Introduction to Machine Learning (BCS-41)

By
Sushil Kumar Saroj
Assistant Professor

Department of Computer Science & Engineering

Madan Mohan Malaviya University of Technology Gorakhpur
(UP State Govt. University)

Email: skscs@mmmut.ac.in

Syllabus

Unit-I

FOUNDATIONS OF LEARNING- Components of Learning – Learning Models – Geometric Models – Probabilistic Models – Logic Models – Grouping and Grading – Learning Versus Design – Types of Learning – Supervised – Unsupervised – Reinforcement – Theory of Learning – Feasibility of Learning – Error and Noise – Training versus Testing – Theory of Generalization – Generalization Bound – Approximation – Generalization Tradeoff – Bias and Variance – Learning Curve

What is Learning?

"Learning denotes changes in a system that ... enable a system to do the same task … more efficiently the next time." - Herbert Simon

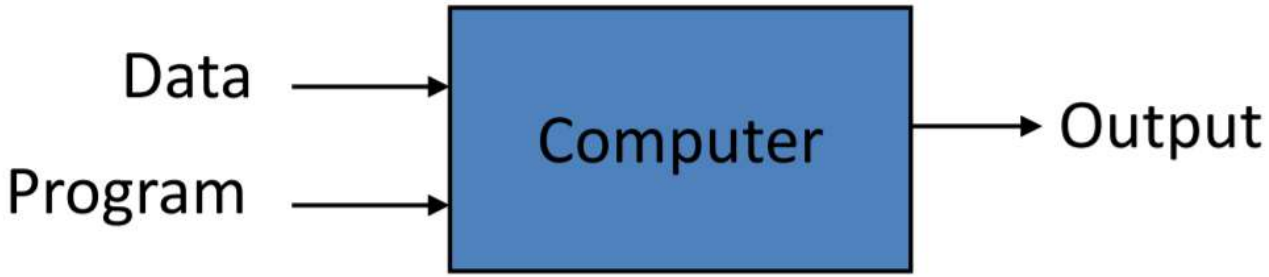
"Learning is constructing or modifying representations of what is being experienced." - Ryszard Michalski

"Learning is making useful changes in our minds." - Marvin Minsky

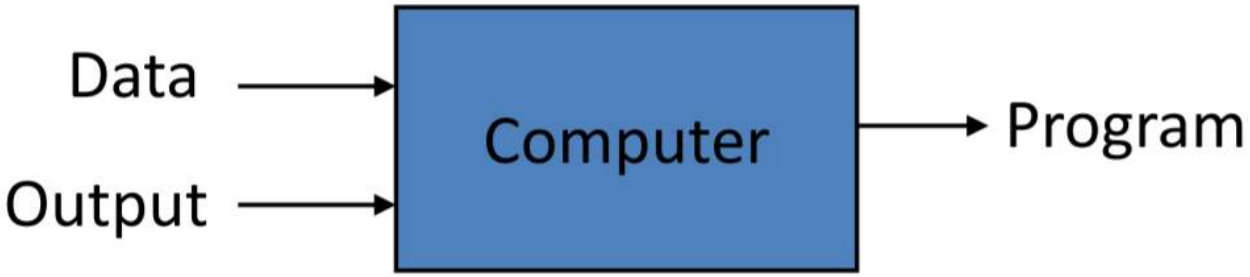
"Machine learning refers to a system capable of the autonomous acquisition and integration of knowledge."

What is Learning?

Traditional Programming



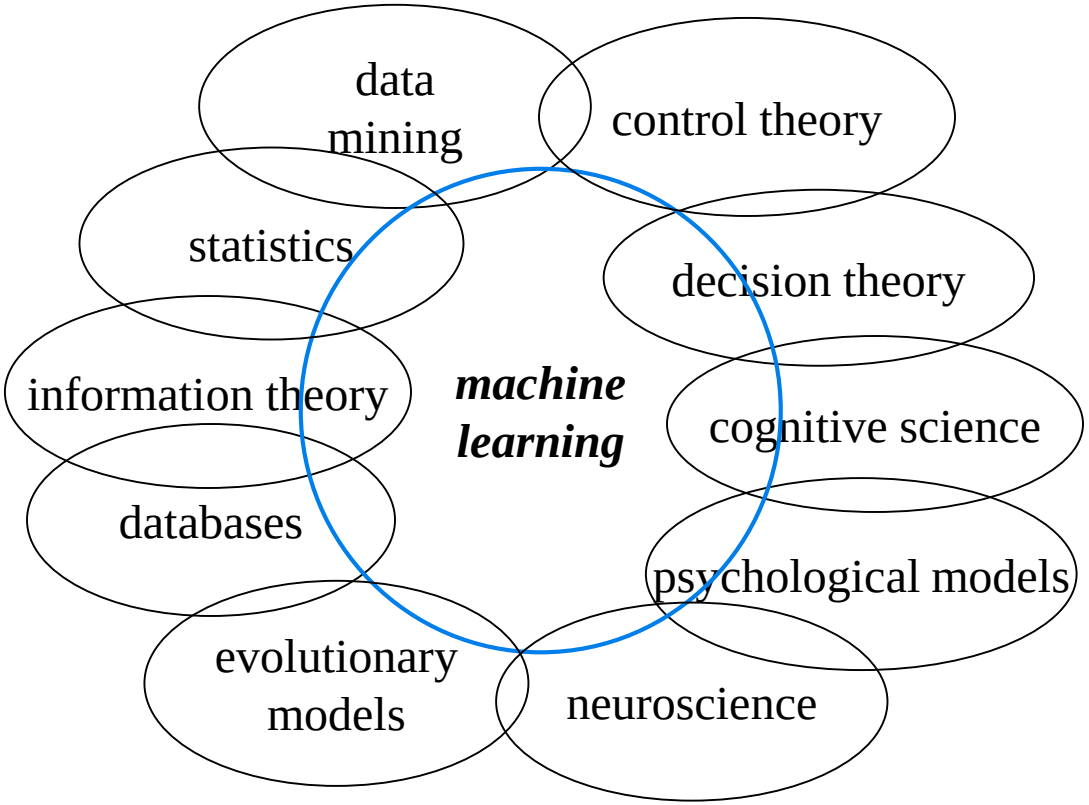
Machine Learning



Why Machine Learning?

- No human experts
  - industrial/manufacturing control
  - mass spectrometer analysis, drug design, astronomic discovery
- Black-box human expertise
  - face/handwriting/speech recognition
  - driving a car, flying a plane
- Rapidly changing phenomena
  - credit scoring, financial modeling
  - diagnosis, fraud detection
- Need for customization/personalization
  - personalized news reader
  - movie/book recommendation

Related Fields



- Machine learning is primarily concerned with the accuracy and effectiveness of the computer system

Components of Learning

- Collecting and preparing data
- Choosing and training a model
- Evaluating a model
- Hyperparameter tuning and Prediction

Learning Models

- Geometric Models
- Probabilistic Models
- Logical Models

Learning Models

Geometric Models

In Geometric models, features could be described as points in two dimensions (x- and y-axis) or a three-dimensional space (x, y, and z). Even when features are not intrinsically geometric, they could be modelled in a geometric manner (for example, temperature as a function of time can be modelled in two axes). In geometric models, there are two ways we could impose similarity.

- We could use geometric concepts like lines or planes to segment (classify) the instance space. These are called **Linear models**
- Alternatively, we can use the geometric notion of distance to represent similarity. In this case, if two points are close together, they have similar values for features and thus can be classed as similar. We call such models as **Distance-based models**

Learning Models

Probabilistic Models

Probabilistic models see features and target variables as random variables. The process of modelling represents and manipulates the level of uncertainty with respect to these variables. There are two types of probabilistic models: Predictive and Generative.

- **Predictive probability models** use the idea of a conditional probability distribution P (Y |X) from which Y can be predicted from X
- **Generative models** estimate the joint distribution P (Y, X). Once we know the joint distribution for the generative models, we can derive any conditional or marginal distribution involving the same variables

Probabilistic models use the idea of probability to classify new entities

Naïve Bayes is an example of a probabilistic classifier.

Learning Models

Logical Models

Logical models use a logical expression to divide the instance space into segments and hence construct grouping models. A logical expression is an expression that returns a Boolean value, i.e., a True or False outcome. Once the data is grouped using a logical expression, the data is divided into homogeneous groupings for the problem we are trying to solve.

There are two types of logical models: Tree models and Rule models.

