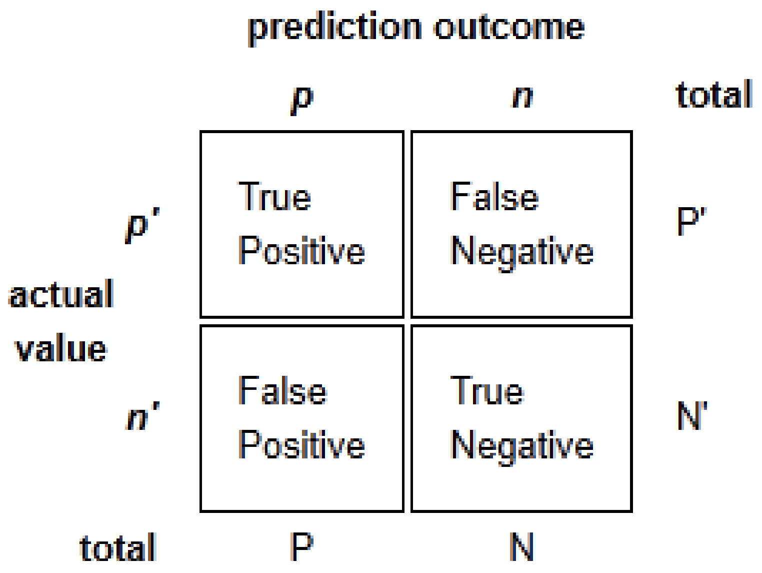
- This is an example of how useful it is to "back out" signal-to-noise ratios from model stats making certain assumptions.
- An interesting question would be to extend this analysis to two or more securities. (Project idea?)

Classifiers and Errors

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

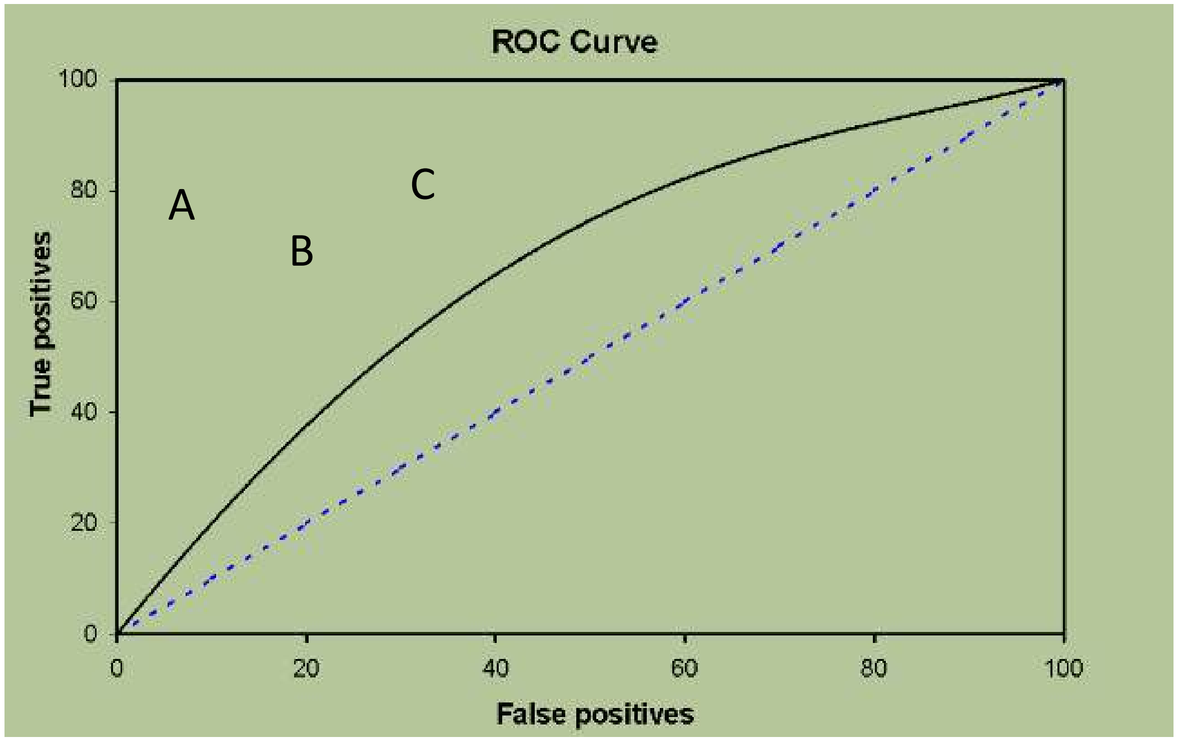
Confusion Matrix

- *Confusion Matrix* Quantifies predicted vs. actual classes (should really be called Classification Matrix).
- Quantifies misclassification error rates and correct classification rates.



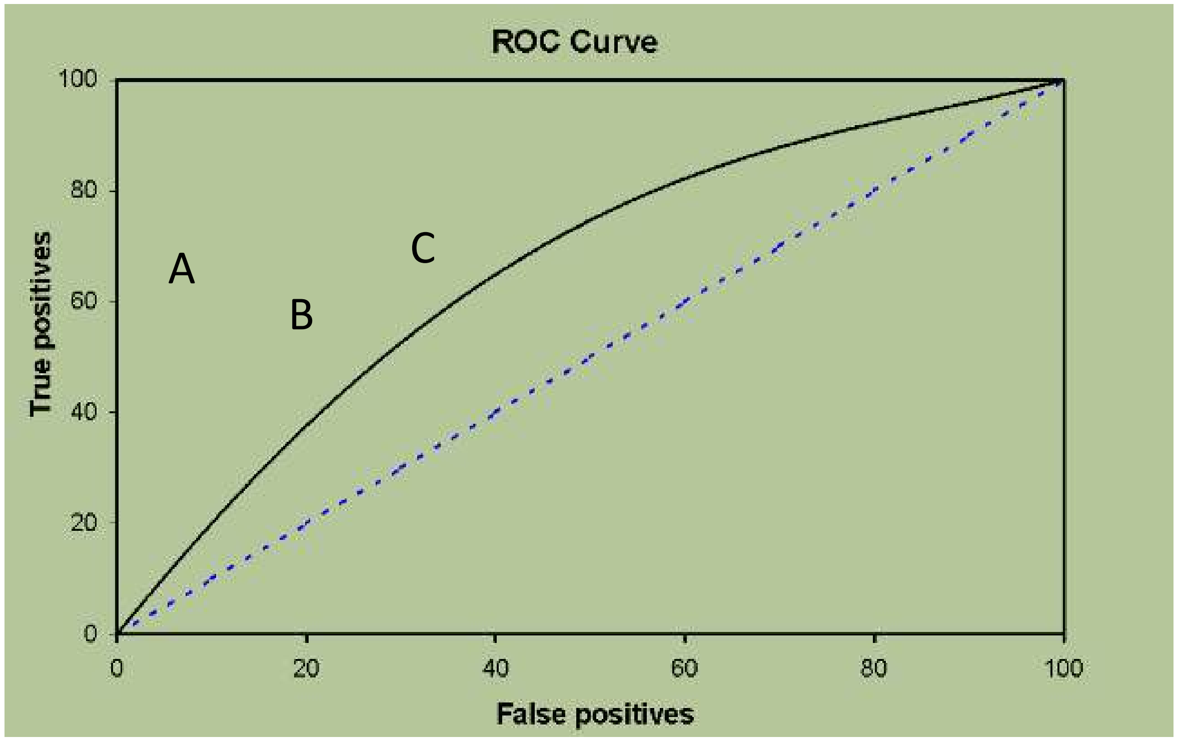
		prediction outcome		
		p	*n*	total
actual value	*p'*	True Positive	False Negative	P'
	n'	False Positive	True Negative	N'
	total	P	N	

ROC Curves



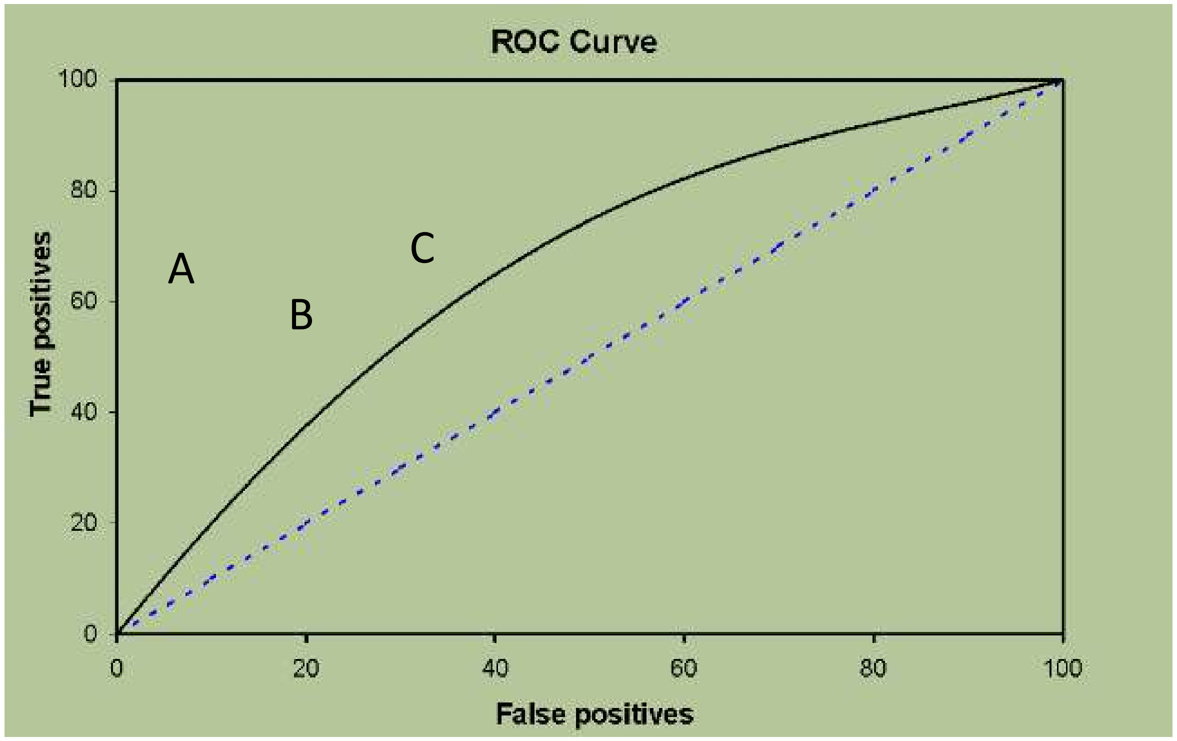
- Dashed line represents the “Random” classifier (does no better than flipping a coin).
- The more “NorthWest” a classifier is, the better; point (0,100) is the Perfect Classifier.
- Classifier A is objectively better than B or C, but it’s not clear that B is better than C.
- Can incorporate costs of errors into ROC curves to make differences among classifiers more relevant to real-life applications.

