ECON484 Machine Learning

1. Concepts of Machine Learning (2/2)

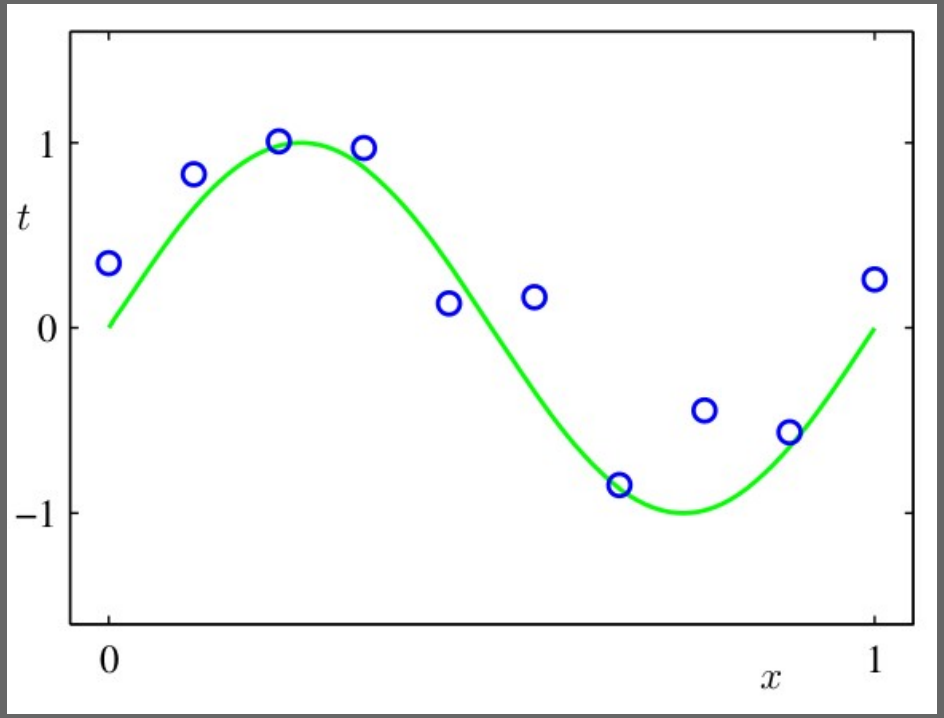
Lecturer:
Bora GÜNGÖREN
bora.gungoren@atilim.edu.tr

An Applied Example

- We will look at the basic problem of polynomial curve fitting (adapted from Bishop 2006).
  - This will involve optimizing the parameters (polynomial coefficients) ad be classified as an estimation problem.
  -

An Applied Example

- Our actual function is a sine wave $t=\sin(2\pi x)$.
  - We are given a training set $X=(x_1, x_2, \ldots, x_N)^T$
  - The data set has generated data and some (Gaussian) noise.



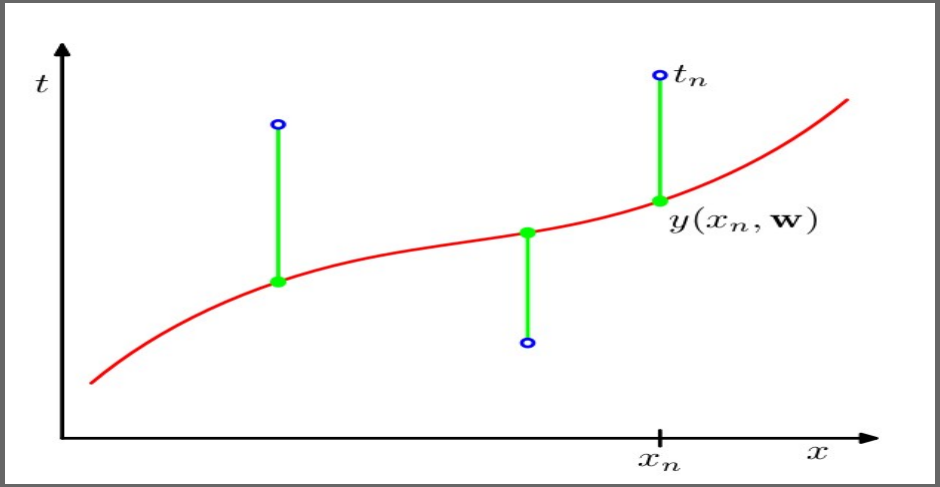
An Applied Example

- We should have a model and the model parameters.
  - $w=(w_0, w_1, w_2, \ldots w_M)$ are the model parameters where M is the dimension of the polynomial.