- **Rule models** consist of a collection of implications or IF-THEN rules. For tree-based models, the 'if-part' defines a segment and the 'then-part' defines the behaviour of the model for this segment. Rule models follow the same reasoning
- **Tree models** can be seen as a particular type of rule model where the if-parts of the rules are organised in a tree structure. Both Tree models and Rule models use the same approach to supervised learning

Learning Models

Groping and Grading Models

- The key difference between grouping and grading models is the way they handle the instance space
- **Grouping models**
- Grouping models break up the instance space into groups or *segments, the number of which is determined at training time*
- They have fixed resolution – cannot distinguish instances beyond a resolution
- At the finest resolution grouping models assign the majority class to all instances that fall into the segment
- Determine the right segments and label all the objects in that segment

Learning Models

Groping Models

- Tree models repeatedly split the instance space into smaller subsets
- Trees are usually of limited depth and don't contain all the available features
- Subsets at the leaves of the tree partition the instance space with some finite resolution
- Instances filtered into the same leaf of the tree are treated the same, regardless of any features not in the tree that might be able to distinguish them.

Learning Models

Grading Models

- They don't use the notion of segment
- Forms one global model over instance space
- Grading models are (usually) able to distinguish between arbitrary instances, no matter how similar they are
- Resolution is, in theory, infinite, particularly when working in a Cartesian instance space
- Support vector machines and other geometric classifiers are examples of grading models
- They work in a Cartesian instance space
- Exploit the minutest differences between instances

Learning Models

Groping versus Grading Models

- Some models combine the features of both grouping and grading models
- linear classifiers are a prime example of a grading model
- Instances on a line or plane parallel to the decision boundary can't be distinguished by a linear model
- There are infinitely many segments

Learning Versus Design

- Machine learning is a powerful tool that drives everything from curated content recommendations to optimized user interfaces
- Machine learning answers questions about user behavior
- Machine learning customizes interfaces to users needs
- Digital product designers need to get familiar with machine learning
- Many warn that designers who don't start learning about ML will be left behind. But I haven't seen one that has explored what design and machine learning have to offer each other
- Design and machine learning function like a flywheel: when connected, each provides value to the other. Together, they open up new product experiences and business value
- Design helps machine learning gather better data

Learning Versus Design

- Machine learning is a hungry beast. To deliver the best results, learning algorithms need vast amounts of detailed data, clean of any confounding factors or built-in biases
- Designers can help create user experiences that eliminate noise in data, leading to more accurate and efficient ML-powered applications
- Design helps set expectations and establish trust with users

Types of Learning

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Types of Learning

Supervised learning

In this type of learning, the data set on which the machine is trained consists of labelled data or simply said, consists both the input parameters as well as the required output.

Supervised Machine Learning Algorithms can be broadly divided into two types of algorithms; Classification and Regression.

- **Classification:** Supervised learning problem that involves predicting a class label
- **Regression:** Supervised learning problem that involves predicting a numerical label

Examples: Linear Regression, Logistic Regression, KNN classification, Support Vector Machine (SVM), Decision Trees, Random Forest, Naive Bayes' theorem.

Types of Learning

Unsupervised learning

Unlike supervised learning algorithms, where we deal with labelled data for training, the training data will be unlabelled for Unsupervised Machine Learning Algorithms. The clustering of data into a specific group will be done on the basis of the similarities between the variables.

- **Clustering:** Unsupervised learning problem that involves finding groups in data
- **Density estimation:** Unsupervised learning problem that involves summarizing the distribution of data
- **Visualization:** Unsupervised learning problem that involves creating plots of data
- **Projection:** Unsupervised learning problem that involves creating lower-dimensional representations of data

Examples: K-means clustering, neural networks

Types of Learning

Reinforcement learning

Reinforcement Learning is a type of Machine Learning in which the machine is required to determine the ideal behaviour within a specific context, in order to maximize its rewards. It works on the rewards and punishment principle which means that for any decision which a machine takes, it will be either be rewarded or punished. Thus, it will understand whether or not the decision was correct. This is how the machine will learn to take the correct decisions to maximize the reward in the long run.

Examples: Q-learning, temporal-difference learning, and deep reinforcement learning.

Error and Noise

Error

Error measures are a tool in ML that quantify the question "how wrong was our estimation". It is a function that compares the output of a learned hypothesis with the output of the real target function. What this means in practice is that we compare the prediction of our model with the real value in data. An error measure is expressed as E(h, f) (a hypothesis $h \in H$, and f is the target function). E is almost always pointwise. It is defined by the difference at two points, therefore, we use the pointwise definition of the error measure e() to compute this error in the different points: e(h(x), f(x)).

Examples:

Squared error: $e(h(x), f(x)) = (h(x)- f(x))^2$

Binary error: $e(h(x), f(x)) = [\![h(x) \neq f(x)]\!]$ (the number of wrong classifications)

Error and Noise

Noise

It refers to the irrelevant information or randomness in a dataset.

We can express noisy target as follows:

Noisy target= deterministic target + noise = $\mathbb{E}[y|x] + \varepsilon$ where $\varepsilon = (y - f(x))$ is the difference between the outcome and the predicted value.

$\mathbb{E}[y|x]$ is the expected value of y knowing x, y is our prediction using the target function h(x) and f(x) is the real value of the data point.

We introduced P(y|x) into our learning scheme to account for the fact that there will always be noise in the relationship between x and y while P(x) represents the random variable x and is necessary for us to use Hoeffding‘s inequality.

Training versus Testing

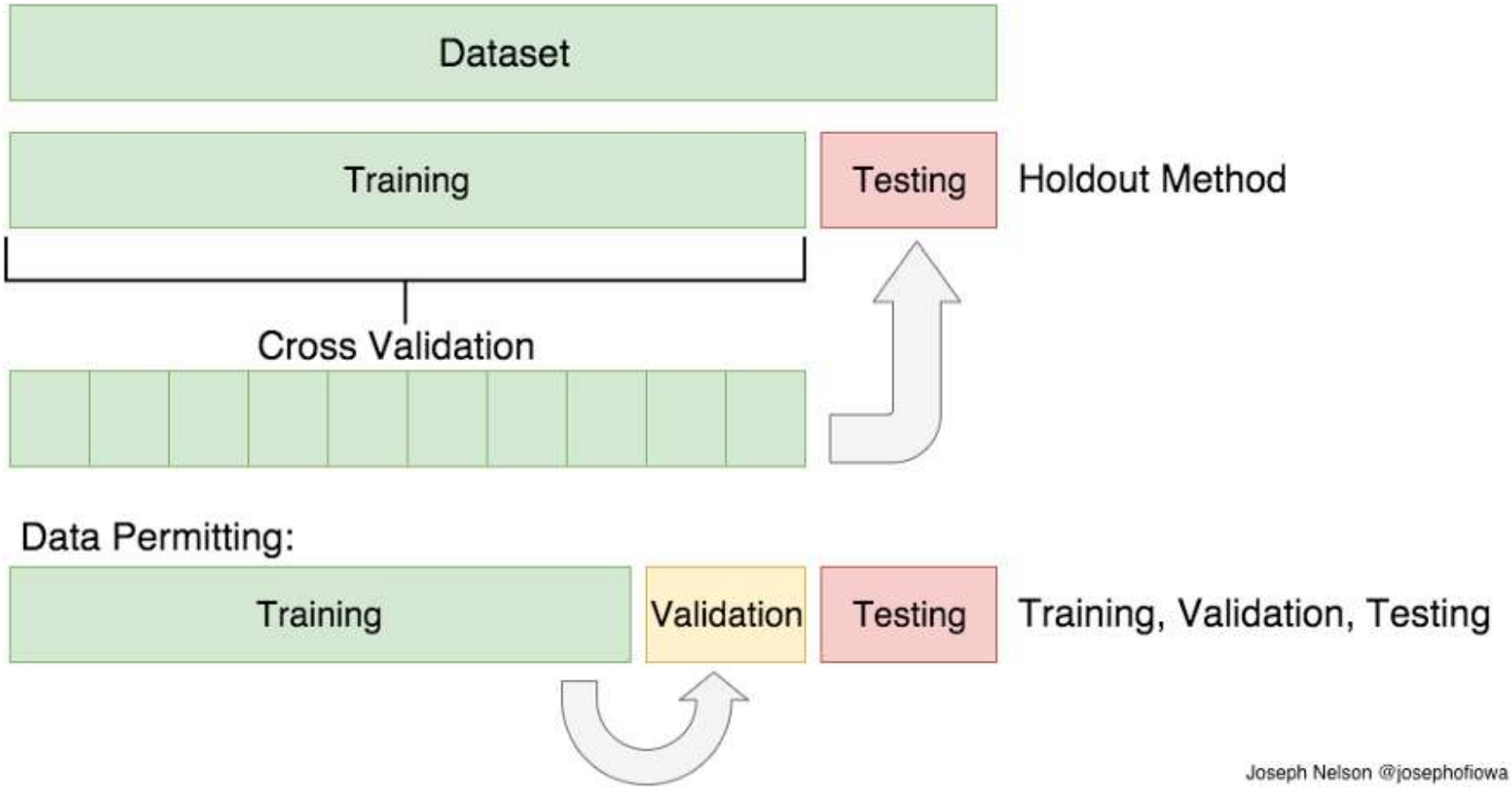
- In a dataset, a training set is implemented to build up a model, while a test (or validation) set is to validate the model built. Data points in the training set are excluded from the test (validation) set
- In Machine Learning, we basically try to create a model to predict the test data
- Usually, a dataset is divided into a training set, a validation set (some people use ‘test set’ instead) in each iteration, or divided into a training set, a validation set and a test set in each iteration