ROC Curves



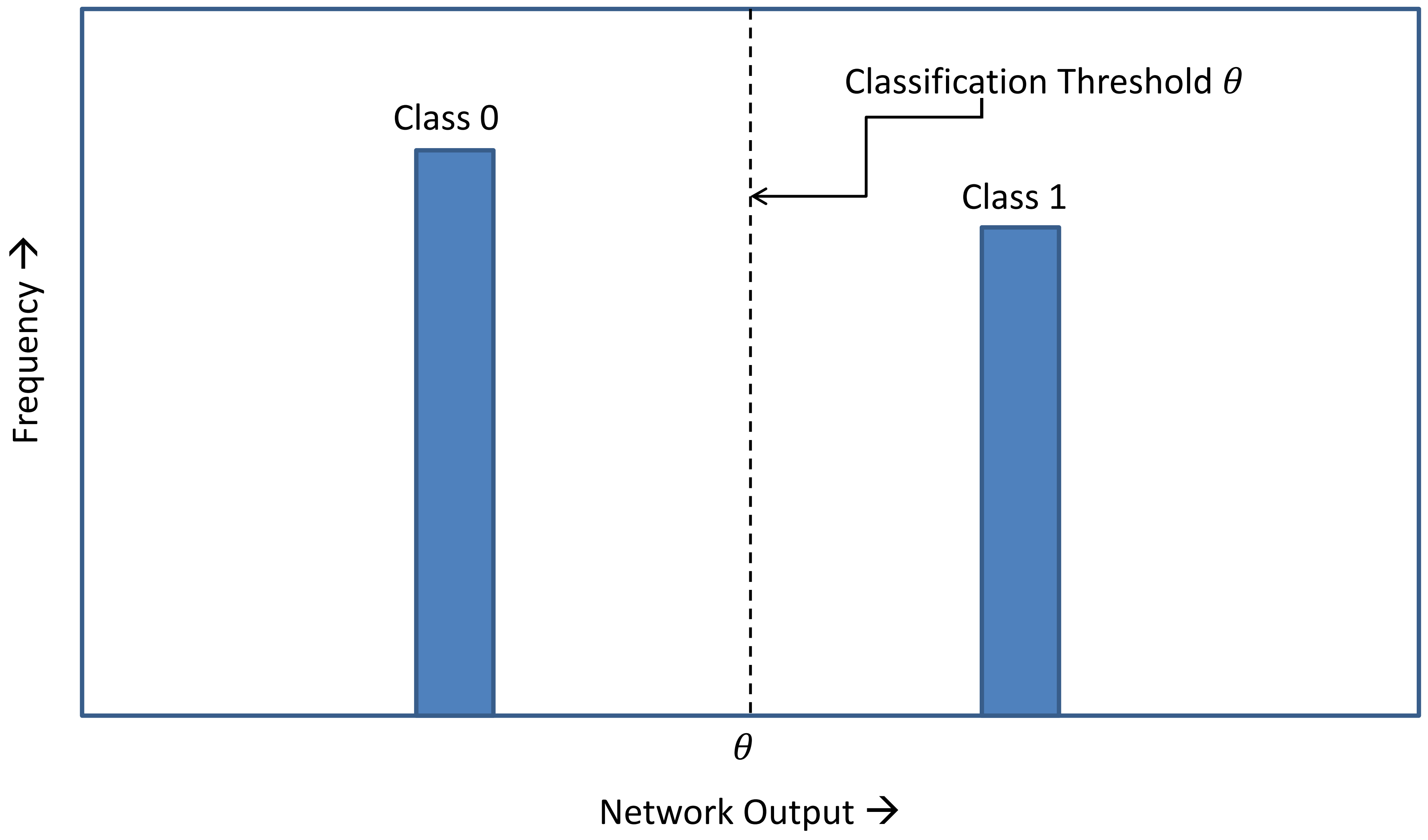
- Consider a Neural Network trained to classify classes 0 and 1.
- Its output is a number between 0 and 1 (*e.g.*, if we use a logistic output activation).
- To turn the output into a 0/1 class we compare the output to a *threshold* θ. If the output is greater than or equal to θ then we assign a class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

ROC Curves



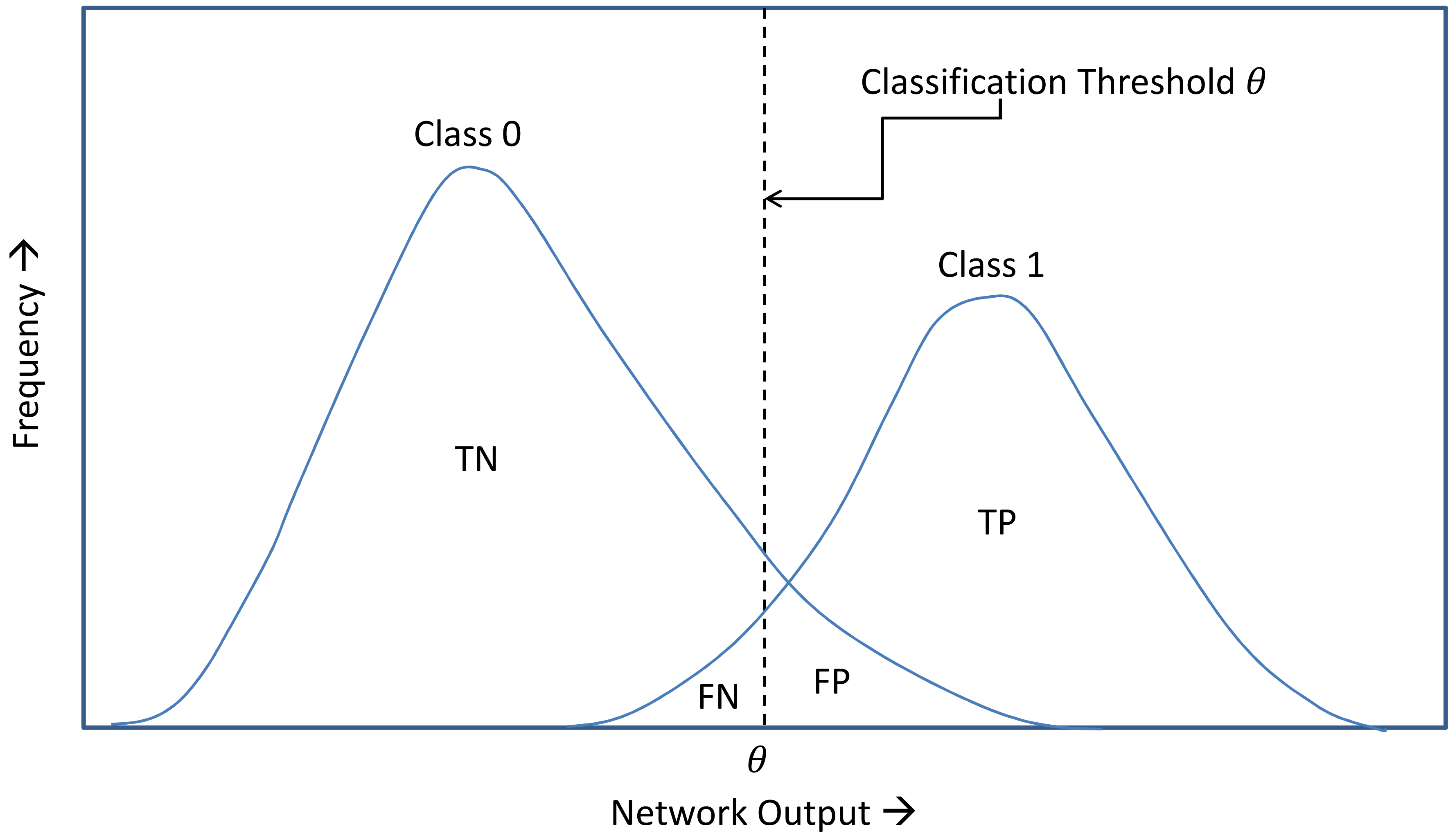
- To compare classifiers (or strictly speaking to compare the family of classifiers associated with a given model, as in the previous slide) we can compare their ROC curves.
- Curves can cross, so a standard procedure is to use the ROC curve's *AUC* (Area Under the Curve) to uniquely rank classifiers.
- Ranking Criterion: the larger the AUC, the better the classifier.
- Note that we're being a bit loose with notation here, confusing a model with a classifier. Strictly speaking, a unique classifier consists of *both* the trained model and a unique threshold, and is represented by a single point in ROC space. An ROC curve (the black curve above) represents a single model but with an infinite number of classifiers corresponding to an infinite number of thresholds. However we often interchangeably use the word "classifier" to refer to a single point or to refer to the ROC curve, when in the latter case we really mean a family of classifiers obtained by a *single model* and many thresholds.

Ideal Classifier



- The ideal classifier is able to separate the classes easily (remember the AND, OR, XOR and other examples that we looked at earlier).

Noisy Classifier



- In the presence of noise, however, classes are blurred and there is an area of confusion (overlap). Notice the tradeoff between FNs and FPs as θ is moved from left to right.

Classifier Vocabulary

- The ***True Positive Rate***, tp, is the proportion of correctly-identified positive outcomes:

$$tp = \frac{TP}{TP + FN},$$

where TP is the number (or frequency) of correctly identified positives and FN is the number (or frequency) of incorrectly identified positives (false negatives).

- The ***False Positive Rate***, fp, is the proportion of incorrectly-classified negative outcomes:

$$fp = \frac{FP}{TN + FP},$$

where FP is the number (or frequency) of incorrectly identified negatives (false positives) and TN is the number (or frequency) of correctly identified negatives.