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \qquad (1.1)$$

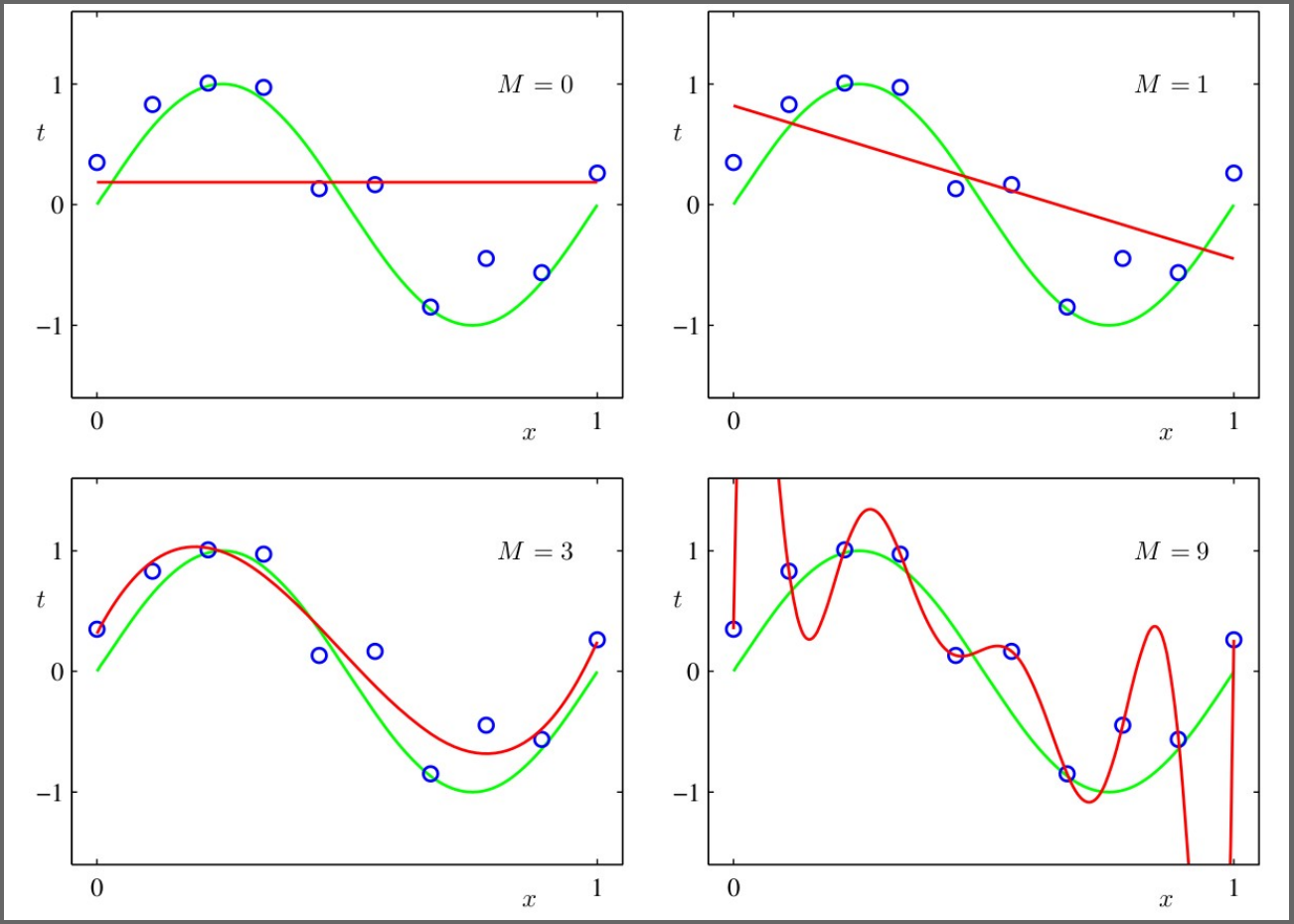
An Applied Example

- Our parameter estimation problem is now expressed as “estimate **w** (as vector) to minimize”.
  - Error is defined as E(w) which is calculated using estimated y_N and actual t_N in the training set.



$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 \qquad (1.2)$$

An Applied Example



An Applied Example

- M=3 has more error in the training data set than M=9, however it will give a better overall fit for the curve.
  - This is why actual performance should be measured with practical application.
  - The problem with M=9 is what we call **over-fitting**.

