Machine Learning - I (CS/DS 864)
Aug–Dec
***Instructor*: Prof. G.Srinivasaraghavan**

Lecture Notes

Learning Problem Formulation

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

Contents

List of Figures

List of Tables

1 Introduction

This course will primarily be concerned with *predictive learning*. Predictive Learning is the act of deriving (estimating) a *useful* model of an unknown system / process from available (finite) *training data*. The model can then be used as a proxy for the 'real' system to predict future events almost like the real system. We are concerned with algorithms and techniques to derive such models from data presented to us for 'learning' about the unknown system. The words / phrases that have been highlighted have a lot of significance. Let's explore their implications briefly below.

- **Useful**: We are merely interested in estimating a model that has utilitarian value. More precisely, the estimated model is NOT expected to be a faithful reproduction of the (unknown) 'system' or the 'process' (characterized by an unknown probability distribution / density function) that we are trying to learn about. The estimated model is only expected to be able to 'mimic' the unknown system or process to a desired level of accuracy on any input. In other words it is expected to be able to predict with a reasonable accuracy something that is of interest about this unknown probability distribution. What is revealed about the system / process that we trying to 'learn' is almost always inadequate to determine the system completely — there are unobserved parameters, data is noisy, data is incomplete. Also there is exactly one system that is identical to any given system — and that is itself — whereas there could be many that share (almost) similar behaviour. All that predictive learning asks for is SOME such model that exhibits a behaviour (according to some measure) that approximates the system under observation up to a known margin of error. So the problem of deriving a useful model is invariably lot more tractable than the problem of recovering or reproducing the unknown system under observation.

  In practice however usefulness of a model typically goes beyond prediction accuracy. It is also expected that the model has these other non-functional characteristics:

  - The model can be easily interpreted by humans, in the language of the domain in which it is expected to be put to use.
  - The model gives us additional insights into the unknown system, the nature of the domain, etc.

- **Data**: Set of *data points*, each of which is a vector of *features* or *attributes* of an entity from the domain. We often treat each feature as a numerical value — all attributes are expected to have been converted into an equivalent numerical value. Therefore a data point with d-features can be thought of as a vector $\boldsymbol{x} \in \mathcal{R}^d$.

- **Training Data**: The learning mechanism typically has access to data points that it can learn from — called the *training data*. The data is clearly finite and typically small — a learning algorithm that produces a model in finite time can at best use a finite amount of training data. However the domain where the data points come from is more-often-than-not infinite or at least intractably large — in any case several orders of magnitude larger than the data points that are available for the system to learn from. We also assume that the training data is 'representative' of the domain. Formally what this

means is that the training data has been derived from the same (unknown) probability distribution that characterizes the domain. An additional assumption often made in the context of learning is that each training data point has been drawn *independently* of the others from the same probability distribution. This is often referred to as the *i.i.d*-assumption — that each data point is *independently* and *identically distributed*. Every data point is sampled afresh from the same distribution without any 'hangover' from the previously sampled data points.

To conclude – here is a famous quote by the British Mathematician / Statistician George Box — "All models are wrong. Some models are useful." In the next section we define the predictive learning problem formally.

2 The Predictive Learning Problem

The learning problem consists of the following components as shown in Figure 1.

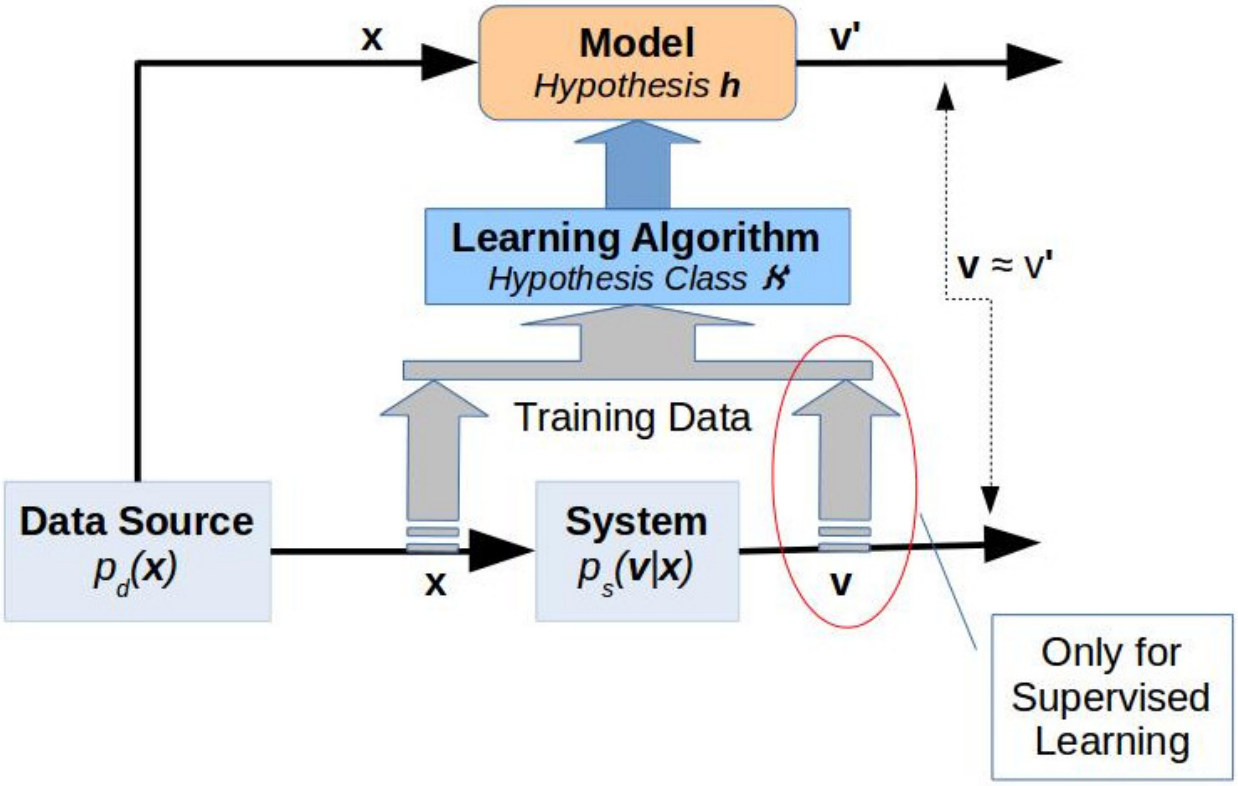

Figure 1: Learning System

1. **Data Source**: Characterizes the pattern in which data points occur. We imagine an unknown data source that generates data points $\boldsymbol{x}$ to be processed by the system according some probability distribution $p_d(\boldsymbol{x})$. This is sometimes called the *marginal distribution*. We denote the data source, the domain of all possible data points as $\mathcal{D}$.