- The ***True Negative Rate***, tn, is the proportion of correctly-identified negative outcomes:

$$tn = \frac{TN}{TN + FP},$$

- The ***False Negative Rate***, fn, is the proportion of incorrectly-classified positive outcomes:

$$fn = \frac{FN}{TP + FN}.$$

Classifier Performance Measures

- ***Recall*** or ***Sensitivity***: $Rec = tp$ (the True Positive Rate).
- ***Specificity***: $Spec = tn$ (the True Negative Rate).
- ***Precision***: $Prec$ is the proportion of correctly-classified positive predictions:

$$Prec = \frac{TP}{FP + TP}.$$

- ***Accuracy***: Acc, is the proportion of correctly-classified outcomes:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}.$$

- ***Confusion***: $Conf$, is the proportion of incorrectly-classified outcomes:

$$Conf = \frac{FP + FN}{TP + TN + FP + FN} = 1 - Acc.$$

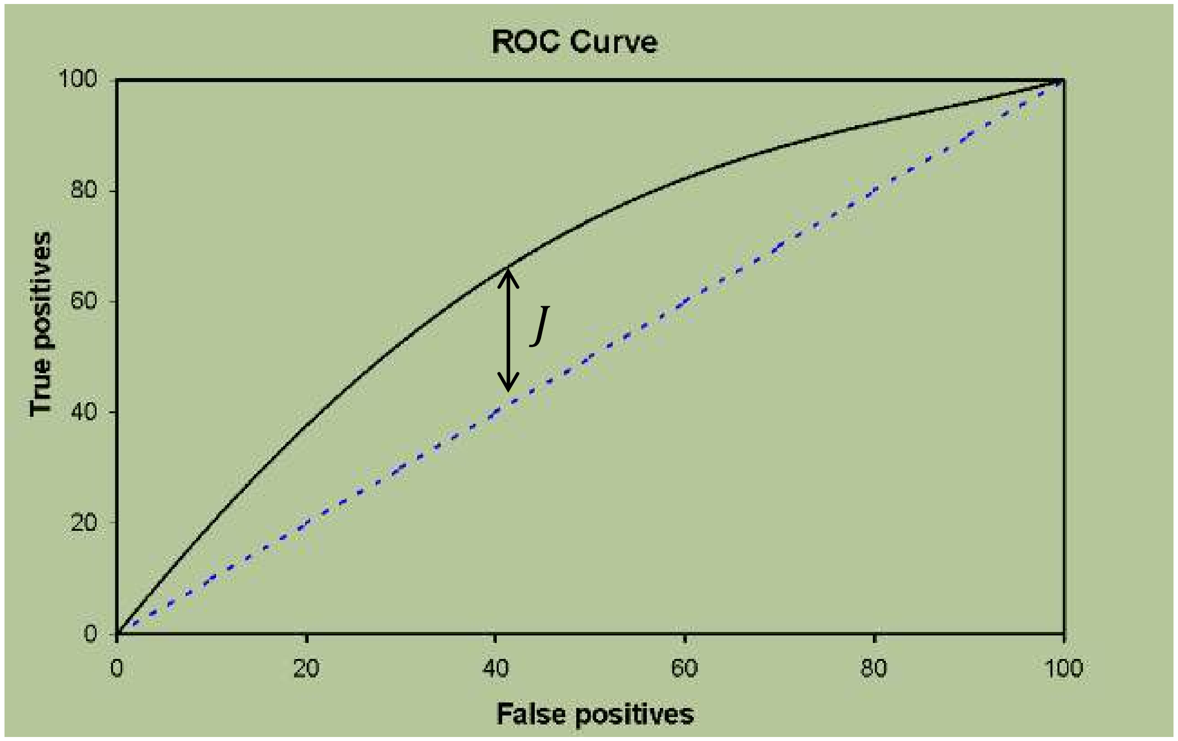
- Note that *Accuracy* has problems as a performance metric. For example, if out of 1000 outcomes there are 990 positive ones, a dumb classifier that returns ***always*** the output "positive" would have an Accuracy of 99%.

Performance Measures Using ROC Curve

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

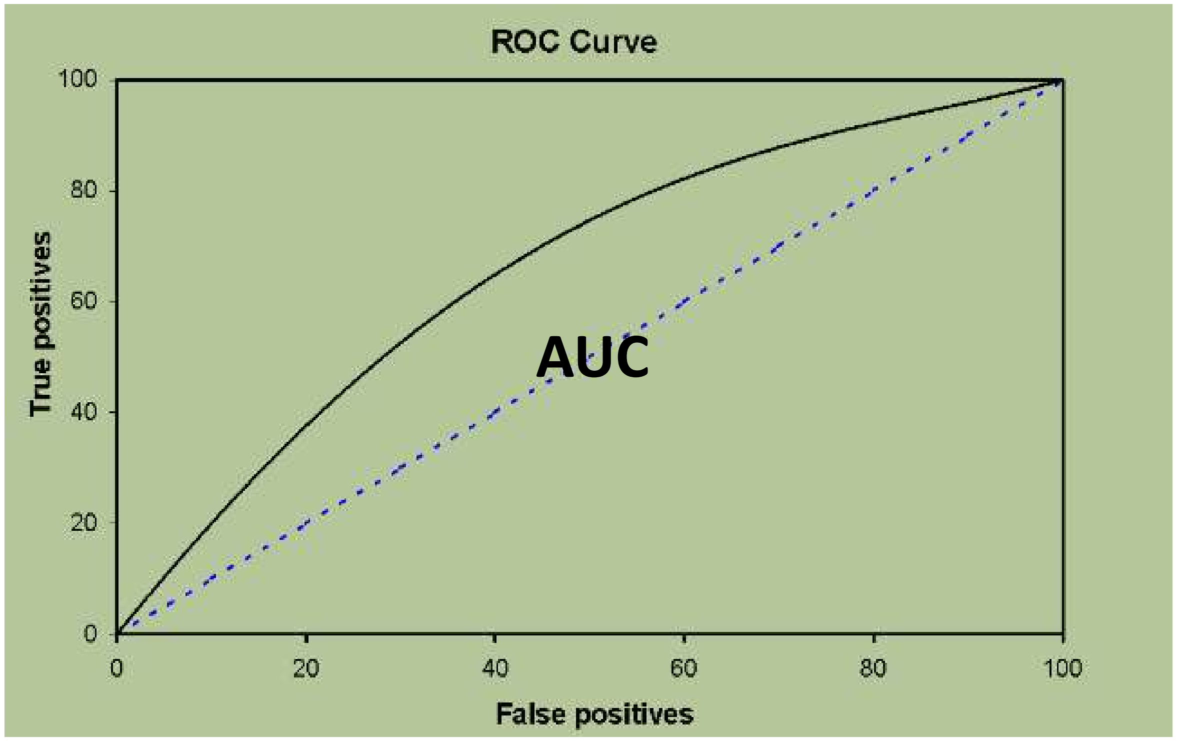
$$J = tp - fp = Sensitivity + Specificity - 1.$$

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



Performance Measures Using ROC Curve

- Another performance measure, as mentioned above, is the ***<u>Area Under the Curve</u>*** (*<u>AUC</u>*) of the ROC curve of a family of classifiers.

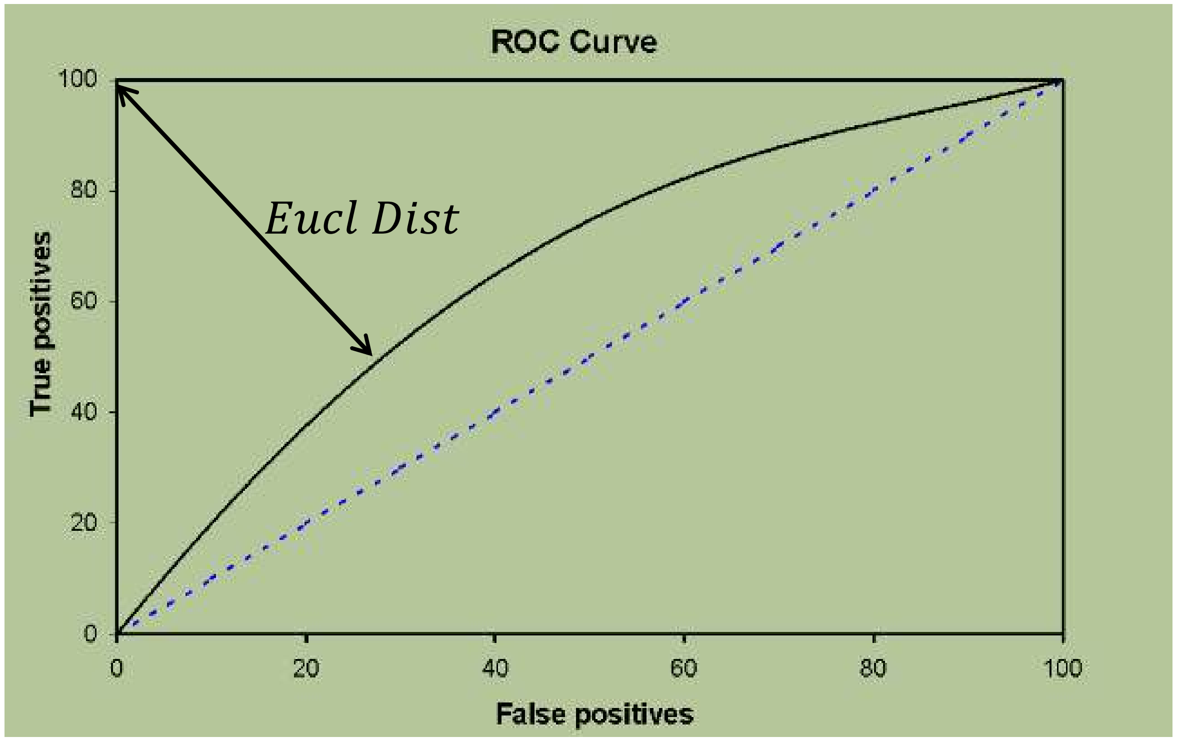


- This metric can compare among different families of classifiers (*i.e.*, among different models), where the larger the AUC, the higher the classifier performance (strictly speaking the higher the *model* performance).

Performance Measures Using ROC Curve

- The Euclidean Distance Accuracy (<u>*DistAcc*</u>) uses the (normalized) Euclidian Distance $(Eucl\ Dist)$ to the perfect classifier:

$$DistAcc = 1 - \sqrt{\tfrac{1}{2}[(1 - tp)^2 + fp^2]}.$$



- The $Eucl\ Dist$ is normalized so that this metric ranges from 0 for the worst classifier, to 1 for the best classifier (note fp and tp are expressed as fractions instead of percentages).