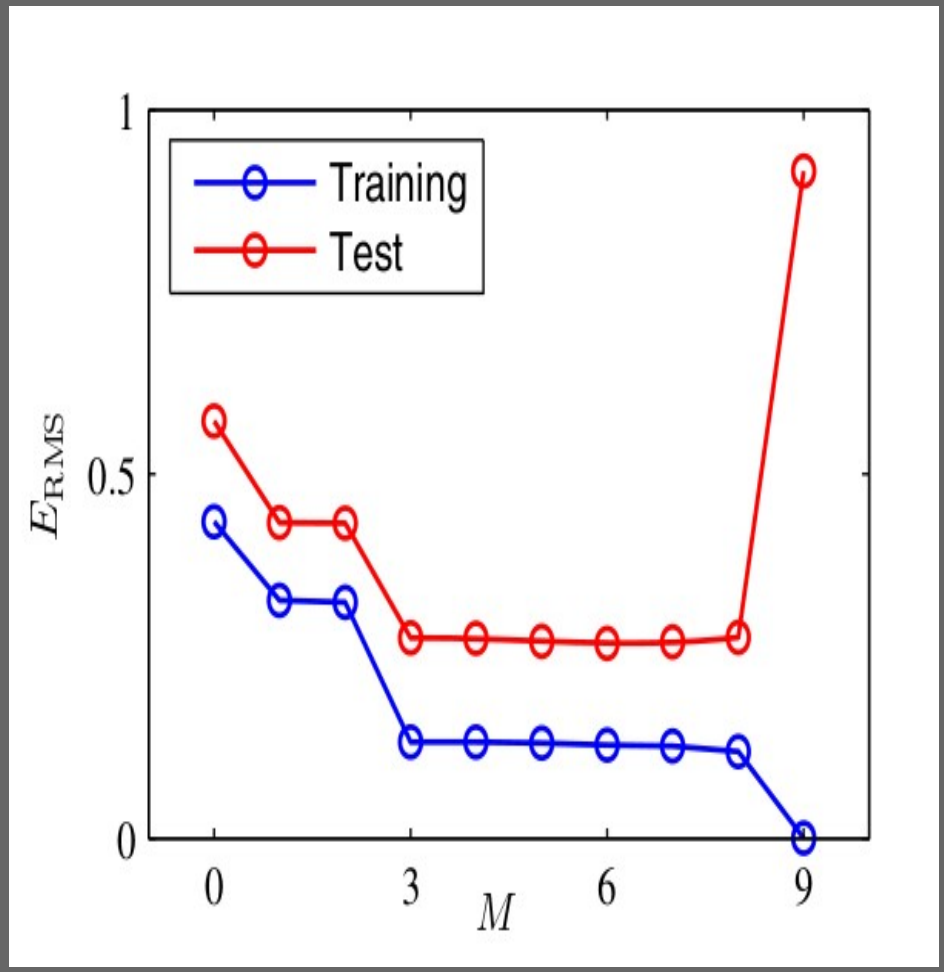
An Applied Example

- To measure performance, we generate a new test data set (again by adding noise) of size 100.
  - Then we define the error measure as E_{RMS}.
  - Let's calculate E_{RMS}