2. **System to be Learnt**: We imagine an unknown system that maps inputs fed by the data source to observable (and measurable) outputs — given by the output vector $\boldsymbol{v} \in \mathcal{V}$ where $\mathcal{V}$ denotes the space of all possible outputs (values) of the system. Since we do not have all the parameters required to define the input-output behaviour of the system completely and/or owing to inaccuracies / noise in measurements, the input-output behaviour of the system as we observe it will typically be probabilistic — probability of an observable conditioned on the input data point. The input-output behaviour is therefore thought of as the conditional probability distribution $p_s(\boldsymbol{v}|\boldsymbol{x})$. We often denote the overall observed behaviour of the system along with the nature of the data source as the joint probability distribution

$$p(\boldsymbol{x}, \boldsymbol{v}) = p_s(\boldsymbol{v}|\boldsymbol{x}).p_d(\boldsymbol{x})$$

3. **Learning Algorithm / Machine**: The learning algorithm $\mathcal{A}$ takes finite (some number $n > 0$) data samples $\mathcal{D}_n$ from the input and in supervised learning also the expected output of the System for each of the inputs to produce a model h (called a *hypothesis*) that is expected to mimic the System. The hypothesis produced in general could be a probability distribution over $\mathcal{V}$. Let $\mathcal{H}$ be the class of all hypotheses that the algorithm $\mathcal{A}$ is capable of producing. The relationship between the learning algorithm $\mathcal{A}$, training data $\mathcal{D}_n$ and the hypothesis h is formalized as

$$\mathcal{A}(\mathcal{D}_n) = h$$

Sometimes it might be convenient to parametrize $\mathcal{H}$ with a finite set of parameters $\boldsymbol{w} \in \mathcal{W}$ for some parameter space $\mathcal{W}$. Every specific choice of the parameters $\boldsymbol{w}$ would denote a hypothesis in $\mathcal{H}$ uniquely. In this view the output of $\mathcal{A}$ is just a parameter vector $\boldsymbol{w}$. We would then denote a hypothesis generated by $\mathcal{A}$ as

$$h_{\boldsymbol{w}}(\boldsymbol{x}) : \mathcal{D} \to \mathcal{V}$$

4. **Loss Function**: We expect that the hypothesis $h_{\boldsymbol{w}}$ mimics the 'real' system — in other words we require that $h_{\boldsymbol{w}}(\boldsymbol{x})$ approximates the 'actual' output $s(\boldsymbol{x})$ of the real system for a given input $\boldsymbol{x}$. How well this approximation is, is determined by a *loss function*

$$\mathcal{L}(\boldsymbol{x}, h_{\boldsymbol{w}}) : \mathcal{D} \times \mathcal{V} \to \mathcal{R}^+$$

that measures the error committed by the model at any given point $\boldsymbol{x}$ — larger the error, poorer the approximation.

5. **(True) Model Risk**: This quantifies the 'risk' associated with using the output hypothesis h of the learning algorithm as a surrogate for the real system S. The risk is thought of as the expected loss (as quantified by the loss function) on a random input from $\mathcal{D}$. Let $\boldsymbol{w}$ be the parameters associated with h. Formally the risk $R(\boldsymbol{w})$ is defined as

$$R(\boldsymbol{w}) = \int_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, h(\boldsymbol{x}|\boldsymbol{w})).p_d(\boldsymbol{x})\ d\boldsymbol{x} \tag{1}$$

The predictive learning problem is the problem of estimating the parameters $\boldsymbol{w}$ of the hypothesis for which model risk is a minimum. Formally

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} R(\boldsymbol{w}) \tag{2}$$

The above problem formulation is the one that is most commonly encountered, most extensively studied and the most well understood. A few notable characteristics of this formulation are:

However this is certainly not a universal formulation — there are several machine learning problems that do not neatly fit into this framework. Some examples are briefly discussed below:

- **Online Learning**: There is a subtle assumption in the above formulation that the training data points are all available before the learning process starts. An 'online' scenario where the data points 'stream' in one at a time and the learning happens as and when new data points come in, will require a different formulation.
- **Interactive Learning**: The 'trainer' (the data source and the system together in our formulation) and the 'learning algorithm' are both assumed to be 'passive'. In this the trainer supplies the data points and that's the only 'communication' between the trainer and the learner. One could imagine a scenario similar to the way humans learn from teachers. The trainer could observe the learner's responses and tailor the next set of training data points accordingly. Similarly the learner could query the trainer during the learning process to decide on how/what will be learnt next. Such training scenarios are referred to as *interactive learning*. **Reinforcement Learning** is a kind of interactive learning where learning happens in stages, with the outcome of each stage being 'judged' and the evaluation of the outcome 'reinforces' the learning in the next stage.
- **Time-Series Data**: The i.i.d assumption about the data points may not hold. For instance learning from time-series data is where data comes in a sequence with every part of the sequence potentially influences the rest of the sequence following it. Learning from signals, learning from natural language are examples of scenarios involving time-series data. These clearly does not fit neatly into the above framework.
- **Transduction Learning**: The above approach first estimates a model that is applicable *everywhere*, i.e., for the entire domain $\mathcal{D}$. The underlying assumption is that there are many many more samples 'out there' on which the model will eventually be tested than the size of the training set. This may be an overkill for problems where we know *beforehand* that we only need to predict the outcome for inputs that belong to a small subset of the domain.
- **Multi-Model Learning**: The above framework assumes that there is a single model that applies to all of the training data / input domain. In Multi-Model formulation we seek to estimate multiple models, each of which describes a different subset of the training data.

We will come up with specific formulations for some of these alternate learning paradigms when we encounter them later in the course.

3 Some Typical Machine Learning Problems

In this section we follow the framework we have laid out above to formulate a few commonly occurring machine learning problems.

3.1 Classification

Binary classification problem is the problem of assigning a label ±1 to any given data point based on its attributes $\boldsymbol{x}$. The generalization of the problem to k-classes with k-distinct labels is straightforward. We assume in general that the output domain $\mathcal{V}$ consists of a finite set of labels that each data point can be classified into. Let $v(\boldsymbol{x})$ be the label associated with a data point $\boldsymbol{x}$. The parameters $\boldsymbol{w}$ are those required to describe the separators. The linear case would correspond to a $(d+1)$-vector representing the coefficients of the hyperplane separator in $(d+1)$-dimensional homogeneous coordinate system. The loss function in this case would be

$$\mathcal{L}(\boldsymbol{x}, h_{\boldsymbol{w}}(\boldsymbol{x})) = \begin{cases} 1 & \text{if } v(\boldsymbol{x}) = h_{\boldsymbol{w}}(\boldsymbol{x}) \\ 0 & \text{otherwise} \end{cases}$$

The model risk would be the expected number of mis-classifications, which we want to minimize.