The Random Classifier

- Consider the following (fraction) confusion matrix:

		Classification	
		+	-
Actual	+	TP	FN
	-	FP	TN

- A classifier would be ***Random*** if (why?):

$$\widehat{TP} = (TP + FN)(TP + FP),$$
$$\widehat{FN} = (TP + FN)(FN + TN),$$
$$\widehat{FP} = (TP + FP)(FP + TN),$$
$$\widehat{TN} = (FP + TN)(TP + FP).$$

- It is easy to show that the *Random Classifier* traces the dashed line in the ROC curves shown earlier. (Why?)

The Random Classifier

- This suggests a classifier performance metric that reflects the deviation from the Random Classifier.
- Consider the squared deviations:

$$\chi^2 = \frac{\left(TP - \widehat{TP}\right)^2}{\widehat{TP}} + \frac{\left(FP - \widehat{FP}\right)^2}{\widehat{FP}} + \frac{\left(FN - \widehat{FN}\right)^2}{\widehat{FN}} + \frac{\left(TN - \widehat{TN}\right)^2}{\widehat{TN}}.$$

- This quantity has a Chi-Squared Distribution with 1 degree of freedom.
- It can be used to test the hypothesis that the classifier is not Random.
- Note it does not distinguish between a better-than-Random or a worse-than-Random classifier (it measures “distance” from the Random classifier, which is symmetric).
- This is not a problem, as we can always reverse the classification output in the worse-than-Random case.

The Random Classifier

- Another Performance Metric is the *Odds Ratio* or *Cross Product*:

$$OR = \frac{tp/fn}{fp/tn} = \frac{tp * tn}{fp * fn} = \frac{TP * TN}{FP * FN}.$$

- Problems arise when either/both of the quantities in the denominator is/are zero.

- Yule's Q transforms the OR to a scale between -1 and +1:

$$Q = \frac{OR - 1}{OR + 1}.$$

- The Yule Q is sensitive to which side of the "random" line the classifier is located in; a negative Q represents a "worse-than-random" classifier (*i.e.*, it's below the dashed line in the ROC curve), whereas a positive Q represents a "better-than-random" classifier (*i.e.*, it's above the dashed line in the ROC curve).

Classifiers and Hypothesis Testing

- As already mentioned, we can frame classifiers in the language of *Hypothesis Testing*:

Null Hypothesis H_0

	True	False
H_0 Rejected	FN (α)	TN
H_0 Accepted	TP	FP (β)

- α = Prob of Type II Error (FN); $1 - \alpha$ is “Confidence”
- β = Prob of Type I Error (FP); $1 - \beta$ is “Power”

Costs and ROC Analysis

- But the above performance metrics don't take into account 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 <u>*Misclassification Errors*</u>?
- 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 <u>*not*</u> 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:

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

Misclassification Costs

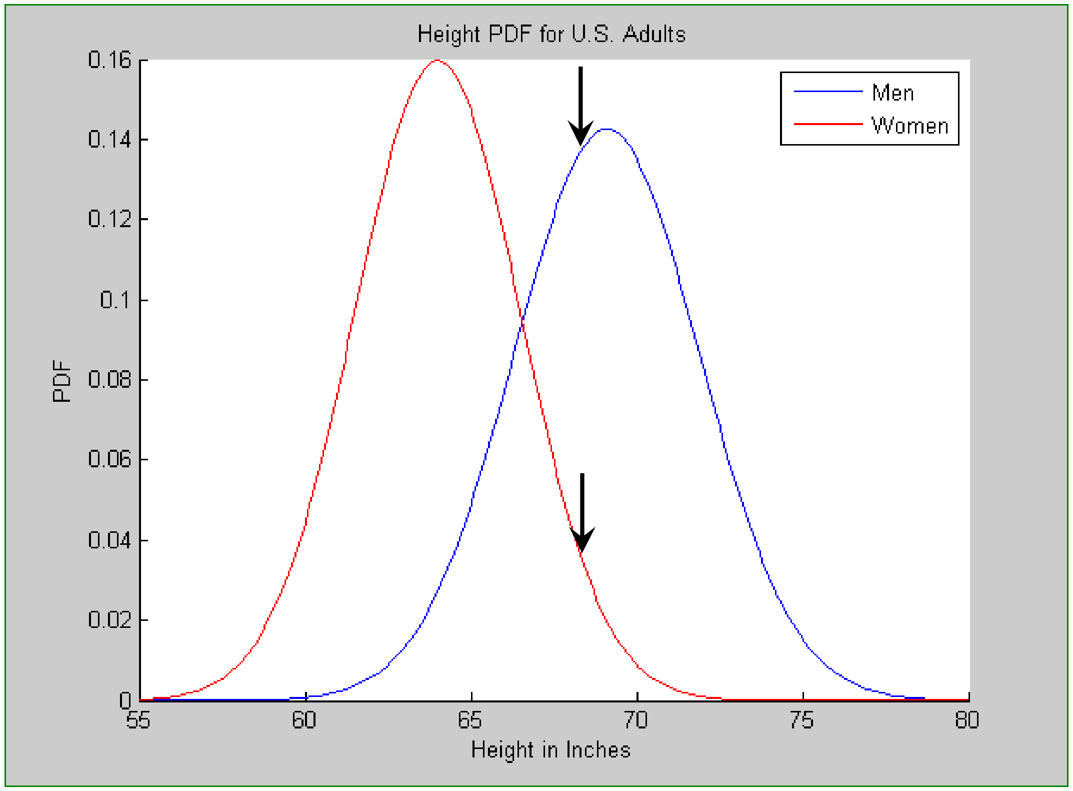
- Generally, there is a tradeoff between Type I Error (FPs) and Type II Error (FNs).
- This can be incorporated into performance metrics to adjust for misclassification costs.
- For example, we can adjust the Euclidean Distance Accuracy as follows:

$$Cost\ Adjusted\ Eucl\ Dist\ Acc = 1 - \sqrt{W(1-tp)^2 + (1-W)fp^2},$$

  where $0 < W < 1$, and where the FNs are penalized by W (why?) and the FPs are penalized by $(1-W)$.
- This cost adjustment models the tradeoff between FPs and FNs, and can be used in the other performance metrics as well.
- Note that in biological organisms, W is generally larger than $1-W$ (Type II Errors are generally more costly); but this is often not the case in HFT (why?).
- In practical problems, as we saw in our trading example earlier, FPs were costlier than FNs due to transaction costs/microstructure frictions. This also holds in a court of law, etc. (why?)

Naïve Bayes Classifiers

- Consider an example with two classes ($c_1 = Male, c_2 = Female$) and a single attribute, ($Height$):



- Suppose we are told an adult's height data measurement is $d = 68\ inches$. What can we infer about the person's sex?
- All else being equal, the person is more likely to be Male than female (75% vs 25% probability) given only the height information.

Naïve Bayes Classifiers

- Formally, we want to know $P(c_j|d)$, the probability of class c_j given that we have observed d.
- From Bayes Theorem this is just:

$$P(c_j|d) = \frac{P(d|c_j)P(c_j)}{P(d)},$$

where:

- $P(d|c_j)$ is the probability of instance d being generated, given that class c_j is true (the <u>*Likelihood*</u>);
- $P(c_j)$ is the probability of class c_j's occurrence (the <u>*Prior*</u>);
- $P(d)$ is the probability of observation d having occurred (the <u>*Evidence*</u>). This can usually be ignored, since it's the same for all classes.

Naïve Bayes Classifiers

- If we have more than one attribute besides height, Naïve Bayes Classifiers make the simplifying assumption that attributes are independent:

$$P(\boldsymbol{d}|c_j) = \prod_i P(d_i|c_j).$$

- This simplifying assumption of attribute independence is why they're called "Naïve."
- Naïve Bayes Classifiers are extremely easy to implement, relatively cheap to compute, and yield surprisingly good results.
- For this reason, NBCs are a good ***benchmark*** to compare other classifier models against.
- If you build a classifier that does not exceed the performance of an NBC, you're either doing something wrong, or your problem really has independent attributes, and you should stick with the NBC.
- Onto our case study…

Case Study: European Companies

- Your group wants to expand its trading footprint in European markets.
- As the ML expert, you're given data(*) on European companies and tasked with discovering trading strategies.
- Your data(*) consists of dozens of quarterly accounting and macro ratios plus a Price Dispersion the following quarter.
- Price Dispersion is a measure of price movement. You realize that if you can predict this metric, you can make a case for a trading strategy.
- (*) This is real data from Aswath Damodaran of the Stern School at NYU. He has a treasure trove of finance data that you may find useful. For this dataset, you can go to his site and search for Eurocompfirm12: http://people.stern.nyu.edu/adamodar/

Case Study (cont.): Data Intimacy

- The automated nature of ML is never an excuse for not becoming intimately familiar with your data. Quite the contrary, it’s a reason why you can get in trouble by detecting spurious patterns.
- The moral: It pays to know your data.
- You have 6,022 records corresponding to European companies in a variety of industry groups.
- You have 73 attributes ranging from quarterly ROEs to Betas.
- You’re interested in predicting the next quarter’s Price Dispersion or “*Hi-Lo Risk*”:

$$HiLoRisk = \frac{Hi - Lo}{Hi + Lo},$$

where Hi and Lo denote the following quarter’s high and low stock prices, respectively.

Case Study (cont.): Data Intimacy

- This are the data fields (attributes, variables):

Industry Group	Stock price (Dec 31, 2012)in US$	Net Profit Margin
Country	Beta	Pre-tax Operating Margin
Broad Group	Correlation with market	Effective Tax Rate
Sub Group	Standard deviation in stock price	% held by institutions
Bottom up Beta for sector	Interest coverage ratio	Net Income
Bottom up levered beta	Current PE	Trailing Net Income
ERP for Country	Trailing PE	Trailing Operating Income
Cost of equity in US$	Forward PE	Revenues
Total Default spread (Company + Country)	PEG	Trailing Revenues
Pre-tax cost of debt in US $	PBV	EBITDA
After-tax cost of debt in US $	PS	Trailing EBITDA
Cost of capital in US$	EV/EBIT	EBIT (1-t)
ROE - Cost of Equity	EV/EBITDA	Net Debt issued (Debt issue - repaid)
ROIC - Cost of Capital	EV/Invested Capital	Change in non-cash Working capital
Market Cap (in US $)	EV/Sales	Net Cap Ex
Total Debt (in US $)	Payout ratio	Reinvestment Rate
Firm Value (in US $)	Dividend Yield	FCFF
Cash	Historical growth in Net Income - Last 3 years	FCFE
Enterprise Value (in US $)	Historical growth in Net Income - Last 5 years	Book Value of Equity - 4 qtrs ago
Cash/ Firm Value	Historical growth in Revenues - Last 3 years	Invested Capital - 4 qtre ago
Liquidity Ratio (Daily trading volume/Shrs outs)	Historical growth in Revenues - Last 5 years	Current Book Value of Equity
Book Debt to capital ratio	Expected growth rate in EPS- Next 5 years	Current Invested Capital
Market Debt to capital ratio	Expected growth in revenues - Next 2 years	Dividends
Book Debt to Equity Ratio	Return on Equity	HiL0 Risk Measure (Hi- lo)/ (Hi+Lo)
Market Debt to Equity ratio	Return on Capital (ROC or ROIC)	

- [...]