Training versus Testing

Sets:

- **Training Set:** Here, you have the complete training dataset. You can extract features and train to fit a model and so on
- **Validation Set:** This is crucial to choose the right parameters for your estimator. We can divide the training set into a train set and validation set. Based on the validation test results, the model can be trained(for instance, changing parameters, classifiers)
- **Testing Set:** Here, once the model is obtained, you can predict using the model obtained on the training set

Training versus Testing

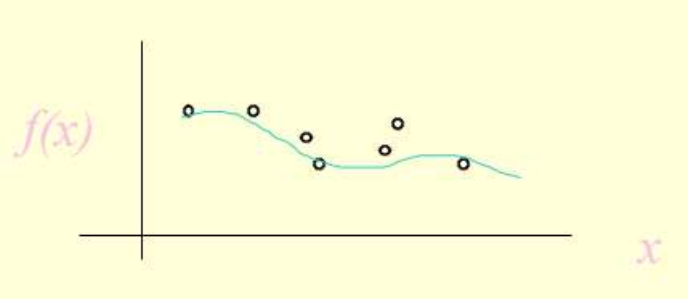


Theory of Generalization

- In machine learning, generalization usually refers to the ability of an algorithm to be effective across a range of inputs and applications
- Our key working assumption is that data is generated by an underlying, unknown distribution D. Rather than accessing the distribution directly, statistical learning assumes that we are given a training sample S, where every element of S is i.i.d and generated according to D. A learning algorithm chooses a function (hypothesis h) from a function space (hypothesis class) H where $H = \{f(x, \alpha)\}$ where α is the parameter vector
- We can then define the generalization error of a hypothesis h as the difference between the expectation of the error on a sample x picked from the distribution D and the empirical loss

Generalization

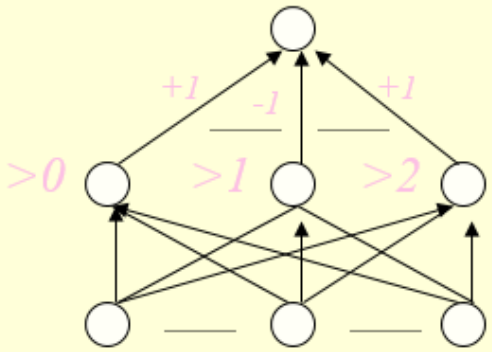
- The objective of learning is to achieve good generalization to new cases, otherwise just use a look-up table
- Generalization can be defined as a mathematical interpolation or regression over a set of training points:



Generalization

An Example: Computing Parity

Parity bit value



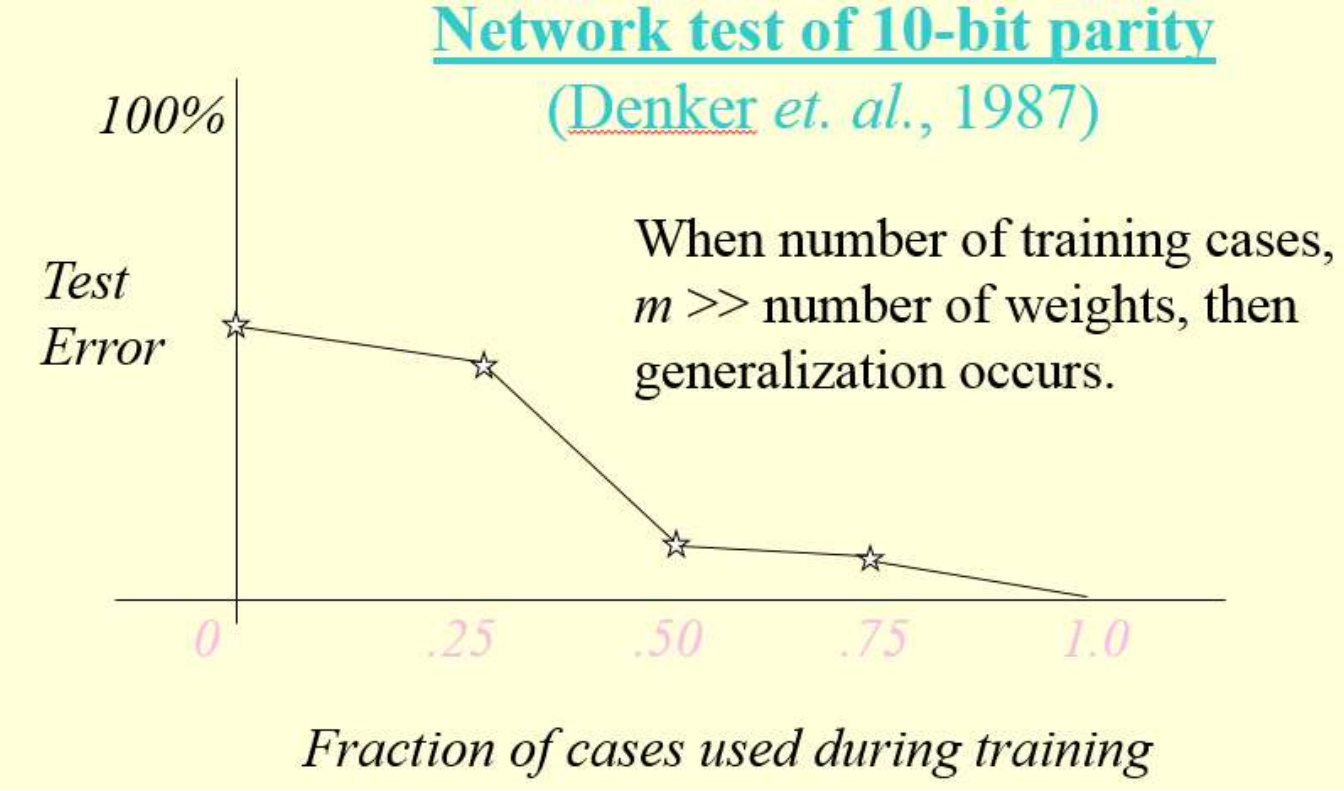
(n+1)^2 weights

n bits of input

2^*n* possible examples

Can it learn from *m* examples to generalize to all 2^*n* possibilities?

Generalization



Generalization

A Probabilistic Guarantee

N = # hidden nodes m = # training cases

W = # weights ε = error tolerance (< 1/8)

Network will generalize with 95% confidence if:

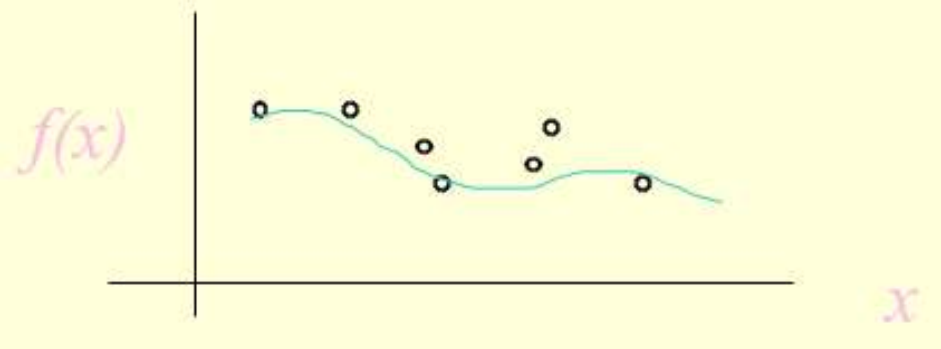
1. Error on training set <
2. $\varepsilon / 2$

$$m > O\left(\frac{W}{\varepsilon}\log_2\frac{N}{\varepsilon}\right) \approx m > \frac{W}{\varepsilon}$$

Based on PAC theory => provides a good rule of practice.