3.2 Clustering / Vector Quantization

The clustering problem tries to arrive at groups or clusters among data points such that intra-cluster separations are minimized and the inter-cluster separations are maximized. We want to group all 'similar' (as defined by a proximity metric $d(\mathbf{x}, \mathbf{y})$ defined between any two data points $\mathbf{x}, \mathbf{y}$) together into a cluster. Each cluster is represented by a point (not necessarily one of the given data points) within the cluster. The number of clusters k is specified beforehand. The learning algorithm produces a hypothesis that maps any given data point to one of k cluster centers $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$. This is an unsupervised learning scheme and hence the 'system' is kind of irrelevant. In other words

$$\mathcal{L}(\mathbf{x}, h_{\mathbf{w}}(\mathbf{x})) = \mathbf{d}(\mathbf{x}, h_{\mathbf{w}}(\mathbf{x})) \quad \text{and} \quad h_{\mathbf{w}}(\mathbf{x}) : \mathcal{D} \to \mathcal{C}$$

The parameter vector $\mathbf{w}$ is a kd-dimensional vector capturing every coordinate of the each cluster center. A common choice for the distance metric $d(\mathbf{x}, \mathbf{y})$ is the square distance from the cluster center (Euclidean distance), i.e.,

$$d(\mathbf{x}, \mathbf{y}) = (\mathbf{y} - \mathbf{x})^T.(\mathbf{y} - \mathbf{x})$$

In this case the model risk will be the cumulative square deviation from the cluster centers. For instance clustering of a set of data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ would result in the model risk given by

$$R(\mathbf{w}) = \sum_{i=0}^{n} \mathbf{d}(\mathbf{x}_i, h_{\mathbf{w}}(\mathbf{x}_i)).p_d(\mathbf{x}_i)$$

3.3 Regression

Regression is the problem of fitting a set of data points into a function — this is the case where the system represents a function $s(\boldsymbol{x})$ and the training set $\mathcal{D}_n$ consists of a set of pairs $\{(\boldsymbol{x}_1, v_1), \ldots, (\boldsymbol{x}_n, v_n)\}$ where $v_i = s(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) + \eta$ for some function $f(\boldsymbol{x})$ and some random noise η with zero mean and variance σ^2. This means

$$\int_v \eta \; p_s(v|\boldsymbol{x}) \; dv = 0 \tag{3}$$

Exercise 1 *Show that the mean over the output conditional probability is $f(\boldsymbol{x})$, i.e.,*

$$f(\boldsymbol{x}) = \int_v v \; p_s(v|\boldsymbol{x}) \; dv$$

We need to find a function that best approximates $s(\boldsymbol{x})$. Suppose the class of hypotheses we can consider is parametrized by the vector $\boldsymbol{w}$. A common choice for the loss function is the square error function, averaged over all possible random outputs v for any given $\boldsymbol{x}$.

$$\mathcal{L}(\boldsymbol{x}, h_{\boldsymbol{w}}(\boldsymbol{x})) = \int_v (v - h_{\boldsymbol{w}}(\boldsymbol{x}))^2 p_s(v|\boldsymbol{x}) \; dv$$

The model risk is given by

$$\begin{aligned}
R(\boldsymbol{w}) &= \int_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, h_{\boldsymbol{w}}(\boldsymbol{x})).p_d(\boldsymbol{x}) \; d\boldsymbol{x} \\
&= \int_{\boldsymbol{x}} \int_v (v - h_{\boldsymbol{w}}(\boldsymbol{x}))^2 p(\boldsymbol{x}, v) \; dv \; d\boldsymbol{x} \\
&= \int_{\boldsymbol{x}} \int_v (v - f(\boldsymbol{x}) + f(\boldsymbol{x}) - h_{\boldsymbol{w}}(\boldsymbol{x}))^2 p(\boldsymbol{x}, v) \; dv \; d\boldsymbol{x} \\
&= \int_{\boldsymbol{x}} \int_v \eta^2 p(\boldsymbol{x}, v) \; dv \; d\boldsymbol{x} + R_f(\boldsymbol{w}) + \int_{\boldsymbol{x}} \int_v 2\eta (f(\boldsymbol{x}) - h_{\boldsymbol{w}}(\boldsymbol{x})) p(\boldsymbol{x}, v) \; dv \; d\boldsymbol{x}
\end{aligned}$$

where

$$R_f(\boldsymbol{w}) = \int_{\boldsymbol{x}} \int_v (f(\boldsymbol{x}) - h_{\boldsymbol{w}}(\boldsymbol{x}))^2 p(\boldsymbol{x}, v) \; dv \; d\boldsymbol{x}$$

is the risk if $f(\boldsymbol{x})$ ($s(\boldsymbol{x})$ without the noise) was the function to be approximated. Consider the last term of the final expression we had for $R(\boldsymbol{w})$. Denoting $(f(\boldsymbol{x}) - h_{\boldsymbol{w}}(\boldsymbol{x}))$ as $\delta(\boldsymbol{x}, \boldsymbol{w})$ we get

$$\begin{aligned}
\int_{\boldsymbol{x}} \int_v 2\eta\delta(\boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{x}, v) \; dv \; d\boldsymbol{x} &= \int_{\boldsymbol{x}} \int_v 2\eta\delta(\boldsymbol{x}, \boldsymbol{w}) p_s(v|\boldsymbol{x}) p_d(\boldsymbol{x}) \; dv \; d\boldsymbol{x} \\
&= 2 \int_{\boldsymbol{x}} \delta(\boldsymbol{x}, \boldsymbol{w}) \left(\int_v \eta p_s(v|\boldsymbol{x}) \; dv \right) p_d(\boldsymbol{x}) \; d\boldsymbol{x} \\
&= 0 \quad \text{(from Equation 3)}
\end{aligned}$$

Therefore we get that

$$R(\boldsymbol{w}) = \int_{\boldsymbol{x}} \int_v \eta^2 p(\boldsymbol{x}, v) \; dv \; d\boldsymbol{x} + R_f(\boldsymbol{w}) \Rightarrow \arg\min_{\boldsymbol{w}} R(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} R_f(\boldsymbol{w})$$

So minimizing the risk based on the observed values v (with the noise) is the same as minimizing the risk based on the actual target function $f(\boldsymbol{x})$. Note that the first term is the noise variance, which is independent of $\boldsymbol{w}$.