Case Study (cont.): Data Intimacy

- You notice that the data set has many null or missing data values (with “N/A” or “NaN” entries): 14.8% of the entries have null values.
- In fact, most of the records (5,446 out of 6,022) have at least one null entry. If you were to discard these records, your data size would shrink down to 576 records from the original 6,022 records.
- Likewise, almost half of the attributes have null values (36 out of the 73 original attributes).
- Clearly, discarding records would greatly degrade your ability to build a robust model.
- Enter: Data Imputation…

Case Study (cont.): Data Imputation

- Data Imputation is a process for “imputing” (*i.e.*, filling in) the missing values in a way that, hopefully, does not significantly bias your data set.
- There are many ways of imputing data:
  - Non-null Attribute Mean (or Median) Imputation.
  - Hot-Deck Imputation
  - KNN Imputation
  - LSR Imputation
  - SVD Imputation, …
- With Mean (Median) Imputation, we simply compute the mean (median) of the non-null entries of a particular attribute and use it to fill that attribute’s null entries.
- With Hot-Deck Imputation, we find the reference record that is “closest” to the record of interest whose null values we that we want to impute, and simply fill the missing entries by copying the reference record’s corresponding values.
  - Here “closeness” can be defined, for example, using Euclidean Distance or Cosine Similarity between the records’ non-null dimensions. *E.g.*:, Euclidean Distance is:

$$Eucl\ Dist_{ij} = \left[\sum_{\{k_i\}} \left(\mathcal{R}_i^{k_i} - \mathcal{R}_j^{k_i}\right)^2\right]^{1/2}$$

where the sum runs over the $\{k_i\}$ non-null entries of the record of interest, $\mathcal{R}_i$, and where the reference record, $\mathcal{R}_j$, has no null entries.

Case Study (cont.): Data Imputation

- Likewise, the Cosine Similarity can be written as:

$$Cos\ Sim_{ij} = \frac{\sum_{\{k_i\}} \mathcal{R}_i^{k_i} \mathcal{R}_j^{k_i}}{\left[\sum_{\{k_i\}} \left(\mathcal{R}_i^{k_i}\right)^2 \sum_{\{k_i\}} \left(\mathcal{R}_j^{k_i}\right)^2\right]^{1/2}}.$$

- KNN (K Nearest Neighbors) Imputation is a generalization of Hot-Deck Imputation where the imputed values are obtained from a similarity-weighted average of the K nearest reference records instead of from just the nearest one. Note that K is an ad hoc parameter.
- LSR (Least-Squares Regression) Imputation involves regressing the reference records' entries corresponding to the missing values of the record of interest against the reference records' entries corresponding to the non-missing values of the record of interest. In other words:
  - For each record of interest $\mathcal{R}_i$ whose missing entries we're trying to impute, we identify the set of K_i attributes corresponding to $\mathcal{R}_i$'s missing entries, and the set of L_i attributes corresponding to $\mathcal{R}_i$'s non-null (non-missing) entries.
  - Assuming that there are a total of J complete records, $\mathcal{R}_j$, in our database, we form the $J \times L_i$ matrix X^i by including the complete records' set of L_i attributes corresponding to $\mathcal{R}_i$'s non-null (non-missing) entries.
  - Next, we form the $J \times K_i$ matrix Y^i by including the complete records' set of K_i attributes corresponding to $\mathcal{R}_i$'s missing entries.
  - We thus impute $\mathcal{R}_i$'s missing entries, $\hat{Y}_*^i$, by regressing onto $\mathcal{R}_i$'s non-missing entries X_*^i:

$$\hat{Y}_*^i = X_*^i \left({X^i}^T X^i\right)^{-1} {X^i}^T Y^i\ .$$

Case Study (cont.): Data Imputation

- SVD (Singular Value Decomposition) Imputation involves repeated applications of SVD as follows:
  - First, we impute all null values of the original data matrix $\mathcal{D}^0$ using another method of imputation, typically Mean Imputation, to create our initial imputed data matrix $\mathcal{D}^1$.
  - Next, we compute the SVD of $\mathcal{D}^1$ and reconstruct it by keeping r singular values such that the normalized cumulative sum of singular values does not exceed a given fraction Δ (usually close to 1):

$$\mathcal{D}^1 = U^1 S^1 {V^1}^T,$$

$$\left\{ r\text{:}\ \frac{\sum_{k=1}^{r} s_k}{\sum_k s_k} \leq \Delta \right\},$$

$$\mathcal{D}_r^1 = U_r^1 S_r^1 {V_r^1}^T.$$

  - Subsequently, we construct the next iteration of the data matrix, $\mathcal{D}^2$ by keeping the original non-null (non-missing) entries of $\mathcal{D}^0$, but substituting $\mathcal{D}^0$'s null entries by the corresponding entries in $\mathcal{D}_r^1$:

$$\mathcal{D}^2 = \mathcal{D}^0(NonNull\ Entries) + \mathcal{D}_r^1(Imputed\ Entries).$$

  - We continue this process until the norm doesn't change by more than some small value δ:

$$Stop\ when\text{:}\ \frac{\left\| \mathcal{D}^{k+1} - \mathcal{D}^k \right\|}{\left\| \mathcal{D}^k \right\|} \leq \delta,$$

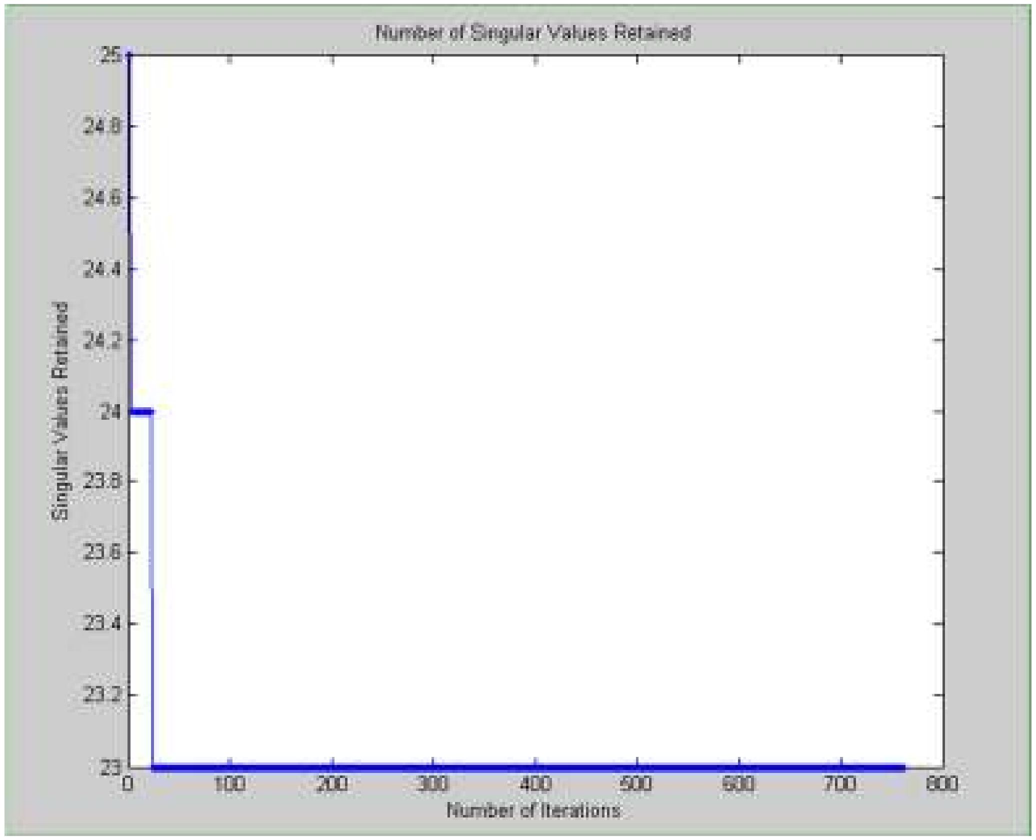
where the norm can be a 2-norm such as the Euclidean or Frobenius norm, for example.

Case Study (cont.): Data Imputation

- To evaluate imputation schemes, we can start with a complete data corpus of interest (without null entries), artificially delete entries, and see how "close" the imputed values are to the artificially-introduced nulls.
- The Mean (Median) Imputation is not very accurate, but it's easy and cheap to implement even on large data corpuses. It is easily parallelizable as well (SIMD).
- Hot-Deck Imputation is commonly used, as it is fairly accurate, relatively cheap to implement, and parallelizable (SIMD).
- KNN and LSR are more accurate than Hot-Deck and give good results, but are more expensive to implement, although they are parallelizable.
- Moreover, Hot-Deck, KNN, and LSR only draw on *part* of the data (the subset of complete records), which can be significantly smaller than the original data set. In our example, there are 576 complete records out of the original 6,022—a small fraction of the original data set. This could significantly bias our data.
- SVD Imputation gives reasonably good results and it has the added advantage of using all of the available data. However, it is expensive to implement, and is suitable only for small to medium-sized data sets. It is not easily parallelizable due to the relative norm computation at each iteration (although the SVD computation itself is parallelizable).
- A drawback of SVD Imputation is its reliance on the ad-hoc parameters Δ and δ. Another drawback is the fact that it is not self-starting, relying on another imputation method to bootstrap the first iteration.
- The SVD Imputation scheme above is my own concoction, and we will be using it in our relatively small data example. Implementation times on our data set were of the order of several minutes on a laptop.
- There are other SVD Imputation schemes, and other methods as well (Bayesian, local-neighborhood methods, etc…).