$$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N} \qquad (1.3)$$

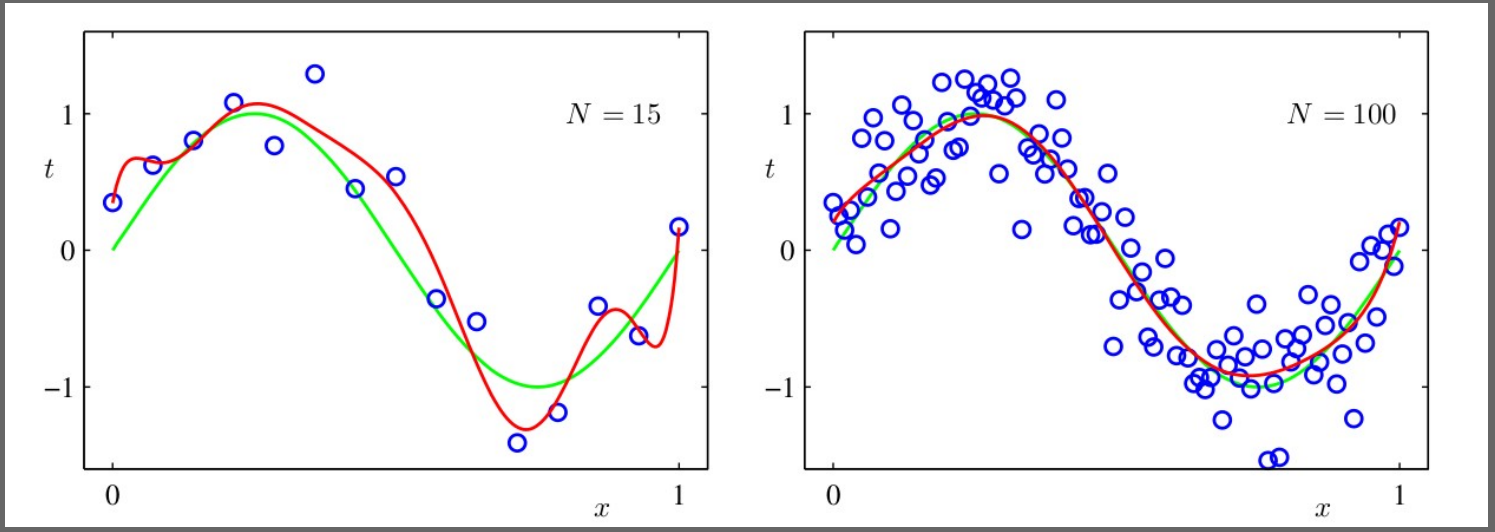
An Applied Example

- What we see
  - In the overtrained model, E_{RMS} increases wildly with test data set.
  - From M=3 to M=8 the error looks minimal.
  - So we keep the simplest form, M=3.



An Applied Example

- A way to reduce the error with M=9 model would be to train with a larger training data set.



An Applied Example

- It seems that having a larger training data set solves some problems.
  - But it is not practical to assume we have a larger training data set in all practical settings.
  - One approach is to retrain our data models frequently.
  - Another approach is to employ different models.
- Model complexity should reflect the complexity of the real world.

An Applied Example

- In the example case, the reason for the large fluctuation of estimated values is the very large coefficients in the model we obtain after training.
  - The better models include smaller coefficients.
  - So why not penalize very large coefficients in the model?

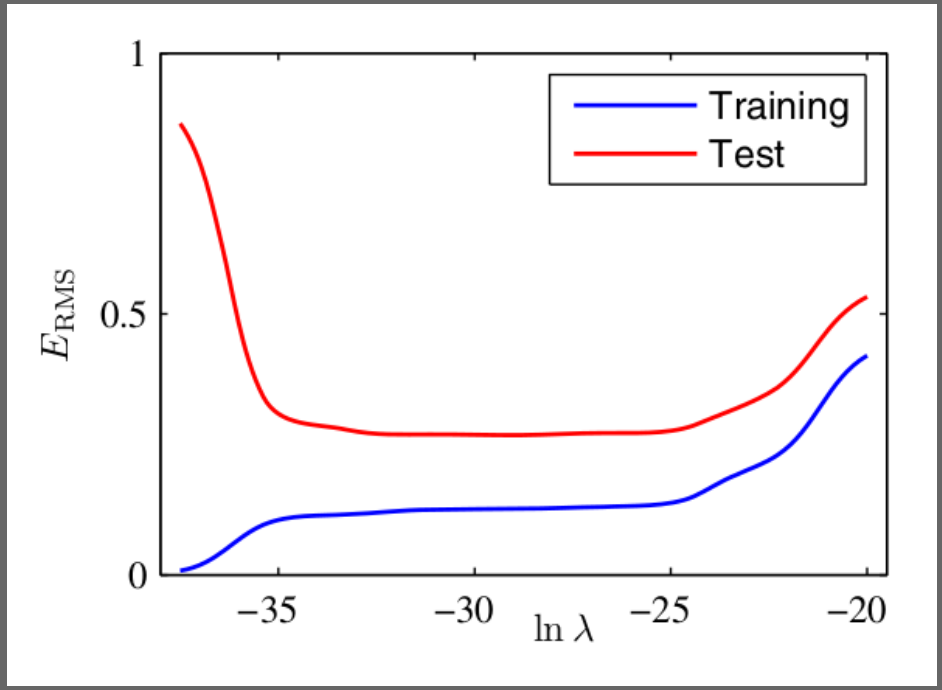
An Applied Example

- We penalize very large coefficients through a modified error definition.
  - This modification type is called a **shrinkage** (or **weight decay**), and this particular one is called a **ridge regression**.

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \qquad (1.4)$$

An Applied Example

- Our selection of model modification has added a new parameter to optimize
  - **Lambda** which symbolized how important the shrinkage is.
  - We should also select this parameter.



An Applied Example

- Another approach to the same estimation problem would be Bayesian.
  - Let our training data $D=(t_1, t_2, \ldots, t_N)$ represent a number of observations.
  - Let us assume our parameter set **w** is now probabilistic with a probability distribution **p(w)**.
  - We try define all quantities as a function of **w**.

An Applied Example

- With p(D|w) as the likelihood function:

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} \tag{1.43}$$

$$p(\mathcal{D}) = \int p(\mathcal{D}|\mathbf{w})p(\mathbf{w})\,\mathrm{d}\mathbf{w}. \tag{1.45}$$

An Applied Example

- With the frequentist school of thought, we usually use a maximum likelihood estimator, to maximize the likelihood function p(D|w).
  - The error is defined as the negative logarithm of likelihood function. As one increases the other should decrease.
  - However the fundamental difference is that we see **w** as a constant and discuss the variation of D given a **w**.