3.4 Density Estimation

Density Estimation is the problem of directly estimating the input probability density function $p_d(\boldsymbol{x})$ of the data source. This problem is often encountered in situations when we need to estimate the noise accompanying a data source and we need to estimate the noise as a probability density function. In this case the 'system' as referred to above is again irrelevant (just as in the Clustering scenario). The training data just consists of several data points $\mathcal{D}_n = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. The hypothesis space $\mathcal{H}$ is the class of all probability density functions that the learning algorithm will potentially consider to pick the 'best' fit among them. We assume that $\mathcal{H}$ is parametrized by a parameter vector $\boldsymbol{w}$. For density estimation we use what is known as the *Maximum Likelihood* principle — picking the density function $h_{\boldsymbol{w}}(\boldsymbol{x}) \in \mathcal{H}$ that is most likely to have generated $\mathcal{D}_n$. The probability with which the point $\boldsymbol{x}_i$ was generated is $h_{\boldsymbol{w}}(\boldsymbol{x}_i)$. The generation probability (called the *likelihood*) for the whole of $\mathcal{D}$ is therefore

$$\prod_{i=1}^{n} h_{\boldsymbol{w}}(\boldsymbol{x}_i)$$

It is often more convenient to work with the negative logarithm of the likelihood (*log likelihood*) — this converts the problem of maximizing the product to minimizing a sum. Therefore

$$\arg\max_{\boldsymbol{w}} \prod_{i=1}^{n} h_{\boldsymbol{w}}(\boldsymbol{x}_i) = -\arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \log h_{\boldsymbol{w}}(\boldsymbol{x}_i)$$

This is equivalent to the following choice of the loss function

$$\mathcal{L}(\boldsymbol{x}, h_{\boldsymbol{w}}(\boldsymbol{x})) = -\log h_{\boldsymbol{w}}(\boldsymbol{x})$$

and using the average of losses across all the data points in $\mathcal{D}_n$ as an approximation for the model risk

$$R(\boldsymbol{w}) = -\frac{1}{n} \sum_{i=1}^{n} \log h_{\boldsymbol{w}}(\boldsymbol{x}_i)$$

4 Learning from Finite Data — Consistency and Uniform Convergence

Two central questions in Machine Learning are

1. *How does one realize the minimum risk $R(\boldsymbol{w})$ as in Equations 1, 2?* This is a significant challenge since the probability density function $p_d(\boldsymbol{x})$ is unknown and the only information available for the learning algorithm to learn from is the finite number of training data points. Strategies to extend observations on finite samples to the entire input space are often referred to as *inductive principles*. We will discuss some common inductive principles along with criteria that inductive principles must satisfy for effective learning, in the sections that follow.

2. The above formulation starts by assuming a *fixed* hypothesis / model space, parametrized by a fixed set of parameters $\boldsymbol{w}$ that the learning algorithm is capable of considering and

picking one from among these to minimize the risk. It is important to realize here that the decision on the class of hypotheses to consider and their parametrization is done *apriori* — the learning algorithm needs a fixed hypothesis space before it starts working with the training data to find the best fit hypothesis. It is not clear why one needs such an apriori assumption about the hypothesis space for learning to start. Assuming we need such an apriori assumption the natural question that follows is "*How can an appropriate hypothesis or model space be chosen for the problem, something with which the learning algorithm will work?*" The strategies used for selecting an appropriate hypothesis space is referred to as *model selection*. We will argue why we need model selection and ways to select models briefly in the sections below.

4.1 Inductive Principles

An inductive principle is a strategy (used by the learning machine) to extend observations from a finite training dataset to the entire domain. An effective inductive principle is expected to be *consistent* or *uniformly convergent*. Consistency of an inductive principle is similar to the law of large numbers in statistics — that under some very general conditions the average of a random variable, no matter what its underlying distribution is, converges to its expected value as the number of i.i.d samples of the random variable becomes very large. We would expect something similar for learning from finite data. To make this formal let's assume that with a loss function $\mathcal{L}$ and hypothesis space parametrized by the parameter vector $\boldsymbol{w}$, $R(\boldsymbol{w}^*)$ is the true risk realized by the 'best' hypothesis with parameters $\boldsymbol{w}^*$ as in Equation 2. Let $\boldsymbol{w}_n^*$ be the 'optimum' hypothesis arrived at using the inductive principle based on a finite i.i.d training sample $\mathcal{D}_n$. Let

$$R(\boldsymbol{w}_n^*) = \int_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, h_{\boldsymbol{w}_n^*}(\boldsymbol{x})).p_d(\boldsymbol{x})\ d\boldsymbol{x}$$
$$R_e(\boldsymbol{w}_n^*) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\boldsymbol{x}_i, h_{\boldsymbol{w}_n^*}(\boldsymbol{x}_i))$$

$R(\boldsymbol{w}_n^*)$ is the true risk (expected loss) of using the hypothesis $\boldsymbol{w}_n^*$ throughout the domain and $R_e(\boldsymbol{w}_n^*)$ average loss, also known as *empirical risk*, among the points in $\mathcal{D}_n$. For inductive learning we expect that both $R(\boldsymbol{w}_n^*)$ and $R_e(\boldsymbol{w}_n^*)$ converge to the true risk as n becomes large. Formally for every probability distribution density function $p_d(\boldsymbol{x})$ it must be that

$$\lim_{n\to\infty} R(\boldsymbol{w}_n^*) = \lim_{n\to\infty} R_e(\boldsymbol{w}_n^*) = R(\boldsymbol{w}^*)$$

Exercise 2 *Argue that always* $\forall n : R(\boldsymbol{w}_n^*) \geq R(\boldsymbol{w}^*) \geq R_e(\boldsymbol{w}_n^*)$.

The relationship between $R(\boldsymbol{w}^*)$, $R(\boldsymbol{w}_n^*)$ and $R_e(\boldsymbol{w}_n^*)$ is illustrated pictorially in Figure 2.

Exercise 3 *Interpret the curves shown in Figure 2. Also explain the initial 'flat' portion (along the n-axis) of the* $R_e(\boldsymbol{w}_n^*)$ *curve.*