Case Study (cont.): Data Imputation

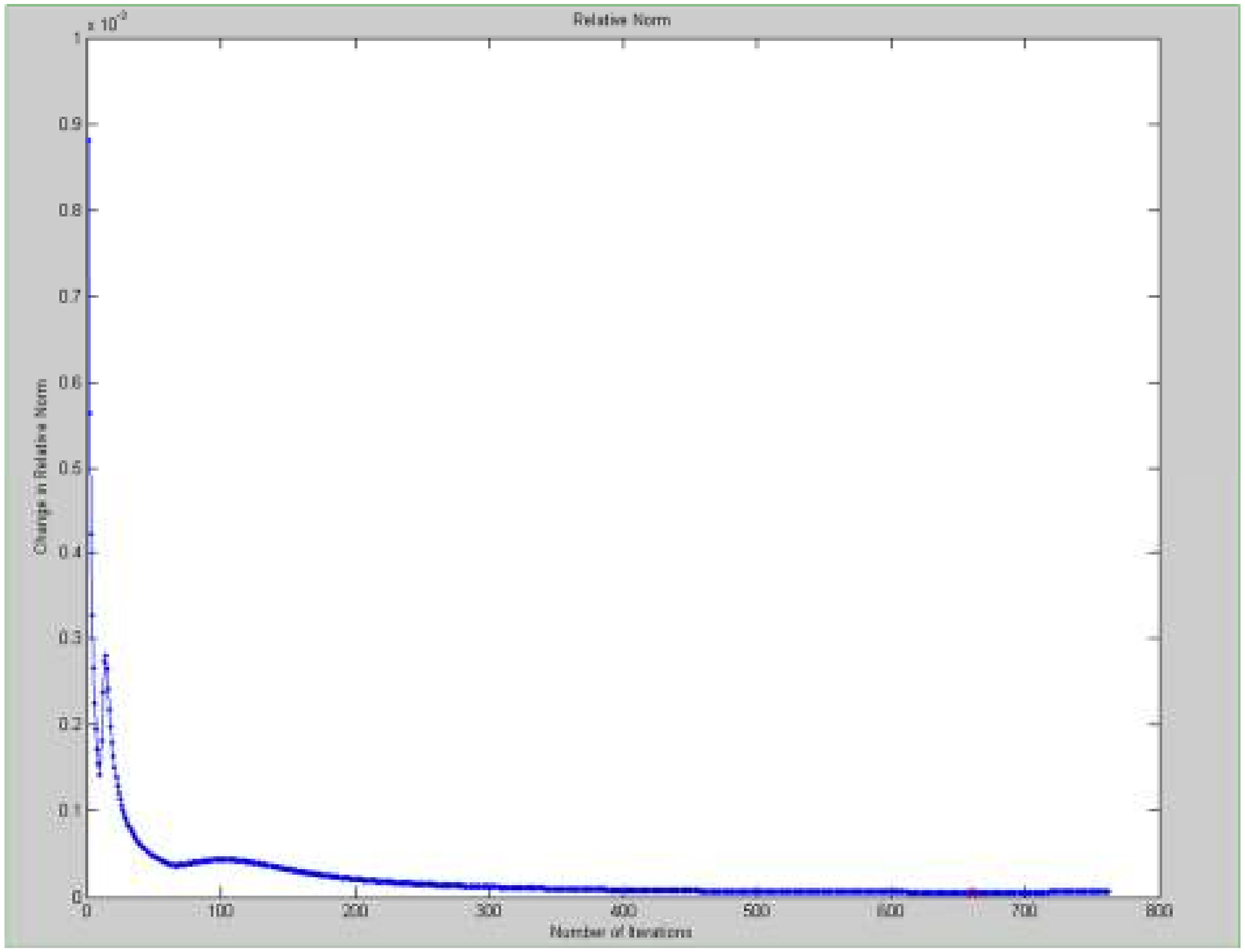
- The SVD Imputation scheme took 1-2 seconds per iteration on our data set using a laptop and ran for about 770 iterations:



- 23 out of a maximum of 69 singular values were retained, using a $\Delta = 0.995$. Note that, at this stage, we're interested in retaining as much of data integrity as possible, and not in filtering out noise from the data. That's why we chose such a high Δ.

Case Study (cont.): Data Imputation

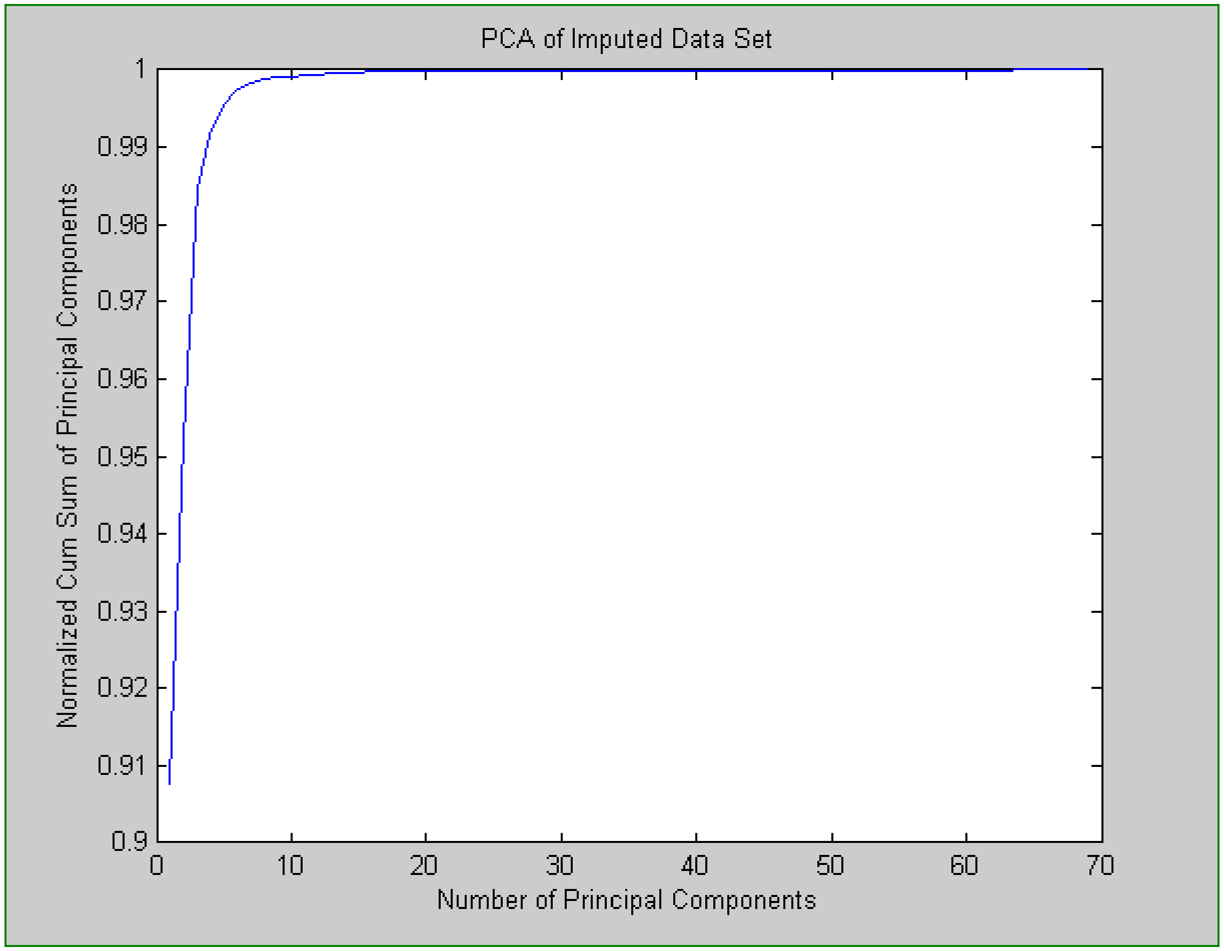
- After initial transient behavior, the relative norm settled to a smooth behavior, and reached a minimum at around 660 iterations (red circle):



- Note that the choice of δ depends on the data size and the number of null entries, and it is arrived at through trial and error.

Case Study (cont.)

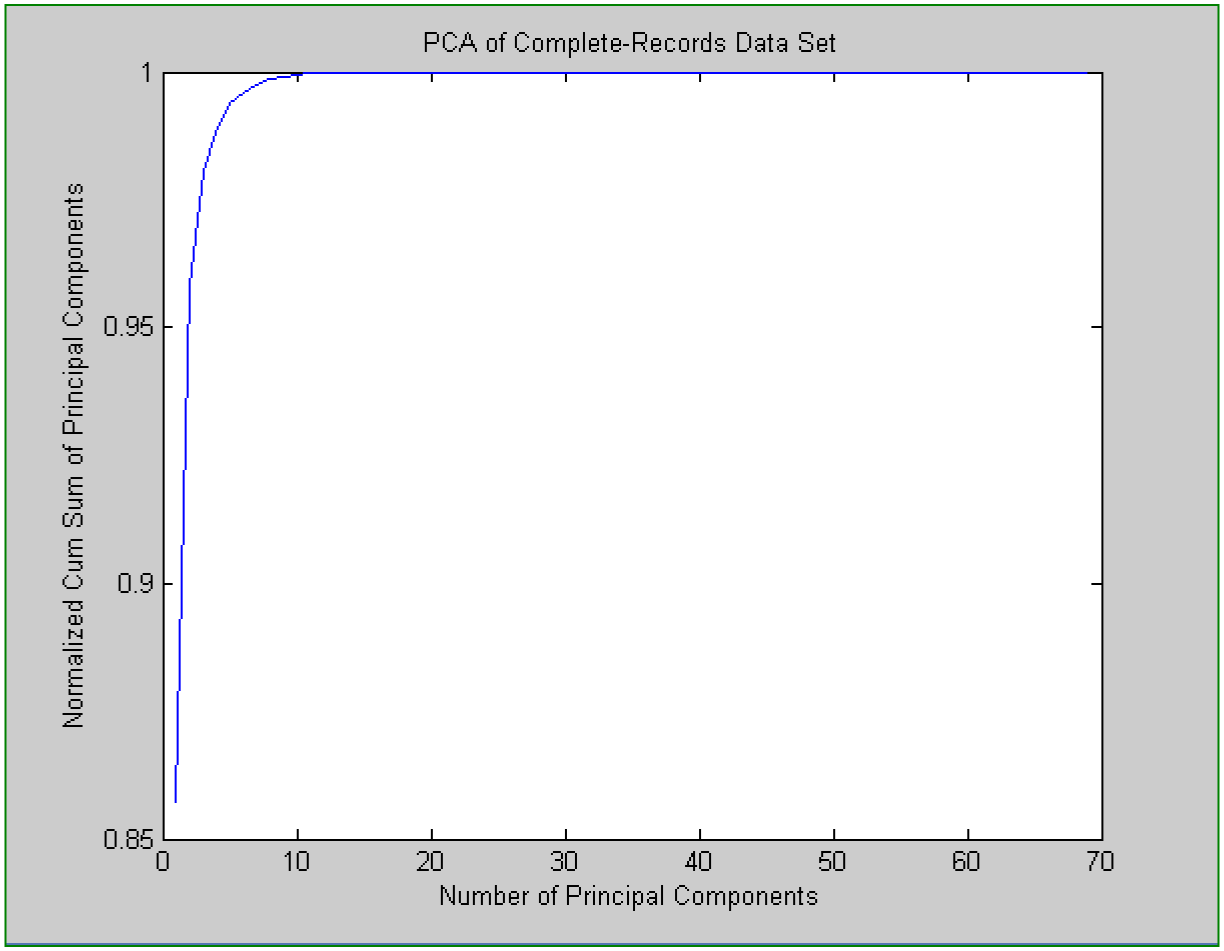
- Now that we have our imputed data set, $\widehat{\mathcal{D}}$, we perform a PCA analysis:



- There is a fair amount of collinearity in the data. Keeping about 20 out of 69 Principal Components retains most of the variability (this is nor surprising given that the number of singular values retained during SVD Imputation was 23).

Case Study (cont.)

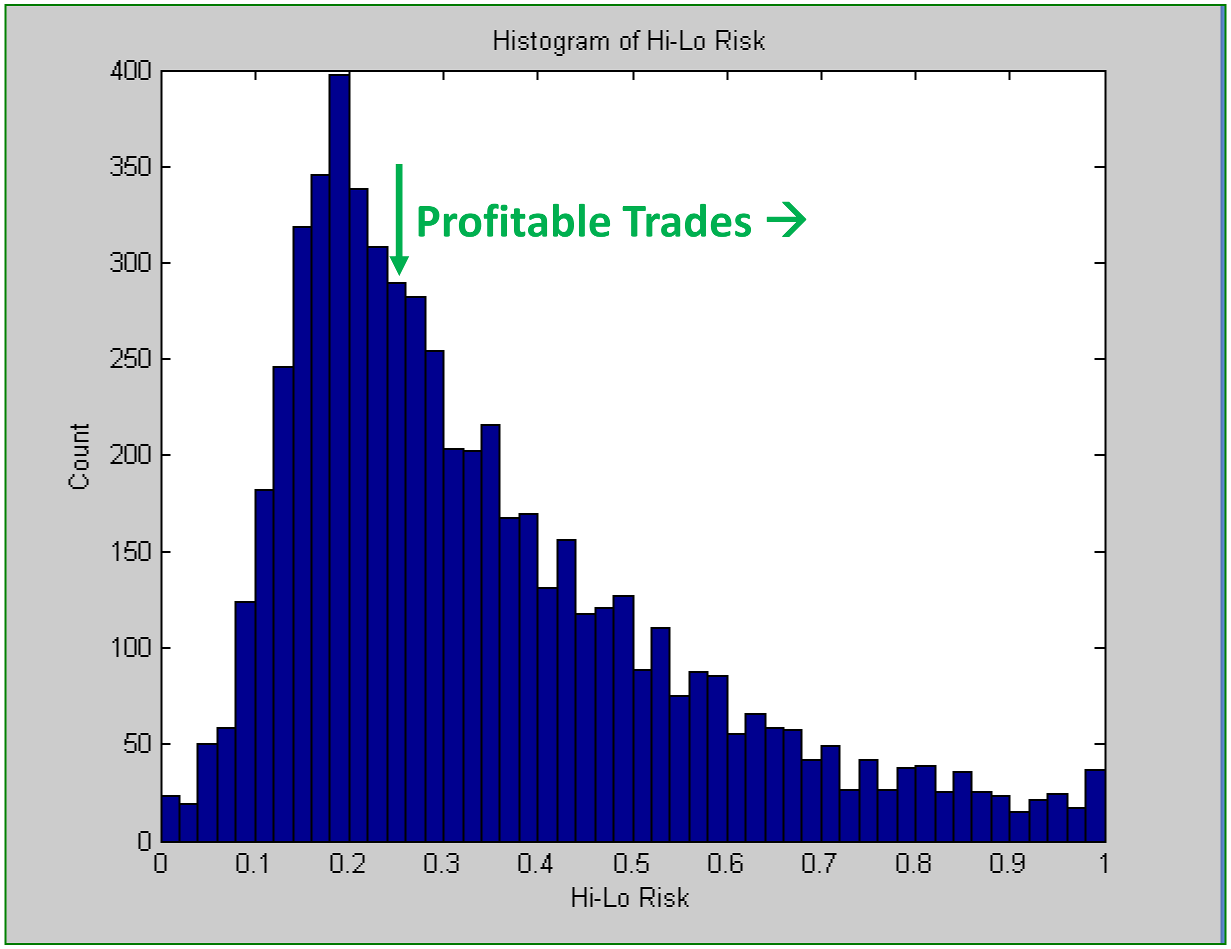
- For comparison, we perform a PCA analysis of the much smaller complete-records data set:



- This has a similar PCA profile to $\widehat{\mathcal{D}}$'s, although variability saturation is reached a bit sooner. This is again not surprising given that $\widehat{\mathcal{D}}$ has a great deal more data and somewhat more variability is expected in $\widehat{\mathcal{D}}$'s case. Had we noticed very different PCA profiles, this would've been cause for concern vis à vis our imputation process.

Case Study (cont.)

- Next let’s take a look at our target variable, $HiLoRisk = (Hi - Lo)/(Hi + Lo)$:



- You calculate that price movements of around 25% or higher ($HiLoRisk \gtrsim 0.25$) would lead to profitable trades (why?), if they could be correctly predicted.
- This suggests a classification problem of Profitable/Not Profitable trades based on a $HiLoRisk$ threshold of 25%.

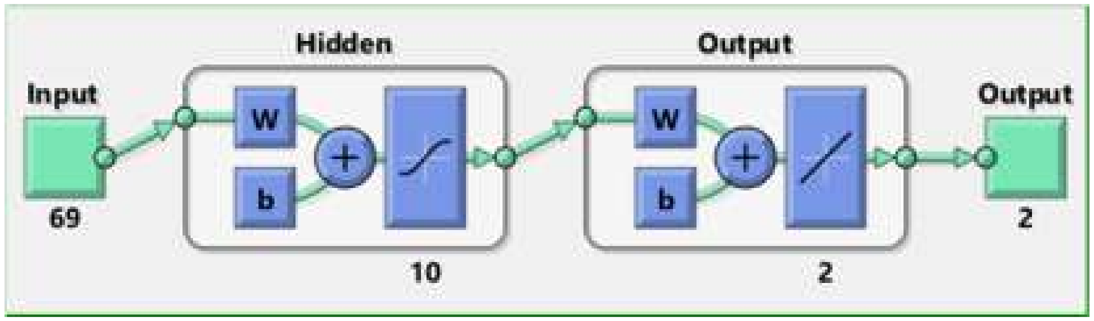
Case Study (cont.)

Some Observations:

- Should we have included the target variable $HiLoRisk$ in the imputation process?
- No!! This would be circular and could lead to “peeking.”
- Null entries in the target variable(s) should be removed, not imputed, as this does not gain us anything and can lead to peeking and thus incorrect prediction.
- In our example, there were only two null entries in the target variable, reducing the number of records from 6,022 to 6,020.
- Of the original 73 attributes, I excluded categorical variables such as *Industry Group*, *Country*, *Broad Group*, and *Sub Group*, ending up with 69 attributes.
- There’s no reason to exclude these attributes, as they could carry useful information. They could be binarized (as we saw in the data representation section of the previous lecture). I chose to exclude these attributes for expediency.

Case Study (cont.)

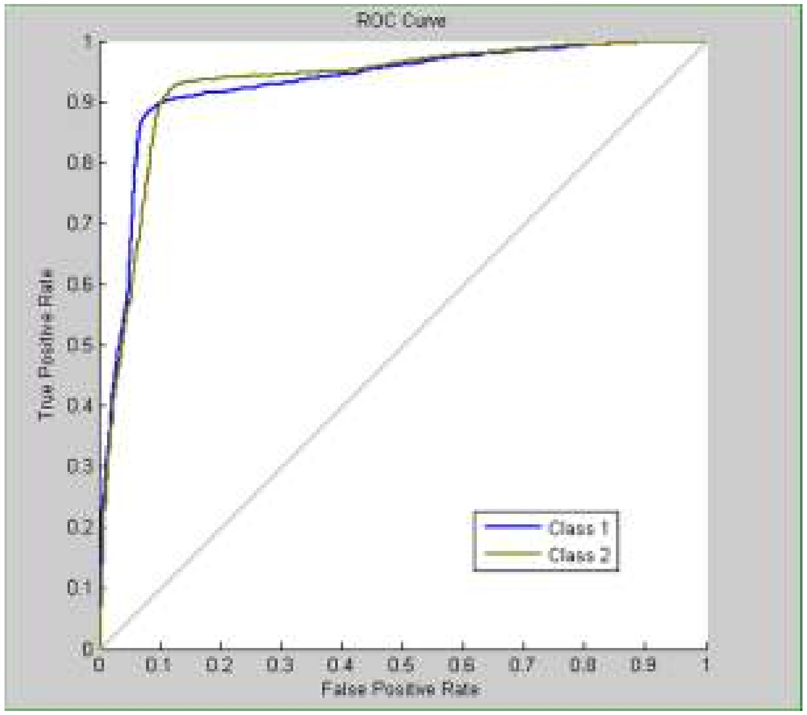
- Now that we've got a handle on our data and a framework for elucidating a strategy, we proceed to binarize the target variable $HiLoRisk$ with a threshold of 0.25, and build a Classifier Neural Network with the following architecture:

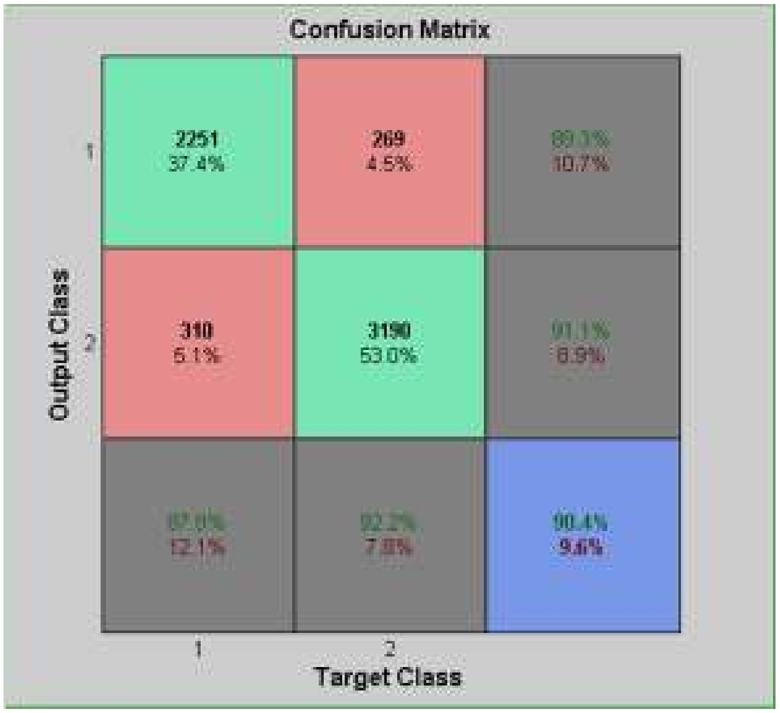


- There are 69 Input Variables (Attributes), 10 Sigmoidal Hidden Nodes and 2 *purelin* output nodes representing the binarized class outputs where the class "*Unprofitable Trades*" is represented with the output $[0,1]$ corresponding to $HiLoRisk < 0.25$ and the class "*Profitable Trades*" is represented with the output $[1,0]$ corresponding to $HiLoRisk \geq 0.25$.
- We proceed with training using Bayesian Regularization and Early Stopping to avoid overfitting…

Case Study (cont.)

- The trained classifier produces the following results:





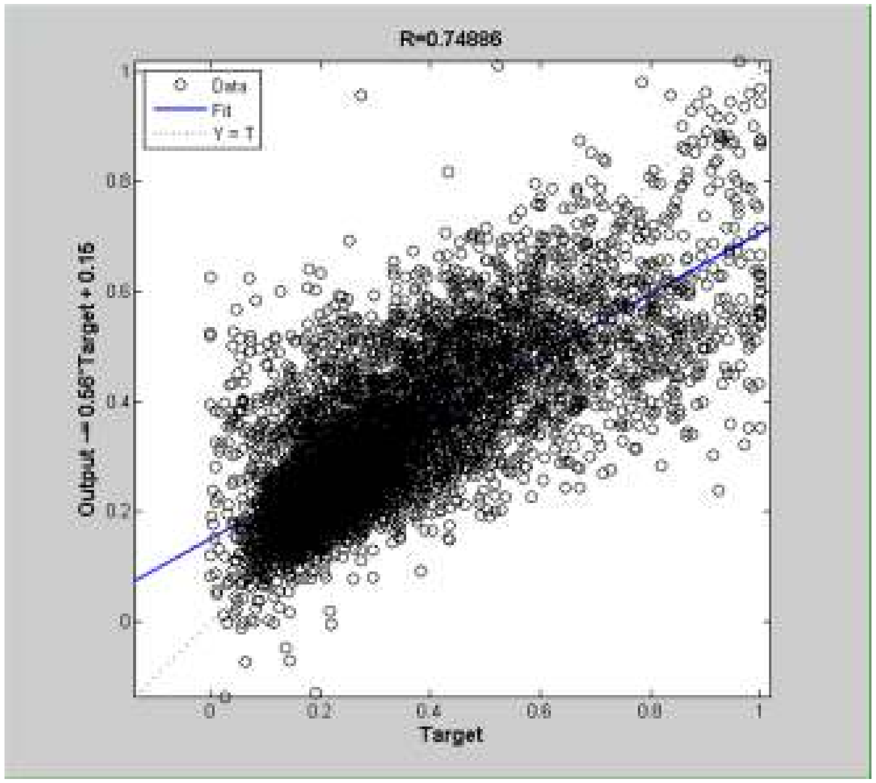
Confusion Matrix

Output Class \ Target Class	1	2	
1	2251 37.4%	269 4.5%	89.3% 10.7%
2	310 5.1%	3190 53.0%	91.1% 8.9%
	87.9% 12.1%	92.2% 7.8%	90.4% 9.6%

- These results look promising: You’re able to classify 90% of the trades correctly!

Case Study (cont.)

- Emboldened by these classification results, you decide to explore whether you can predict the <u>*continuous*</u> $HiLoRisk$ target variable.
- You build an FFNN with 69 inputs, 10 Sigmoidal Hidden Nodes, and a single *purelin* output to represent the continuous $HiLoRisk$ target. You train using Bayesian Regularization with Early Stopping and obtain:



- Again, these are promising results with correlations in the 75% range (R^2's around 56%).

Case Study (cont.)

- With results like these, you could be in the “kill zone.” However, before you start dreaming of a big bonus, pats on the back, and smiles all around, your homework is not nearly done yet.
- While it appears that you’re able to predict next-quarter price moves, you still need to predict their *direction*.
- You need to worry about whether the signal you’ve found might be untradeable due to a variety of market frictions idiosyncratic to each of the individual countries whose companies you’ve considered.
- You need to perform a sensitivity analysis to pick out salient attributes and streamline your model to make it more parsimonious.
- You need to account for portfolio risk while trading all these names as a portfolio.
- You need to assess how this potential new strategy is complementary to and diversifies your group’s current strategies.
- You need to be able to present your results in a cogent manner to a management who may or may not be familiar with the techniques you’ve used (you may face the “black box stigma” that comes along with ANNs), etc.
- However, all in all, this is a good starting point for a feasibility study.
- Back to our regularly scheduled program…

Multidimensional Classifier Performance

- What happens when we have more than 2 classes?
- Extending ROC curves to 3 or more classes is not trivial.
- Confusion matrices are still useful.
- However, objective measures for confusion matrices are not as clear. (Project Idea: Re-arrange confusion matrix so that errors are off-diagonal and measure matrix “diagonality;” hint: size of confusion matrix’s Gershgorin disks (?), …)
- Incorporating asymmetric costs among several classes is difficult…
- Let’s move on to a new topic.

Unsupervised Learning Paradigm

- So far we've looked at Supervised Learning (*e.g.* Function Approximation, Classification) by examining a model in some depth (Neural Networks).
- Next, we will consider <u>*Unsupervised Learning*</u>.
- Examples of Unsupervised Learning: Clustering, Association Rules.
- We will consider Unsupervised Learning by examining another model in some depth: <u>*Association Rules*</u>.

New Topic: Association Rules

- *Association Rules* (also known as Market Basket Analysis) are an example of Unsupervised Learning.
- They're also an example of a Non-Parametric technique.
- 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 purchases vitamin supplements then calcium supplements are often also purchased."*
- The idea is to discover these Association Rules *Automatically.*
- Rules will be valuable to the extent that they are unexpected (Example: beer & diapers purchased together at night).
- Problem: The Number of possible items (occurrences) can be very large, and the number of transactions can be huge (talk about Big Data).

Association Rules

- More formally, Let $\mathcal{I} = \{i_1, i_2, \dots, i_n\}$ be a set of literals called ***Generalized Items***. 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 ***Database*** 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 ***Itemset***.

The Database of Transactions

$$\mathcal{D}\quad \begin{array}{c|ccccccc|}
 & i_1 & i_2 & i_3 & i_4 & \cdots & i_{n-1} & i_n \\
\hline
d_1 & 0 & 1 & 1 & 0 & \cdots & 0 & 1 \\
d_2 & 0 & 0 & 1 & 0 & \cdots & 1 & 0 \\
d_3 & 1 & 0 & 1 & 0 & \cdots & 0 & 0 \\
d_4 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\
\vdots & & & & \vdots & & & \\
d_{m-1} & 1 & 1 & 0 & 1 & \cdots & 0 & 1 \\
d_m & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\
\hline
\end{array}$$

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 <u>*Association Rule*</u> "$X \Rightarrow Y$" 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).

Association Rules

- In Association Rules of the form $X \Rightarrow Y$ we usually refer to X as the "*Antecedent*" and Y as the "*Consequent*."
- Notice that this distinction is *artificial* when dealing with *temporal concurrency*.
- E.g.: $X =$ "*A customer is female and she buys multi-vitamins,*" $Y =$ "*The customer buys calcium supplements*."
  - It is unclear which action causes which.
  - It is unclear which is the antecedent and which the consequent.
- E.g.: $X =$ "*A customer is female and she buys calcium supplements,*" $Y =$ "*The customer buys multi-vitamins*."
- In practice, the consequent is taken to be a hypothesis that leads to an *actionable policy* (an "*actionable*"), such as: "*Place calcium supplements near feminine products.*"
- Usually, we cannot infer causality from concurrency measurements.

Association Rules

- In HFT, we construct consequents by introducing time delays.
- E.g.: $X =$ "*INTC price went up and OIH went down this second.*" $Y =$ "*One second later, AMAT goes up*."
- It is clear that X cannot be the consequent of Y since X happens before Y. So Y is the clear consequent candidate.
- We still cannot infer causality because there could be a common cause, etc.
- In practice, we care about the ***repeatability*** of temporal patterns, regardless of the true causal connection (the "story" behind the pattern).
- Hypothesis: "*AMAT price will go up by more than transaction costs in the next second (or relevant time period)*."
- Actionable: "*Buy AMAT now*."

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 (as found in the literature) 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, defined as follows…
- *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 intractable 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.

Tractability

- Another approach is to find antecedents to interesting consequents.
- Works well in HFT because interesting consequents are uncommon.
- Can be used with the other pruning techniques.

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 *<u>Non-Random Associations</u>*, 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 *<u>By Random Chance Alone</u>* 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 that Support/Confidence would throw away this potentially useful information.

4. We'll fix all this in the next lecture…

Questions/Comments