An Applied Example

- With the Bayesian school of thought, there is only the observations D, and we evaluate the uncertainty in w, after we observed D.
  - Therefore D is a constant, and there is a variation in w, which we discuss (and optimize).
- Also note that the likelihood function is not a probability distribution.
- The Bayesian approach requires a lot more computation than the frequentist approach, and is often infeasible to compute by hand.
  - But with computer resources at hand, it has become more popular.

An Applied Example

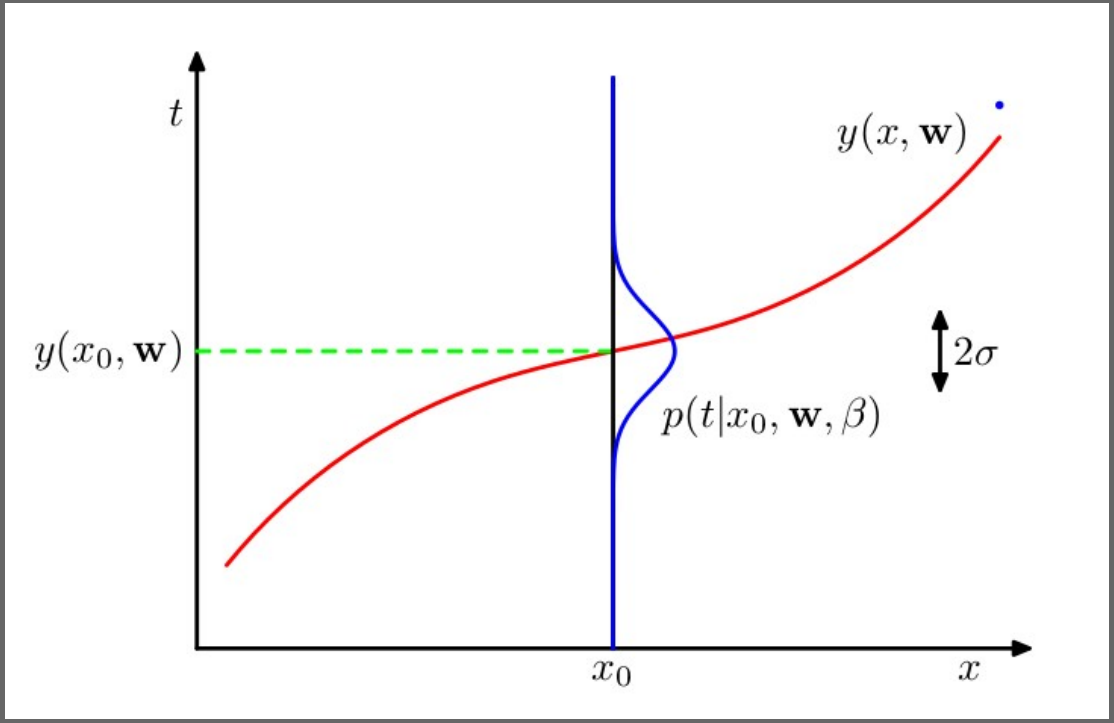
- So how to approach curve fitting from a Bayesian perspective?
  - We treat the observations $t=(t_1,t_2, .., t_N)^T$ as having some uncertainty with a selected probability density function (pdf). Our **model selection** is now based on our choice of PDF.
    - This is a **very big** issue. Frequentists often criticize Bayesian approach saying “selection of PDF after observation of training data is cheating”.
  - Let’s assume that the distribution of error is Gaussian (which we know to be true in this case).

An Applied Example

- With this model selection any observation t has now a conditional probability based on x with the parameters **w**, and beta.
  - We use the training data set and a **maximum likelihood estimator to find the parameter values**.
  - In order to minimize error, defined as negative logarithm of the maximum likelihood function, we will try to **maximize the logarithm of the maximum likelihood function**, with respect to beta.
  - Once we have the parameters, we can start making predictions.

An Applied Example

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}\left(t|y(x, \mathbf{w}), \beta^{-1}\right) \tag{1.60}$$



An Applied Example

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n | y(x_n, \mathbf{w}), \beta^{-1}\right). \tag{1.61}$$

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = -\frac{\beta}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi). \tag{1.62}$$

An Applied Example

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}_{\mathrm{ML}}) - t_n\}^2 . \qquad (1.63)$$

- Having determined w and beta, we can make predictions.
  - What we have is a predictive distribution p(t | x, w, beta) which gives **a probability distribution over t, not a point estimate.**
  - We could also assume w is probabilistic, and have **hyperparameters** on the distribution of w. For example **p(w|alpha)**.