Generalization

- The objective of learning is to achieve good generalization to new cases, otherwise just use a look-up table
- Generalization can be defined as a mathematical interpolation or regression over a set of training points:



Generalization

Over-Training

- Is the equivalent of over-fitting a set of data points to a curve which is too complex
- Occam's Razor (1300s): "plurality should not be assumed without necessity"
- The simplest model which explains the majority of the data is usually the best

Generalization

Preventing Over-training

- Use a separate test or tuning set of examples
- Monitor error on the test set as network trains
- Stop network training just prior to over-fit error occurring- early stopping or tuning
- Number of effective weights is reduced
- Most new systems have automated early stopping methods

Generalization

How can we control number of effective weights?

- Manually or automatically select optimum number of hidden nodes and connections
- Prevent over-fitting = over-training
- Add a weight-cost term to the bp error equation

Generalization Bound

In order for the entire hypothesis space to have a generalization gap bigger than ϵ, at least one of its hypothesis: h_1 **or** h_2 **or** h_3 **or** ... etc should have. This can be expressed formally by stating that:

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}}|R(h)-R_{\text{emp}}(h)|>\epsilon\right]=\mathbb{P}\left[\bigcup_{h\in\mathcal{H}}|R(h)-R_{\text{emp}}(h)|>\epsilon\right]$$

Where $\bigcup$ denotes the union of the events, which also corresponds to the logical **OR** operator. Using the union bound inequality, we get:

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}}|R(h)-R_{\text{emp}}(h)|>\epsilon\right]\leq\sum_{h\in\mathcal{H}}\mathbb{P}[|R(h)-R_{\text{emp}}(h)|>\epsilon]$$

We exactly know the bound on the probability under the summation from our analysis using the Heoffding's inequality, so we end up with:

Generalization Bound

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}}|R(h)-R_{\text{emp}}(h)|>\epsilon\right]\leq 2|\mathcal{H}|\exp(-2m\epsilon^2)$$

Where $|\mathcal{H}|$ is the size of the hypothesis space. By denoting the right hand side of the above inequality by δ, we can say that with a confidence $1-\delta$:

$$|R(h)-R_{\text{emp}}(h)|\leq\epsilon\Rightarrow R(h)\leq R_{\text{emp}}(h)+\epsilon$$

And with some basic algebra, we can express ϵ in terms of δ and get:

$$R(h)\leq R_{\text{emp}}(h)+\sqrt{\frac{\ln|\mathcal{H}|+\ln\frac{2}{\delta}}{2m}}$$

Generalization Bound

- There are two types of bound
  - VC generalization bound
  - Distributed function based bound

VC generalization bound

$$R(h) \lessapprox \widehat{R}_n(h) + \epsilon(\mathcal{H}, n)$$

Approximation- Generalization Tradeoff

Given a set $\mathcal{H}$, find a function $h \in \mathcal{H}$ that minimizes $R(h)$

Our goal is to find an $h \in \mathcal{H}$ that approximates the Bayes classifier, or some true underlying function

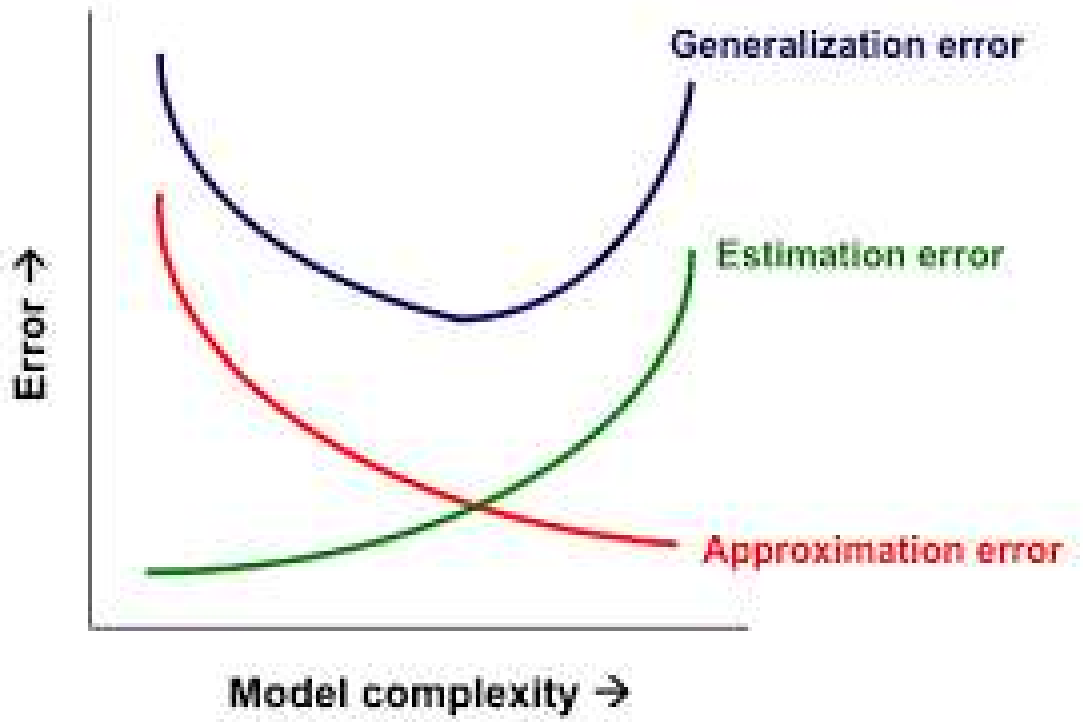
More complex $\mathcal{H}$ ➡ better chance of ***approximating*** the ideal classifier/function

Less complex $\mathcal{H}$ ➡ better chance of ***generalizing*** to new data (out of sample)

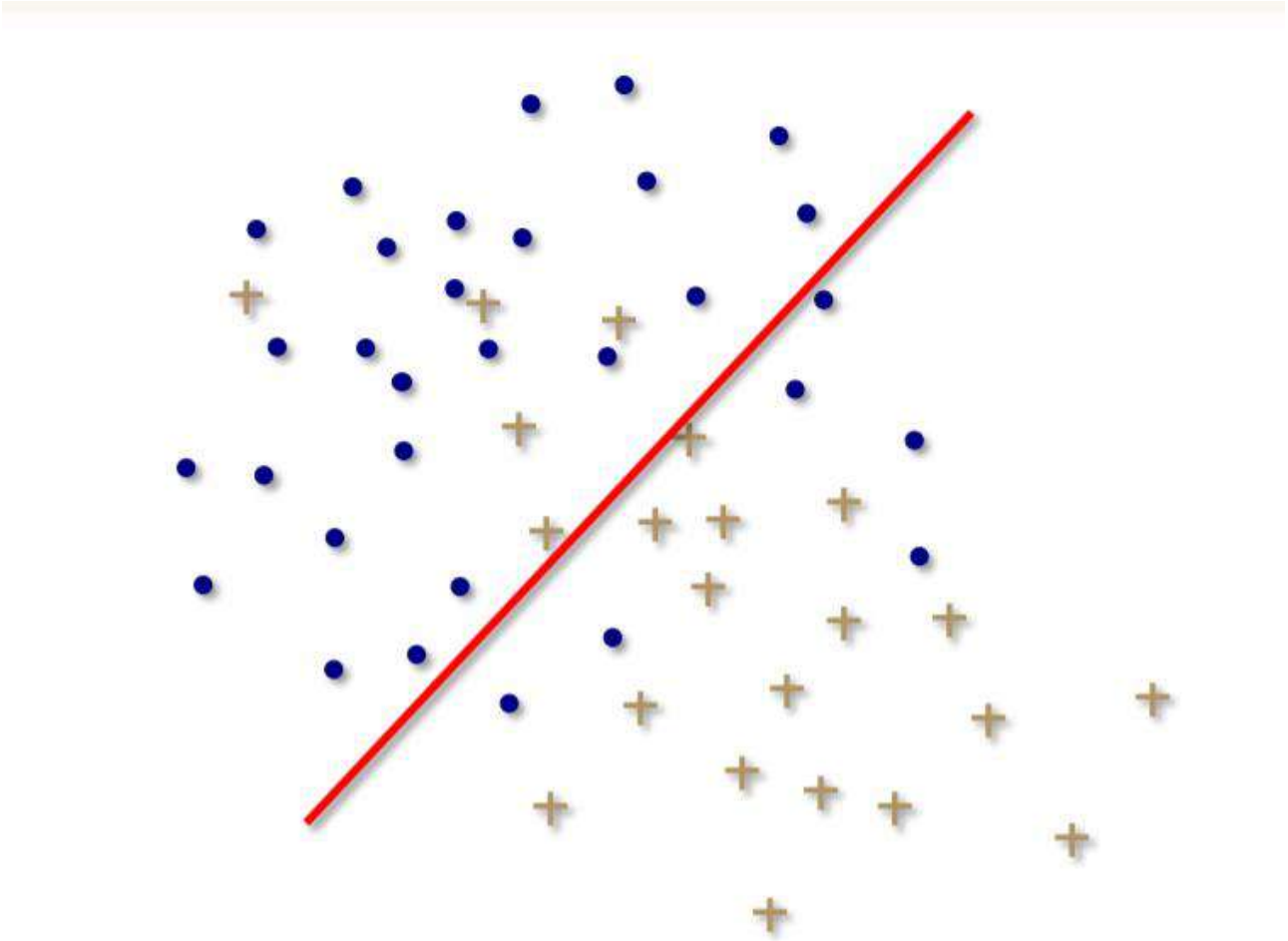
Regularization plays a similar role, by biasing us away from complex classifiers/functions