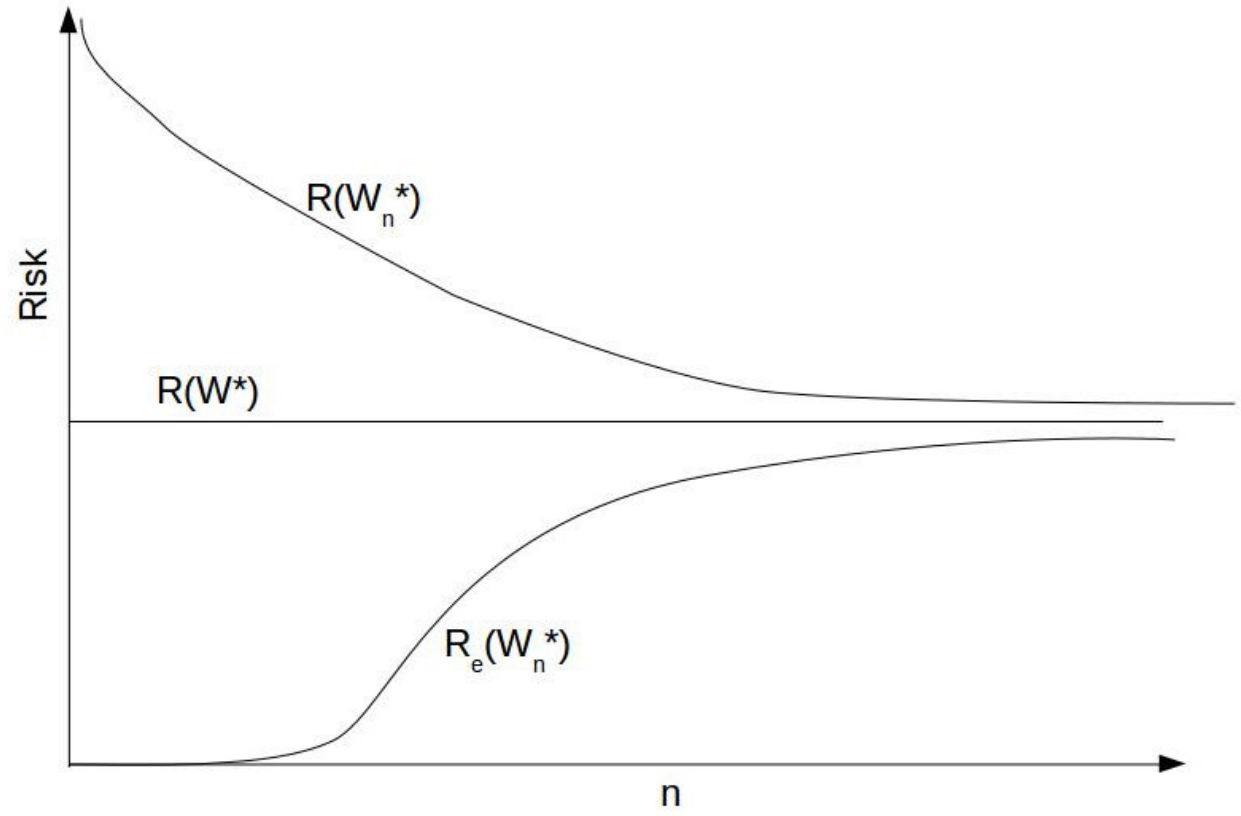

Figure 2: Consistency of Inductive Inference

Exercise 4 *Let $h \in \mathcal{H}$ be a hypothesis belonging to some class of binary classifiers $\mathcal{H}$. Show that*

$$E_{\mathcal{D}_n}[R_e(h)] = R(h)$$

where the expectation on the LHS is over all possible training datasets $\mathcal{D}_n$. Note that $R_e(h)$ is computed over $\mathcal{D}_n$.

We would often need a stronger notion — that the empirical risk not just converges to the true risk, but rather converges uniformly. A model with a loss function is uniformly convergent if any 'large enough' i.i.d sample is representative of the domain in the sense that the empirical risk of the model on the sample is close enough to the true model risk. The formal definition of uniform convergence is given below.

Definition 1 *A learning algorithm $\mathcal{A}$ using a hypothesis class $\mathcal{H}$ with some loss function $\mathcal{L}$ is said to be* uniformly convergent *if there exists a function $m_{\mathcal{H}}(\epsilon, \delta) : \mathcal{R} \times \mathcal{R} \to \mathbb{N}$ such that for every $\epsilon, \delta \in (0, 1)$, for every distribution $p_d(\boldsymbol{x})$ on the input domain $\mathcal{D}$, if $\mathcal{D}_n$ is a training sample of size $n > m_{\mathcal{H}}(\epsilon, \delta)$ then with probability (the probability is taken over all possible training samples of size n) at least $(1 - \delta)$*

$$|R^* - R_e(h^*)| < \epsilon$$

where $h^* = \mathcal{A}(\mathcal{D}_n)$, $R_e(h^*)$ is the average loss for h^* computed over the training sample $\mathcal{D}_n$ and $R^* = \min_{h \in \mathcal{H}} R(h)$.

It is important to make a few observations about this definition.

- The definition incorporates two independent notions – approximating the risk to the desired degree ϵ, and being able to get a good enough approximation with a certain minimum probability $(1-\delta)$. We argue below that both these notions are necessary and neither can be done away with.
- Any learning algorithm that attempts to predict based on a finite sample will certainly make mistakes — it is easy enough to create a distribution that makes the error committed by the model produced by the algorithm, non-zero. The best one can hope for is that we can make the error arbitrarily small by picking a large enough sample.
- It is always (with a non-zero probability) possible to pick a finite i.i.d sample that is not representative of the domain. However larger the sample size less likely is this possibility.

4.1.1 Some Inductive Principles

This section introduces a few well known inductive principles for learning from finite data.

1. *Empirical Risk Minimization (ERM)*: Chooses the hypothesis that minimizes the average loss of the hypothesis on the training data. Vapnik-Chervonenkis Theory provides a very strong theoretical justification for ERM and gives necessary and sufficient conditions for its consistency. Naturally much of statistical learning uses ERM as the key inductive principle to learn from finite data and extend it to unseen test instances.
2. *Bayesian Induction*: Starts with a 'prior' model that encodes the algorithm designer's 'insight' to the problem domain and arrives at a posterior probability for a model given the data using Bayes Theorem.
3. *Stochastic Approximation*: Iteratively refines an initial 'guess' for the final hypothesis taking each data point into account, one by one. Stochastic Approximation also has fairly strong provable conditions that characterize consistency.

We discuss these three inductive principles in a little more detail in later sections on of this document.

4.2 Model Selection

We will examine this issue later in the course, after our discussion on Generalization theory. For now it suffices to say that the general thumb-rule is that we must more-often-than-not use the Occam's Razor — favor simpler/less complex models against the more complex ones. A model that explains everything isn't really a model worth considering. So a complex model that explains (fits) all of the training data given is in fact a bad sign and is often referred to as *overfitting*. The VC-theory we discuss subsequently gives a strong theoretical justification for this often used heuristic. Exercise 13 helps in getting a feel for this phenomenon.