An Applied Example

- When we assume w is probabilistic, we estimate the **most likely value of w given the observations D**.
  - This is equivalent to minimizing the error function to find w.
  - Note that in Bayesian approach we take integrals which is computationally expensive.
- In the maximum likelihood approach we have demonstrated the importance of overfitting, and having a large training data set.
- In the Bayesian approach we have demonstrated how complex it could become, and how much computation intensive our tasks can become.

How to work with a small training data set?

- This is an essential question.
  - When there is a large training data set, we select some for training and some for validation.
  - When there is a small training data set, we employ **cross-validation**.
    - In this approach, we have multiple runs, each time with a different part of the training set reserved for validation.
    - We develop descriptive statistics (ie. averages) from the performance of these runs and evaluate the performance based on these descriptives.

How to work with a small training data set?

- S-fold cross-validation
  - We create S (equal sized) parts of the training data set.
  - We train using the majority (white) and validate with the withheld (red) data set.
  - We take the average of performance values.
- Extreme case is to have S=N runs, which is also called **leave-one-out**.

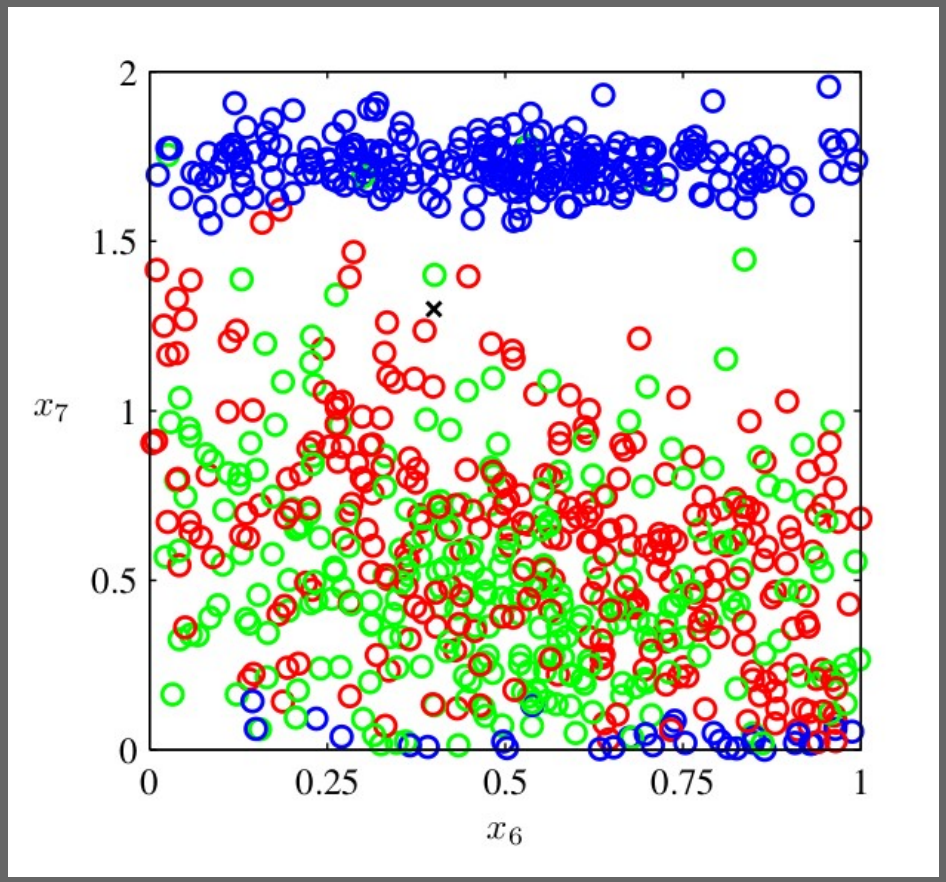
Dimensionality

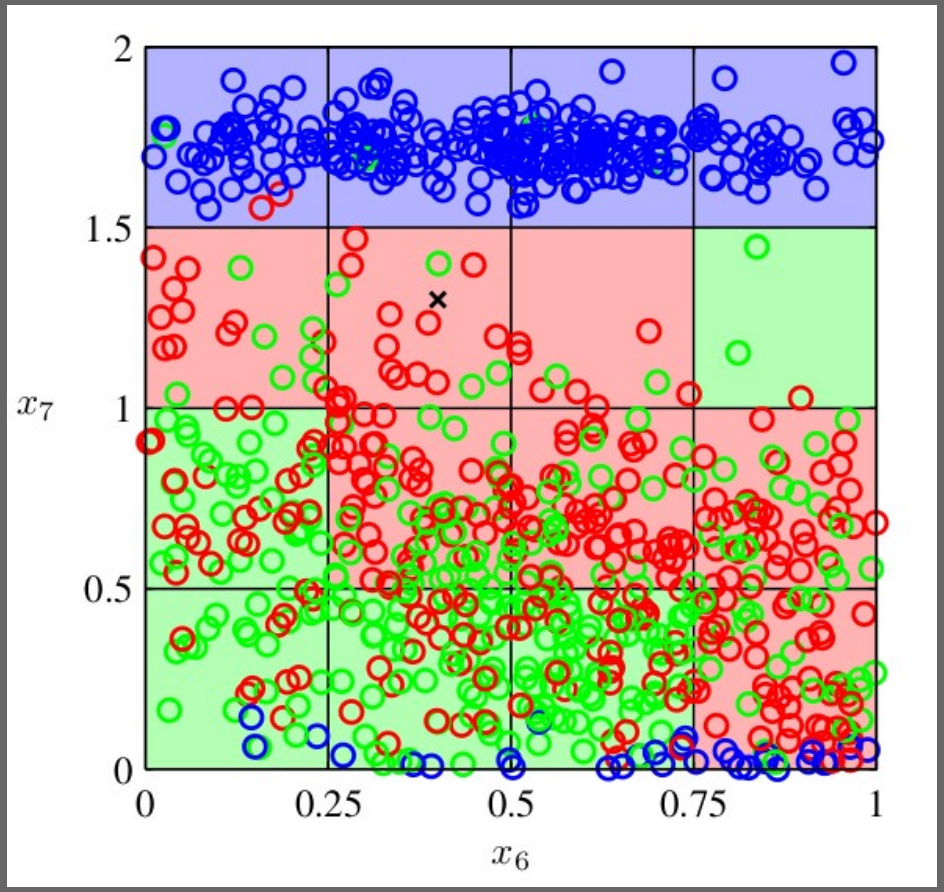
- In our simple example we discussed how we develop a model to **estimate a function of a single variable**.
  - When we have multiple variables (higher dimensionality) this becomes much more complex.

Dimensionality

- We will again use the example from Bishop 2006.
  - The example involves focusing on two variables (out of 12) in a classification problem.