We must carefully limit "complexity" to avoid ***overfitting***

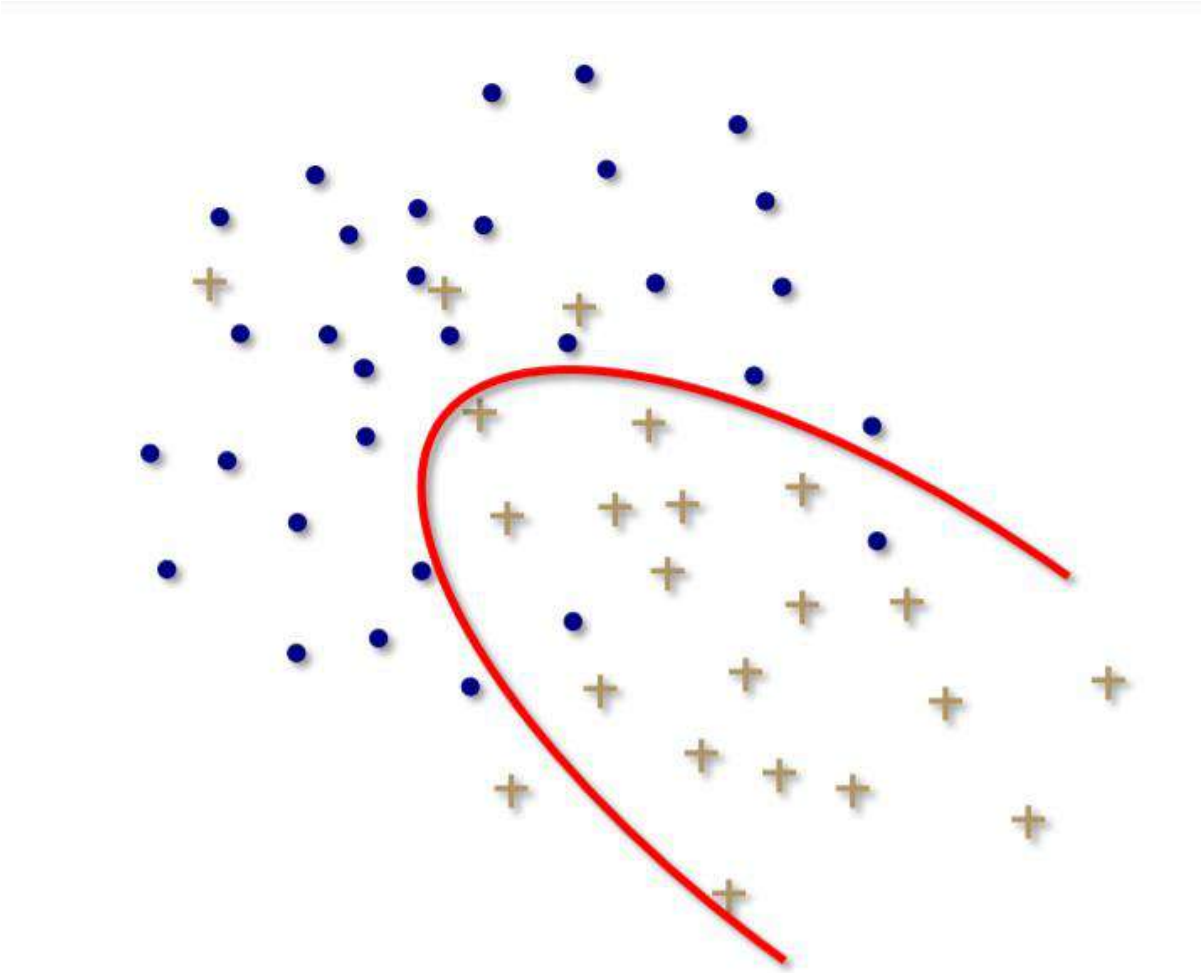
Approximation- Generalization Tradeoff



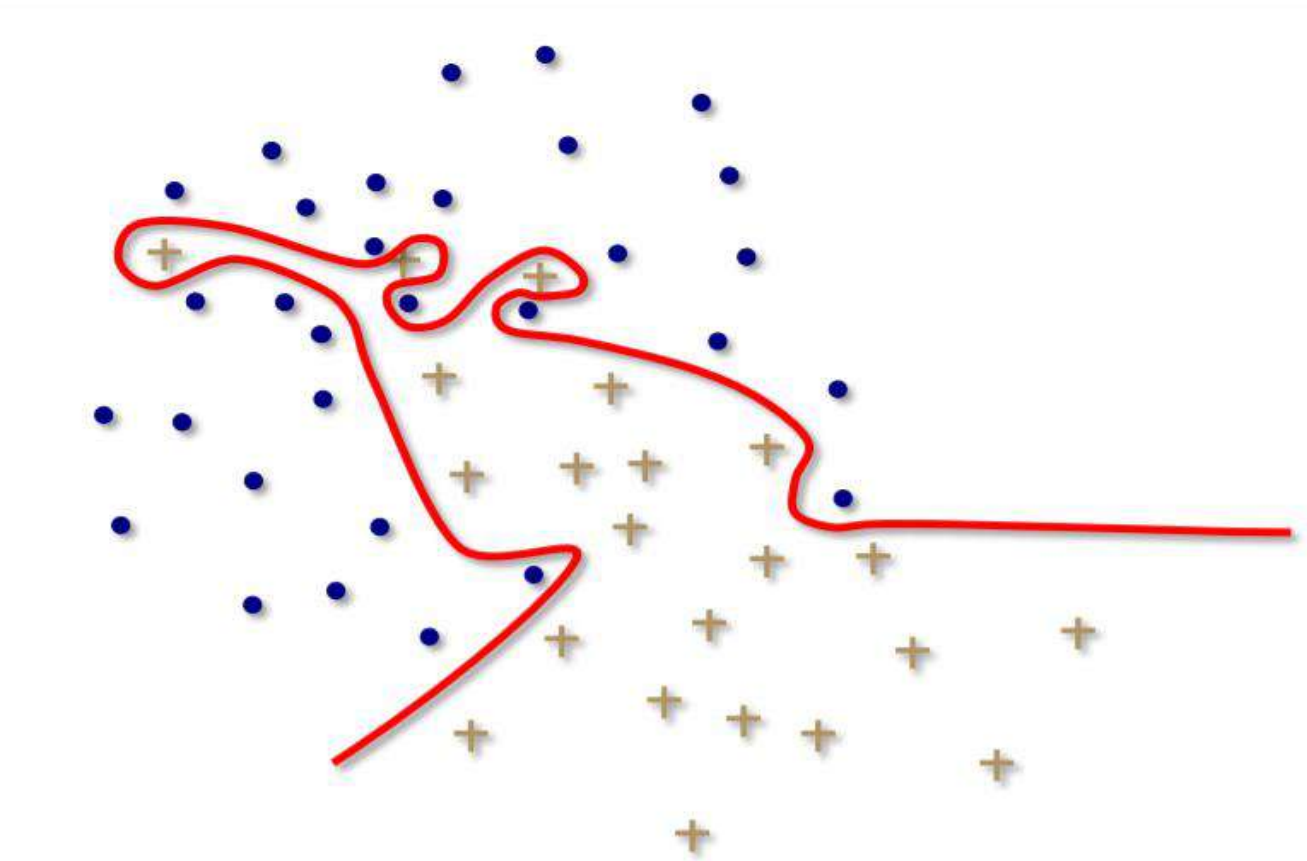
Overfitting



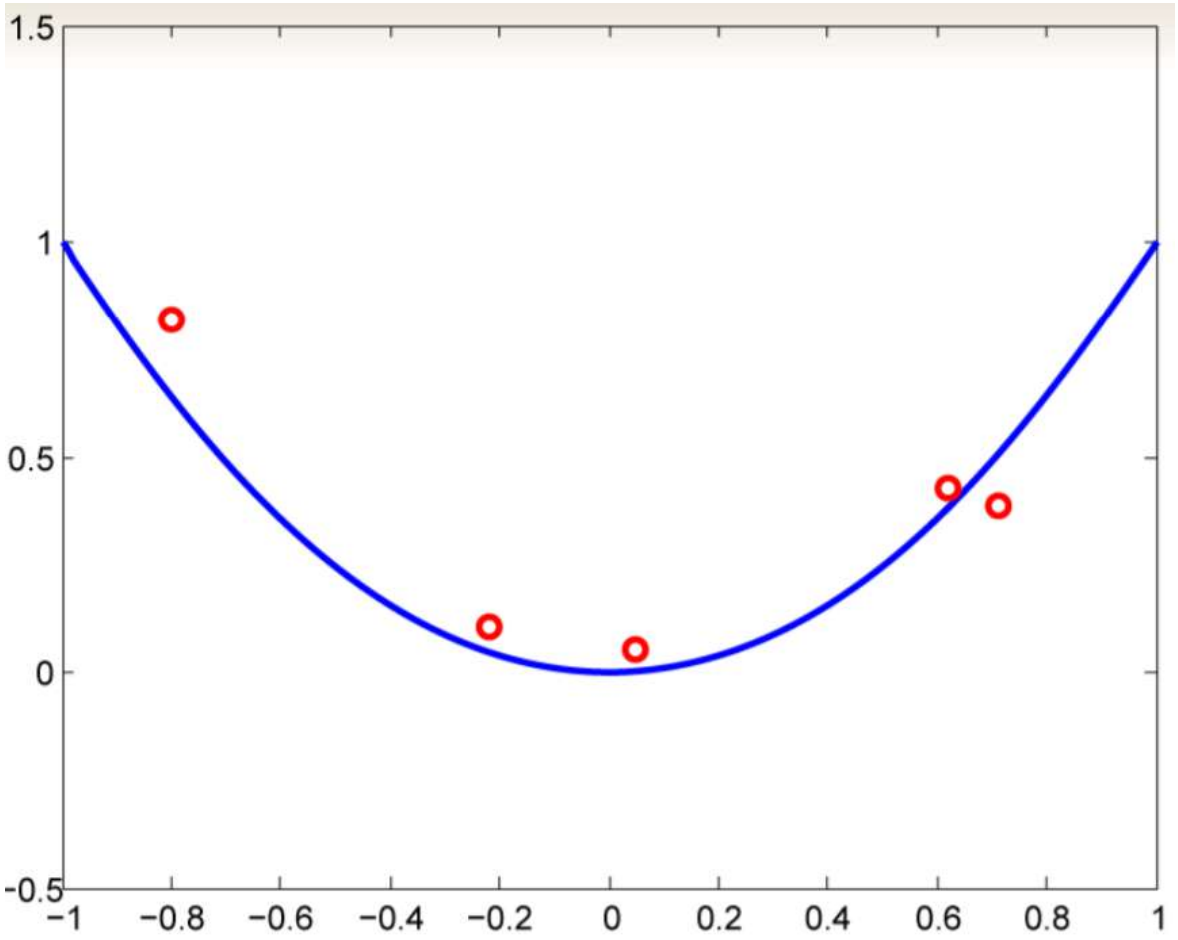
Overfitting



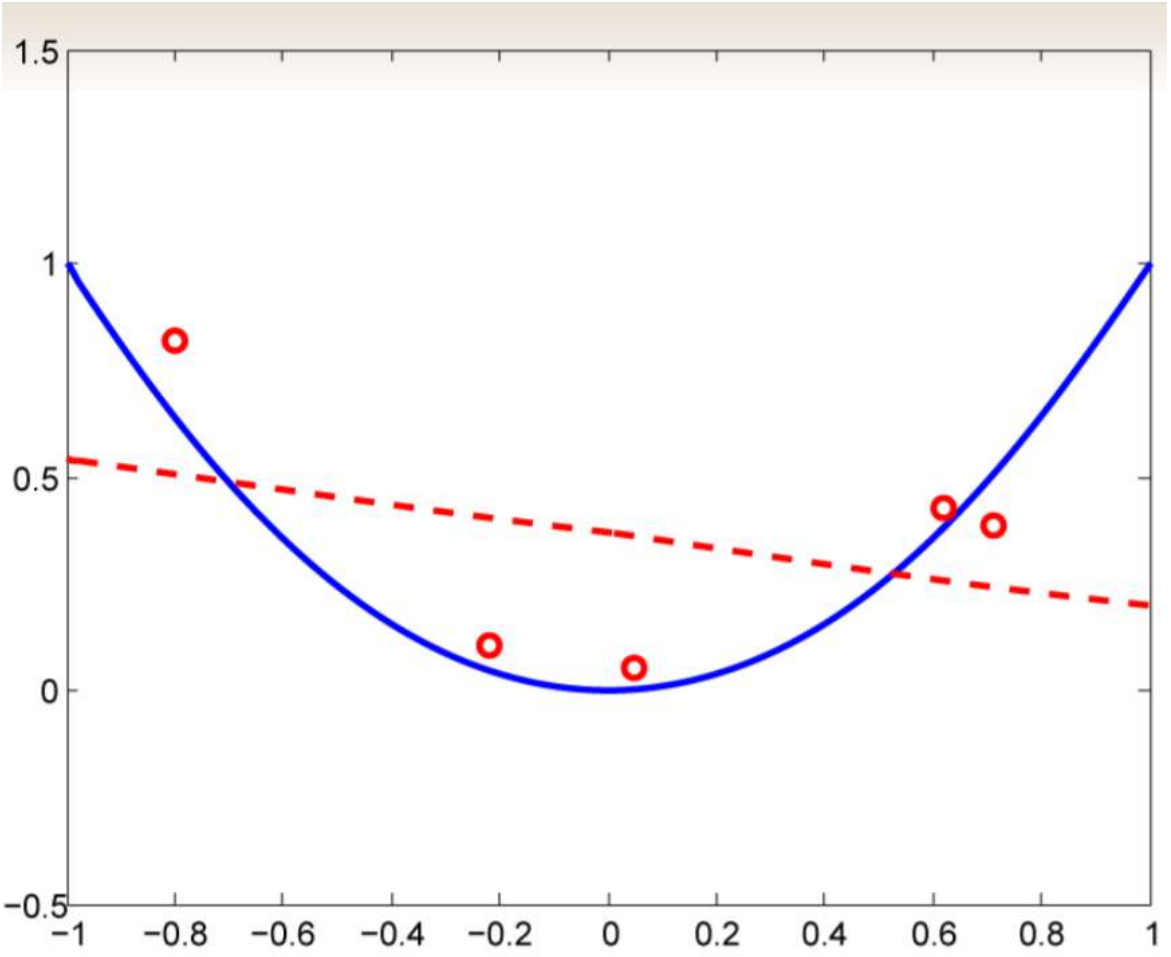
Overfitting



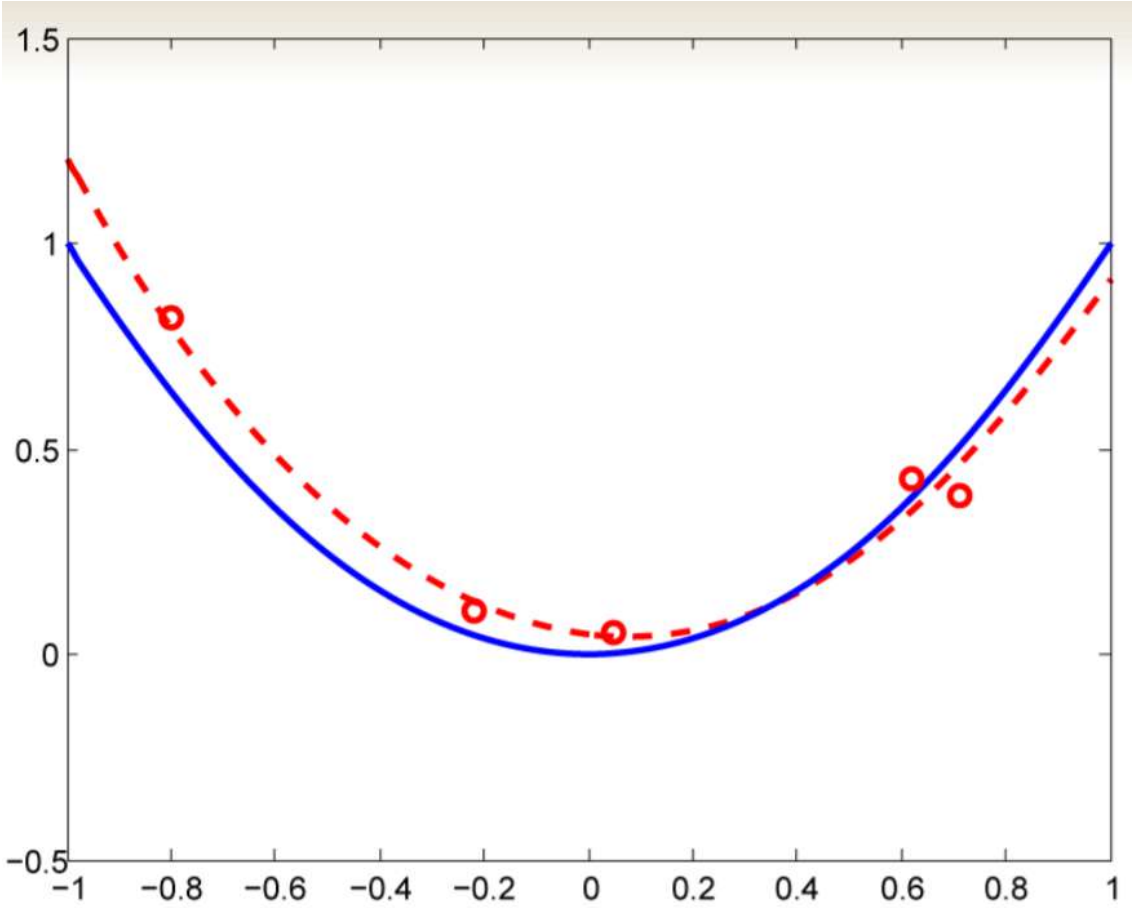
Overfitting



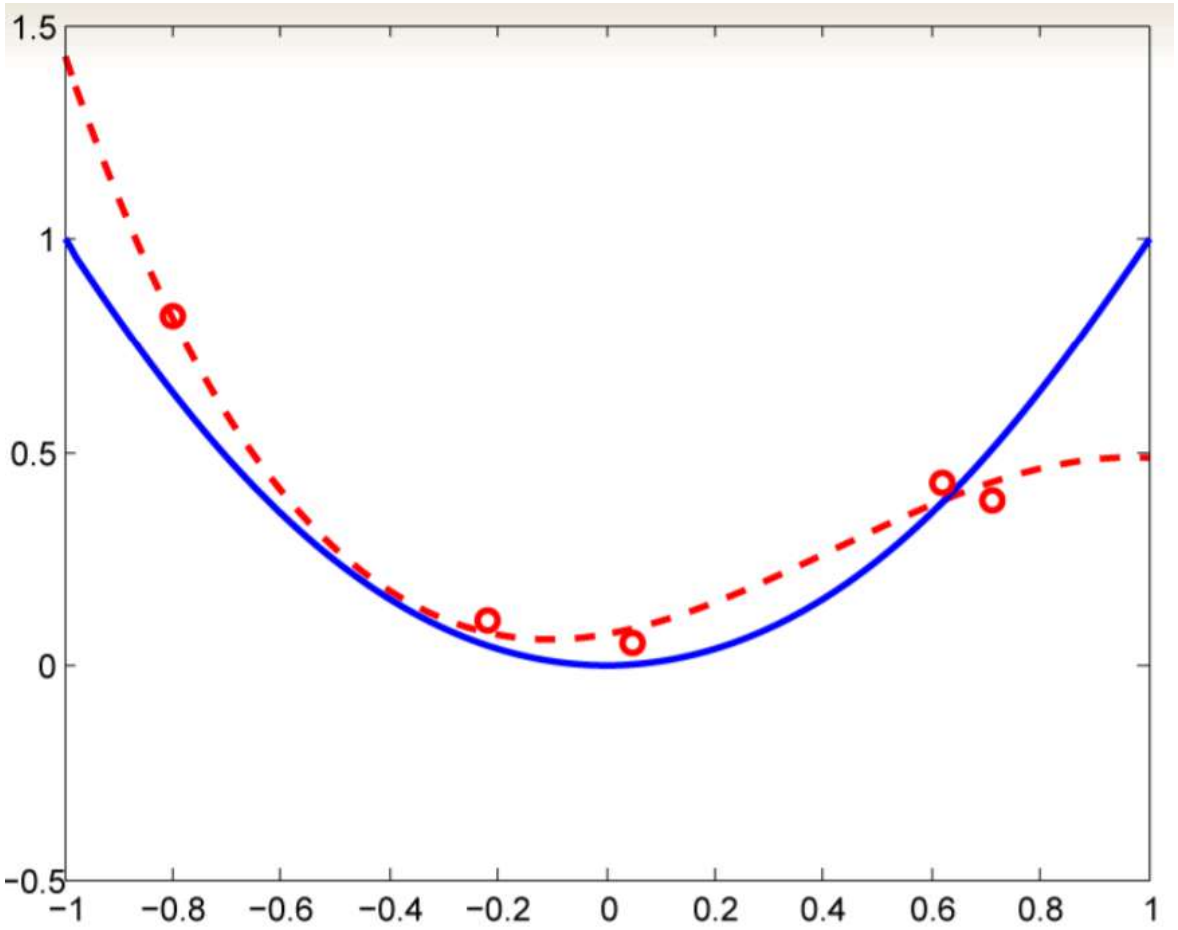
Overfitting



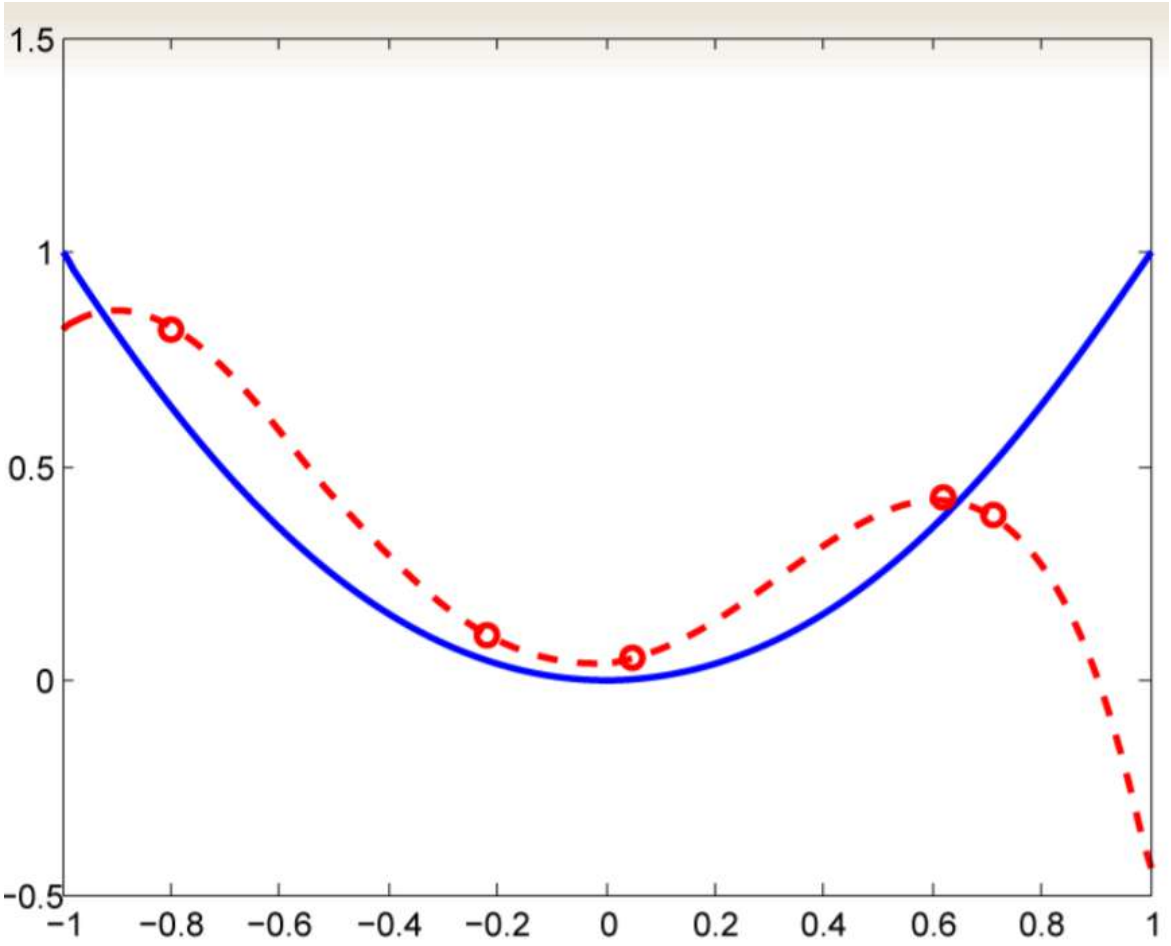
Overfitting



Overfitting



Overfitting



Bias and Variance

Bias

Bias is the difference between the Predicted Value and the Expected Value.

Mathematically, let the input variables be X and a target variable Y. We map the relationship between the two using a function f. Therefore,

$$Y = f(X) + e$$

Here ‘e’ is the error that is normally distributed. The aim of our model f'(x) is to predict values as close to f(x) as possible. Here, the Bias of the model is:

$$Bias[f'(X)] = E[f'(X) – f(X)]$$

As I explained above, when the model makes the generalizations i.e. when there is a high bias error, it results in a very simplistic model that does not consider the variations very well. Since it does not learn the training data very well, it is called Underfitting.