4.3 Bayesian Induction

4.4 Stochastic Approximation

Stochastic Approximation is a way to incrementally account for training data points, one at a time. This does not require all the data points in the training set to be presented at once. This works in a way similar to the way the perceptron algorithm worked — incrementally updating the current best 'guess' for the risk minimizing hypothesis for every new training data point encountered. The update is carried out using a scheme that resembles the gradient descent method for optimization. Assuming a loss function $\mathcal{L}(\boldsymbol{x}, h_{\boldsymbol{w}}(\boldsymbol{x}))$ and appropriate initial (prior) hypothesis $\boldsymbol{w}_0$, the i^{th} iteration of the Stochastic Induction procedure carries out the following update:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \gamma_i \bigtriangledown_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{x}_i, h_{\boldsymbol{w}_i}(\boldsymbol{x}_i)) \tag{4}$$

where γ_i is the *learning rate* and $(\boldsymbol{x}_i, \boldsymbol{v}_i)$ is the i^{th} training data point. This update rule leads to a consistent inductive scheme the loss function satisfies a general convexity condition and the series of γ_is meets the following convergence criteria

$$\lim_{i \to \infty} \gamma_i = 0$$
$$\sum_{i=1}^{\infty} \gamma_i = \infty$$
$$\sum_{i=1}^{\infty} \gamma_i^2 < \infty$$

Note that in this process the same data point can get presented to the learning algorithm multiple times.

Stochastic Induction does not require too much memory, can work in the 'online' mode, and is usually much simpler to implement. One issue (not very unusual for optimization problem) is to effectively decide when the stop updating $\boldsymbol{w}$. One common stopping criterion that is often used is that the process stops when the gradient computed as in Equation 4 is smaller than a threshold. Examples of machine learning problems for which stochastic approximation algorithms are often used are: k-means, Adalines, Perceptrons, Multi-Layer Neural Networks, etc. Refer [**stochasticlearning**] for a nice survey of stochastic learning.

Exercise 5 *Formulate problem of finding a straight line fit through a set of points* $\{(x_1, y_1), \ldots, (x_n, y_n)$ *with a square-error loss function as a stochastic approximation problem. Carry out a data experiment to see how well the model produced fits the data. Compare your results with the results of Exercise 13.*

4.5 Empirical Risk Minimization

The ERM strategy uses the average risk on the training sample as an approximation for the true risk given by Equations 1, 2. Empirical risk $R_e(\boldsymbol{w})$ with a particular hypothesis $h_{\boldsymbol{w}}(\boldsymbol{x})$

and loss function $\mathcal{L}(\boldsymbol{x}, h_{\boldsymbol{w}}(\boldsymbol{x}))$ on a training data set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ is defined as

$$R_e(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(\boldsymbol{x}_i, h_{\boldsymbol{w}}(\boldsymbol{x}_i))$$

and the hypothesis is chosen as

$$\boldsymbol{w}_n^* = \arg\min_{\boldsymbol{w}} R_e(\boldsymbol{w})$$

The VC-Theory gives conditions for the ERM principle to be consistent. The conditions are summarized in the following theorem, that will prove later.

Theorem 1 ***Key Theorem of Learning Theory (Vapnik & Chervonenkis)****: For bounded loss functions the ERM principle is consistent if and only if for any* $\epsilon > 0$

$$\lim_{n\to\infty} Pr\left[\sup_{\boldsymbol{w}} |R(\boldsymbol{w}) - R_e(\boldsymbol{w})| > \epsilon\right] = 0$$

It is interesting to note that key theorem insists that ERM is consistent if and only if the risk associated with the 'worst' hypothesis in the hypothesis space converges to the true risk as n grows large. So effectively convergence does not really depend on the specific hypothesis chosen — either every hypothesis in the hypothesis space is consistent or none is. The 'worst-case' scenario is what matters as far as consistency is concerned.

Here is an informal justification for the Key Theorem stated above. Let $\mathcal{A}$ be an ERM learning algorithm. Note that $\mathcal{A}$ only has access to a finite training dataset $\mathcal{D}_n$ to choose a hypothesis h. However there could be several hypotheses in general all of which agree on all points in $\mathcal{D}_n$ and $\mathcal{A}$ has nothing to choose between them. So we expect that even the hypothesis that has the worst model risk among these will in general be uniformly convergent. Note that the Key Theorem does not hold for Bayesian Induction.

We see a few examples of the use of the ERM principle for some typical machine learning problems.

4.5.1 Gaussian Density Estimation

Consider a d-dimensional training data set $\mathcal{D}_n = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ from which we need to infer the density function of the input distribution. We will carry out a maximum likelihood estimation of the distribution, just as in Section 3.4. We assume a Gaussian fit for the input data. So the hypothesis family we are considering is the class of multidimensional Gaussians, the parameters of the family being the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$. The Gaussian family is given by:

$$\mathcal{N}(\boldsymbol{\mu}; \boldsymbol{\Sigma}) = p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

The generation probability (likelihood) of the training data using the family of Gaussians is therefore

$$p(\mathcal{D}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \left(\frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}}\right)^n \exp\left(-\frac{1}{2}\sum_{i=1}^{n}\left((\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})\right)\right)$$

giving us a negative log-likelihood of

$$-\log p(\mathcal{D}_n|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{nd}{2}\log(2\pi) + \frac{n}{2}\log(|\boldsymbol{\Sigma}|) + \frac{1}{2}\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})$$

Minimizing this with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ we get (using the identities in Exercise 6)

$$\frac{\partial}{\partial\boldsymbol{\mu}}\left(-\log p(\mathcal{D}_n|\boldsymbol{\mu},\boldsymbol{\Sigma})\right) = \boldsymbol{\Sigma}^{-1}\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu}) = \mathbf{0} \Rightarrow \boldsymbol{\mu}_{ML} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i$$

where $\boldsymbol{\mu}_{ML}$ is the maximum likelihood mean. Similarly using $\boldsymbol{\mu}_{ML}$ and taking

$$\frac{\partial}{\partial\boldsymbol{\Sigma}}\left(-\log p(\mathcal{D}_n|\boldsymbol{\mu}_{ML},\boldsymbol{\Sigma})\right) = 0$$

we get

$$\boldsymbol{\Sigma}_{ML} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu}_{ML})(\boldsymbol{x}_i-\boldsymbol{\mu}_{ML})^T \tag{5}$$

Exercise 6 *Prove the following gradient identities involving vectors and matrices.*

$$\frac{\partial}{\partial\boldsymbol{w}}(\boldsymbol{w}^T.\boldsymbol{b}) = \frac{\partial}{\partial\boldsymbol{w}}(\boldsymbol{b}^T.\boldsymbol{w}) = \boldsymbol{b}$$
$$\frac{\partial}{\partial\boldsymbol{w}}(\boldsymbol{w}^T\boldsymbol{A}\boldsymbol{w}) = (\boldsymbol{A}+\boldsymbol{A}^T)\boldsymbol{w}$$

for any constant vector $\boldsymbol{b}$ *and matrix* $\boldsymbol{A}$.