  - The proposed technique divides the space into “zones” which belong to the majority class in the test data set.
  - If a point falls into a zone, then it is classified as from that class.

Dimensionality





Dimensionality

- This approach works “OK” for 3 classes and a 2-dimensional space.
  - How about more classes and the full 12-dimensional space?
  - How many zones do we have? 2^{12}? 3^{12}? 4^{12}?
  - How many coefficients will we need? 12^2? 12^3? 12^4?
- Naive approaches just don’t work with higher dimensions.
  - This is called the **curse of dimensionality** (Bellman, 1961).

Decision Theory and Machine Learning

- Decision theory proposed many methods to make decisions. They usually focus on **optimizing some parameters** in order to:
  - Minimizing false decisions
  - Minimizing expected losses
  - Detecting hard to make decisions

Decision Theory and Machine Learning

- Also consider von Neumann's minimax theorem and Yao's principle with multiple turn decision making in a competitive environment.
  - The observations **D** we have on the competitor's decisions under partially observed environment (test data set) can be used to learn about (ie. estimate a range on) the parameters (**w**), hence **the range of decisions provided by a particular decision making model** we assume she is using.
  - Model selection, parameters as probability distributions, observations as given facts...
  - These are the cornerstones of a Bayesian approach.
  - Bayesian Decision Theory uses machine learning in practical applications.

Model Evaluation Criteria

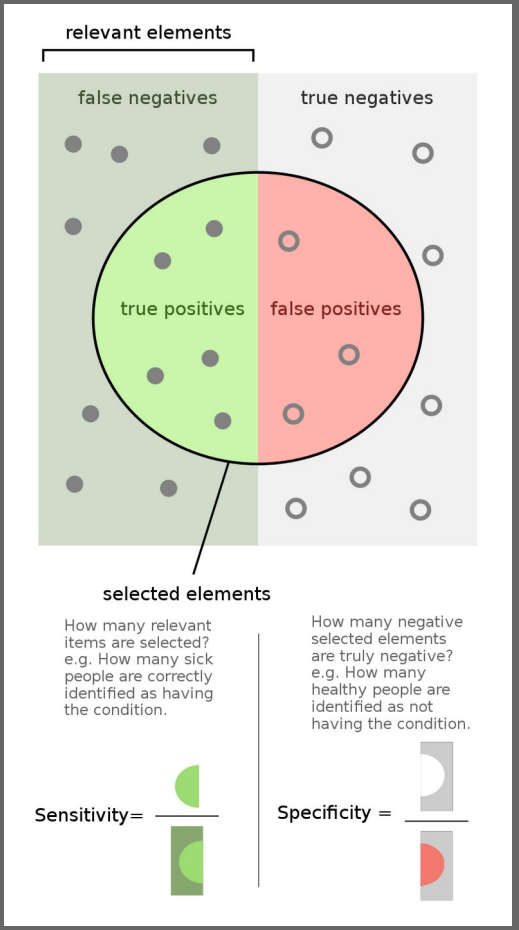
- As we see, we usually design our model prior to training it with the training data set, and then validate it with the validation data set.
  - We should be able to observe problems with the model before we use it in practice.
- Hence we need some model evaluation criteria.