Bias and Variance

Variance

Contrary to bias, the Variance is when the model takes into account the fluctuations in the data i.e. the noise as well. So, what happens when our model has a high variance?

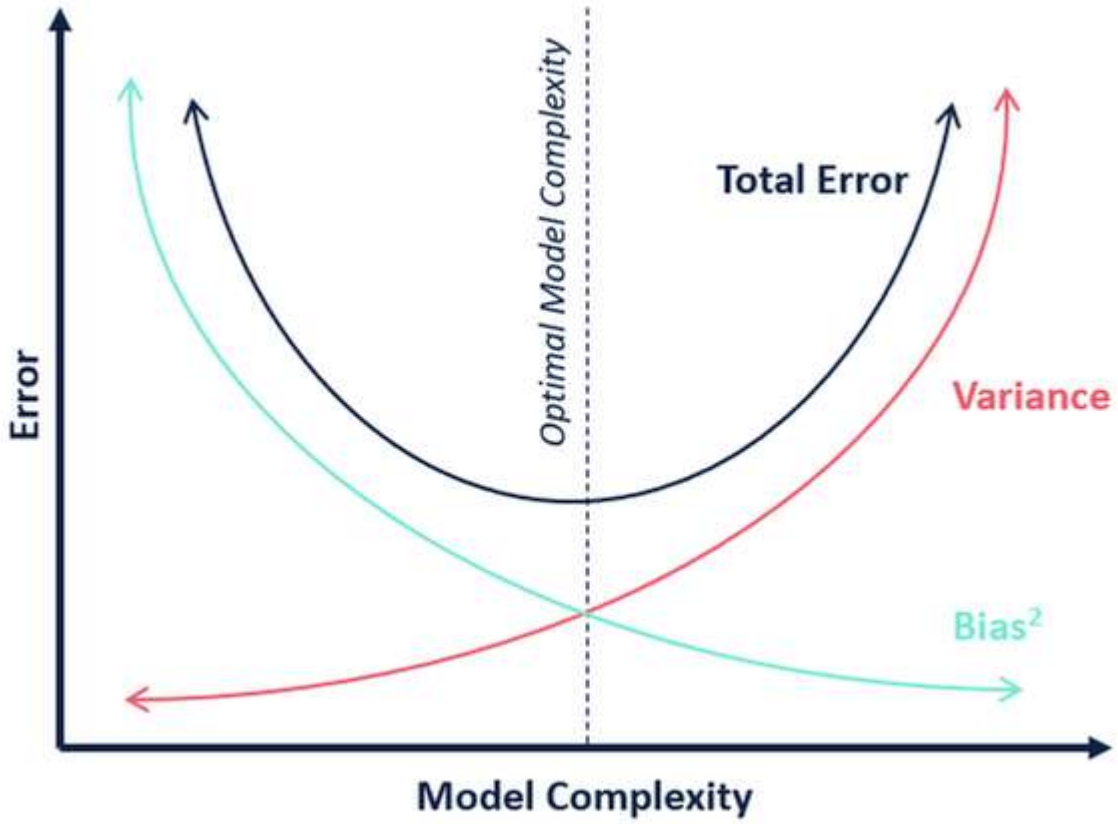
The model will still consider the variance as something to learn from. That is, the model learns too much from the training data, so much so, that when confronted with new (testing) data, it is unable to predict accurately based on it.

Mathematically, the variance error in the model is:

$$\text{Variance}[f(x)) = E[X^2] - E[X]^2$$

Since in the case of high variance, the model learns too much from the training data, it is called **overfitting**.

Bias and Variance



Learning curves

- Learning curves are plots that show changes in learning performance over time in terms of experience
- Learning curves of model performance on the train and validation datasets can be used to diagnose an underfit, overfit, or well-fit model
- Learning curves of model performance can be used to diagnose whether the train or validation datasets are not relatively representative of the problem domain