Exercise 7 *Verify Equation 5.*

Exercise 8 *Design and implement an experiment to experimentally verify the maximum likelihood Gaussian fit for a training sample, as demonstrated in this section.*

Exercise 9 *Consider this variant of the Gaussian model fit discussed above. Consider the* 1*-dimensional version of the problem. Suppose* $\{x_1,\ldots,x_n\}$ *is the data set given. We assume that the data has been produced by a mixture of two Gaussian processes — one with its mean and variance unknown and the other Gaussian process with mean at* 0 *and variance* 1*. The family of distributions we are considering is*

$$\mathcal{N}(\mu;\sigma^2) = \frac{1}{2}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right) + \frac{1}{2}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

with parameters μ *and* σ*. Show that there is no maximum likelihood solution with this family of distributions for any given set of data points.* ***Hint****: Consider a possible solution where* $\mu = x_i$ *for some* $1 \le i \le n$*. Check what happens to the* σ *required to maximize the likelihood in this case.*

Exercise 10 *The maximum likelihood estimates arrived at for μ_{ML} and σ_{ML} earlier (assume a 1-dimensional case) are functions of the data points given. Suppose the i.i.d x-values were drawn from a Gaussian distribution with mean μ and variance σ^2. Now treating μ_{ML} and σ_{ML} as random variables depending on random choices of the training data. Show that*

$$E[\mu_{ML}] = \mu$$
$$E[\sigma^2_{ML}] = \left(\frac{n-1}{n}\right)\sigma^2$$

where the expectation is over random choices of training data sets. (Note that the maximum likelihood estimate variance is off from the true variance by a factor of $\frac{1}{n}$.)

4.6 PAC Learning

This section finally introduces the most widely accepted model of learning — *Probably Approximately Correct* (PAC) learning. Before we can define PAC-learnability formally we will first convince ourselves that, at least in the ERM setting, there is no universal learner. That will also tell us why we need an apriori choice of a hypothesis space for any effective learning algorithm.

4.6.1 No Free Lunch Theorem

Here we prove formally that there cannot be a learning algorithm in the sense that given any learning algorithm $\mathcal{A}$ one can always produce a learning task (characterized by an underlying distribution $p_d(\boldsymbol{x})$) on which the algorithm fails. We say the algorithm fails if the model produced by the algorithm does not converge uniformly for the distribution $p_d(\boldsymbol{x})$. This notion is formalized in the No Free Lunch theorem that we state and prove below, for binary classification problems.

Theorem 2 ***No Free Lunch Theorem****: Let $\mathcal{A}$ be any ERM learning algorithm for binary classification on a domain $\mathcal{D}$ with the loss function defined as $\mathcal{L}(\boldsymbol{x}, h(\boldsymbol{x})) = \mathbb{I}(v(\boldsymbol{x}) = h(\boldsymbol{x}))$ where $\mathbb{I}$ is the $0-1$ indicator function and $v(\boldsymbol{x})$ denotes the ± 1 label associated with the data point $\boldsymbol{x}$. Assume that $\mathcal{A}$ has access to training datasets $\mathcal{D}_n$ of size n where $n \leq \frac{|\mathcal{D}|}{2}$. Then there exists a distribution $p_d(\boldsymbol{x})$ over the domain $\mathcal{D}$ such that*

1. *there exists a hypothesis $g^* : \mathcal{D} \to \pm 1$ for which $R(g^*) = 0$, and*
2. $$Pr\left[R_e(g) \geq \frac{1}{8}\right] \geq \frac{1}{7}$$
   where $\mathcal{A}(\mathcal{D}_n) = g$ and the probability is over all possible training data sets $\mathcal{D}_n$ of size $n \leq \frac{|\mathcal{D}|}{2}$.

Proof: We employ a trick often used in learning theory — we take a 'ghost' dataset $\mathcal{D}'_n$ of size n such that $\mathcal{D}'_n \cap \mathcal{D}_n = \phi$. This is possible since $n \leq \frac{|\mathcal{D}|}{2}$. Let's denote $\mathcal{D}'_n \cup \mathcal{D}_n$ as $\mathcal{D}_{2n}$.

1. Let $g^* : \mathcal{D}_{2n} \to \pm 1$ be some labelling function (hypothesis). We now define $p_d(\boldsymbol{x})$ as

$$p_d(\boldsymbol{x}) = \begin{cases} \frac{1}{2n} & \text{if } \boldsymbol{x} \in \mathcal{D}_{2n} \text{ and } v(\boldsymbol{x}) = g^*(\boldsymbol{x}) \\ 0 & \text{otherwise} \end{cases}$$

Clearly for any $\boldsymbol{x} \in \mathcal{D}_{2n}$, $\mathcal{L}(\boldsymbol{x}, g^*(\boldsymbol{x})) = 0$ and for any $\boldsymbol{x} \notin \mathcal{D}_{2n}$, $p_d(\boldsymbol{x}) = 0$. Therefore the model risk for g^* under $p_d(\boldsymbol{x})$ is zero, establishing the first part of the theorem.

2. We first show that there exists a distribution $p_d(\boldsymbol{x})$ for which $E[R_e(\mathcal{A}(\mathcal{D}_n))] \geq \frac{1}{4}$, where the expectation is over all possible training data sets $\mathcal{D}_n$ of size $n \leq \frac{|\mathcal{D}|}{2}$. It is a simple exercise using Markov Inequality to show that this implies the second part of the No Free Lunch Theorem — Exercise 11. We need to show that

$$E_{\mathcal{D}_n}[R_e(\mathcal{A}(\mathcal{D}_n))] = E_{\mathcal{D}_n}\left[E_{\boldsymbol{x}\in\mathcal{D}}[\mathcal{L}(\boldsymbol{x}, g(\boldsymbol{x}))]\right] \geq \frac{1}{4}$$

Going by the Key Theorem (Theorem 1) and the argument following the key theorem, it is enough for us to show the above claim for the 'worst' hypothesis among those that minimize the empirical error within $\mathcal{D}_n$. We will in fact prove a stronger claim — that the expectation taken across all model choices that agree on $\mathcal{D}_n$ satisfies the $\frac{1}{4}$ bound. Therefore we show that

$$E_g\left[E_{\mathcal{D}_n}\left[E_{\boldsymbol{x}\in\mathcal{D}}[\mathcal{L}(\boldsymbol{x}, g(\boldsymbol{x}))]\right]\right] \geq \frac{1}{4}$$

where the outermost expectation is over all the hypotheses g such that $R_e(g) = 0$. Flipping the order of expectations, this is equivalent to showing that

$$E_{\mathcal{D}_n}\left[E_{\boldsymbol{x}\in\mathcal{D}}\left[E_g[\mathcal{L}(\boldsymbol{x}, g(\boldsymbol{x}))]\right]\right] \geq \frac{1}{4} \tag{6}$$