Model Evaluation Criteria

- In classification problems we will have multiple type of problems.
  - The core performance measure, **classification accuracy** is the ratio of correct predictions (true positives and true negatives) to total predictions made. However in many cases, **error rate** as the rate of all other cases (false negatives, false positives) is also important.
  - In the two-class problem we define the true positive, false positive, true negative, false negative approach easily.

Model Evaluation Criteria

- Sensitivity (recall) and specificity are two important performance criteria we use. (Figure from is from Wikipedia).
  - High sensitivity is always good.
  - High specificity may be more important than high sensitivity because in many applications false positives cause hard to repair damages.



Model Evaluation Criteria

- A confusion matrix is a technique for summarizing the performance of a classification algorithm.
  - We have a series of classifications, actual and predicted and we see their true/false positive/negative nature.
  - We summarize using a matrix.
  - We calculate precision, accuracy and recall.

		Predicted	
		Negative (**N**) **-**	Positive (**P**) **+**
Actual	Negative **-**	True Negatives (T**N**)	False Positives (F**P**) **Type I error**
	Positive **+**	False Negatives (F**N**) **Type II error**	True Positives (T**P**)

Model Evaluation Criteria

		CONDITION determined by "Gold Standard"			
	TOTAL POPULATION	CONDITION POS	CONDITION NEG	PREVALENCE $\frac{\text{CONDITION POS}}{\text{TOTAL POPULATION}}$	
TEST OUT-COME	TEST POS	True Pos TP	*Type I Error* False Pos FP	*Precision* Pos Predictive Value $PPV = \frac{TP}{TEST\ P}$	False Discovery Rate $FDR = \frac{FP}{TEST\ P}$
	TEST NEG	*Type II Error* False Neg FN	True Neg TN	False Omission Rate $FOR = \frac{FN}{TEST\ N}$	Neg Predictive Value $NPV = \frac{TN}{TEST\ N}$
	ACCURACY ACC $ACC = \frac{TP + TN}{TOT\ POP}$	*Sensitivity (SN), Recall* Total Pos Rate TPR $TPR = \frac{TP}{CONDITION\ POS}$	*Fall-Out* False Pos Rate FPR $FPR = \frac{FP}{CONDITION\ NEG}$	Pos Likelihood Ratio LR + $LR+ = \frac{TPR}{FPR}$	Diagnostic Odds Ratio DOR $DOR = \frac{LR+}{LR-}$
		Miss Rate False Neg Rate FNR $FNR = \frac{FN}{CONDITION\ POS}$	*Specificity (SPC)* True Neg Rate TNR $TNR = \frac{TN}{CONDITION\ NEG}$	Neg Likelihood Ratio LR - $LR- = \frac{TNR}{FNR}$	

Model Evaluation Criteria

- When we have more than 2 classes, calculating TP, TN, etc becomes more complex, but not more difficult.

Model Evaluation Criteria

- R^2 (coefficient of determination) is the proportion of the variation in the dependent variable that is predictable from the independent variable(s).
  - It is used to evaluate the performance of regression based estimation models.
    - An R^2 of 1 indicates that the regression predictions perfectly fit the data.
  - However, a high R^2 can occur in the presence of mis-specification of the functional form of a relationship or in the presence of outliers that distort the true relationship.
  - In many cases SSE (sum of squared errors) is preferred over R^2.

Homework Assignment #3

- Construct a Confusion Matrix based on the classifications made by a hypothetical classifier.
  - You will use the data set that will be provided in the course Github site. The data set has classification results in the validation phase of a 5-class classifier.
  - You will do this by developing code.
    - Submissions in R, Python, Java, etc are accepted.
    - You can also submit a Notebook file.
  - Due Date: March 28th
  - Submission: Through Github.

Questions?

CONTACT:
bora.gungoren@atilim.edu.tr