- Generally, a learning curve is a plot that shows time or experience on the x-axis and learning or improvement on the y-axis

Learning curves are deemed effective tools for monitoring the performance of workers exposed to a new task. LCs provide a mathematical representation of the learning process that takes place as task repetition occurs.

Learning curves

- **Train Learning Curve:** Learning curve calculated from the training dataset that gives an idea of how well the model is learning
- **Validation Learning Curve:** Learning curve calculated from a hold-out validation dataset that gives an idea of how well the model is generalizing
- **Optimization Learning Curves:** Learning curves calculated on the metric by which the parameters of the model are being optimized, e.g. loss
- **Performance Learning Curves:** Learning curves calculated on the metric by which the model will be evaluated and selected, e.g. accuracy
- There are three common **dynamics** that you are likely to observe in learning curves; they are:
  - Underfit
  - Overfit
  - Good Fit

Learning curves

- Underfitting refers to a model that cannot learn the training dataset.
- A plot of learning curves shows underfitting if:
  - The training loss remains flat regardless of training
  - The training loss continues to decrease until the end of training

- Overfitting refers to a model that has learned the training dataset too well, including the statistical noise or random fluctuations in the training dataset.
- A plot of learning curves shows overfitting if:
  - The plot of training loss continues to decrease with experience
  - The plot of validation loss decreases to a point and begins increasing again

Learning curves

- A good fit is the goal of the learning algorithm and exists between an overfit and underfit model
- A good fit is identified by a training and validation loss that decreases to a point of stability with a minimal gap between the two final loss values
- A plot of learning curves shows a good fit if:
  - The plot of training loss decreases to a point of stability
  - The plot of validation loss decreases to a point of stability and has a small gap with the training loss

References

[1] Bartlett, Peter L., Dylan J. Foster, and Matus J. Telgarsky. "Spectrally-normalized margin bounds for neural networks." Advances in Neural Information Processing Systems. 2017.

[2] Shawe-Taylor, J. and Rivasplata, O., "Statistical Learning Theory: A Hitchhiker's Guide", 2012, [link]


THANK YOU