We show that $p_d(\boldsymbol{x})$ is the required distribution for which the claim in the statement of the theorem holds. We assume henceforth in the proof that the underlying distribution is $p_d(\boldsymbol{x})$. Clearly model risk and the empirical risk associated with g^* is zero. Since $\mathcal{A}$ is an ERM machine, g in the innermost expectation in Equation 6 is over all the hypotheses that agree with g^* on the whole of $\mathcal{D}_n$, i.e., $\forall \boldsymbol{x} \in \mathcal{D}_n : \mathcal{L}(\boldsymbol{x}, g(\boldsymbol{x})) = 0$ and the empirical risk associated with g is zero. Considering the two inner expectations of Equation 6 we get

$$\begin{aligned} E_{\boldsymbol{x}\in\mathcal{D}}\left[E_g[\mathcal{L}(\boldsymbol{x}, g(\boldsymbol{x}))]\right] &= \frac{1}{2n}\left(\sum_{\boldsymbol{x}\in\mathcal{D}_n} E_g[\mathcal{L}(\boldsymbol{x}, g(\boldsymbol{x}))] + \sum_{\boldsymbol{x}\in\mathcal{D}'_n} E_g[\mathcal{L}(\boldsymbol{x}, g(\boldsymbol{x}))]\right) \\ &= \frac{1}{2n}\sum_{\boldsymbol{x}\in\mathcal{D}'_n} E_g[\mathcal{L}(\boldsymbol{x}, g(\boldsymbol{x}))] \\ &= \frac{1}{4} \end{aligned}$$

Note that for any given point $\boldsymbol{x} \notin \mathcal{D}_n$, since g is 'unaware' of anything outside $\mathcal{D}_n$, a random g can assign either +1 or −1 with probability $\frac{1}{2}$ each. The expected loss at $\boldsymbol{x}$ is therefore $\frac{1}{2}$, the probability of the event $(g(\boldsymbol{x}) = v(\boldsymbol{x}))$. The expected risk across $\mathcal{D}'_n$ is therefore $\frac{n}{2}$. This bound holds for any $\mathcal{D}_n \subset \mathcal{D}$. Therefore we get

$$E_{\mathcal{D}_n}\left[E_{\boldsymbol{x}\in\mathcal{D}}\left[E_g[\mathcal{L}(\boldsymbol{x}, g(\boldsymbol{x}))]\right]\right] \geq E_{\mathcal{D}_n}\left[\frac{1}{4}\right] \geq \frac{1}{4}$$

■

The No Free Lunch Theorem essentially tells us that an all powerful learner is actually useless. We will need to put some apriori restrictions on the hypothesis space (that will in turn avoid the kind of constructions outlined in the No Free Lunch Theorem). The question we therefore ask is

Definition 2 *A hypothesis class $\mathcal{H}$ with parametrization $\boldsymbol{w}$ is said to be* PAC-Learnable *with respect to a risk function $R(\boldsymbol{w})$ (for a given loss function $\mathcal{L}$ and true risk $R(\boldsymbol{w}^*)$) if there exists a learning algorithm $\mathcal{A}$ and a function $m_{\mathcal{H}}(\epsilon,\delta) : \mathcal{R}\times\mathcal{R} \to \mathbb{N}$ such that for every $\epsilon,\delta \in (0,1)$, for every distribution $\mathcal{D}$ on the input domain, if $\mathcal{A}$ is run on a training sample $\mathcal{D}_n$ of size $n > m_{\mathcal{H}}(\epsilon,\delta)$ then it returns a hypothesis $\boldsymbol{w}_n^*$ such that*

$$Pr\left[|R(\boldsymbol{w}^*) - R_e(\boldsymbol{w}_n^*)| \leq \epsilon\right] \geq (1-\delta)$$

where the probability is over all possible training samples $\mathcal{D}_n$. The number $m_{\mathcal{H}}(\epsilon,\delta)$ is called the Sample Complexity*.*

Exercise 11 *Use Markov's Inequality (Theorem 8 in the prerequisite document on Probability Theory) to prove the following tail bound: Let X be any random variable that takes values in the range $[0,1]$ with expected value $E[X] = \mu$. Show that for any $0 < a < 1$*

$$Pr(X > 1-a) \geq \frac{\mu-(1-a)}{a}$$
$$Pr(X > a) \geq \frac{\mu-a}{1-a} \geq \mu - a$$

Exercise 12 *Given a set of training data points $\{(x_1,y_1),\ldots,(x_n,y_n)\}$ in 2-d and a model $y = f(x)$ to fit the data in, consider the loss function $\mathcal{L}(y,f(x)) = (y-f(x))^2$. Derive the expression for the coefficients of $f(x)$ in terms of the data points, that will minimize the Empirical Risk, for the following two cases.*

1. *The function $f(x)$ is a quadratic of the form ax^2+bx+c.*
2. *The function $f(x)$ is a linear function of the form $ax+b$.*

Exercise 13 *Carry out the following data experiment. Consider the function $y = f(x) = x^2+\eta$ where η is a Gaussian random variable with zero mean and say $\sigma^2 = 0.3$. Do the following:*

1. *Generate a set of say 10 points on the plane using $f(x)$ with x picked uniformly at random from the interval $[0,1]$ and η sampled from a Gaussian distribution.*
2. *Fit a quadratic $y = q(x)$ on the ten points using the result of Exercise 12.*
3. *Fit a line $y = s(x)$ on the ten points using the result of Exercise 12.*

4. *Compute the mean square error loss on the training set with both $q(x)$ and $s(x)$. Compare the two.*

5. *Generate a test set from $f(x)$ and measure the mean square error loss on both $q(x)$ and $s(x)$. Compare the two.*

6. *Repeat all the above steps say a 100 times. Do you see a trend?*

7. *Try the above experiment with different sizes of training sets. Do you see a pattern in the way the losses of $q(x)$ and $s(x)$ change with the training set size?*

Exercise 14 *In the above experiment remove the Gaussian noise term η and fit a quadratic or a linear function through the training set. How different is this from the fit arrived at in the earlier exercise with noise? Interpret your results in the light of what we established in Section 3.3.*

Exercise 15 ***Monotonicity of Sample Complexity**: Show that the sample complexity $m_{\mathcal{H}}(\epsilon, \delta)$ for a PAC learnable hypothesis class is non-increasing in both ϵ and δ. In particular show that for any $0 < \epsilon_1 \leq \epsilon_2 < 1$, $m_{\mathcal{H}}(\epsilon_1, \delta) \geq m_{\mathcal{H}}(\epsilon_2, \delta)$. Similarly for any $0 < \delta_1 \leq \delta_2 < 1$, $m_{\mathcal{H}}(\epsilon, \delta_1) \geq m_{\mathcal{H}}(\epsilon, \delta_2)$.*