Introduction to Machine Learning (CS419)

Instructor: Saketh

Contents

Chapter 1

Basic Concepts and Definitions

Machine learning aims at developing algorithms that mimic the ability in humans to *learn* i.e., improve their "performance" with experience. By performance, we mean their various cognitive abilities. A short note about this is presented below. It is easy to observe that machine learning algorithms will have far reaching consequences in all aspects of living and hence are worth developing.

1.1 Human Learning

Humans seem to have various cognitive abilities. Some of which[1] are: i) finding similarities or patterns or groupings in data, ii) categorizing objects not seen before as novel iii) sequence completion iv) recognition abilities including speech and object recognition v) Summarizing or abstracting text/images/multi-media vi) Decision making vii) Problem solving etc. A little thought will convince that all these abilities improve[2] with increasing experience.

Above discussion convinces that associated with learning there is always a target/unknown concept/ability. Hence-forth, we will call this as the **unknown-concept**. For e.g., in the phenomenon of learning to group data, the unknown concept is the relationship between data/objects and group-ids. In the phenomenon of speech recognition, it is the relationship between speech utterances and English transcriptions etc.

Needless to say, learning happens through experience. Hence **experience** is an important aspect of learning. It is easy to see that **experience** is simply a finite-sized realization (sampling) of the truth in the **unknown-concept**. Depending on

[1]Examples are picked based on the settings that machine researchers have formally studied.
[2]Not necessarily strictly monotonically improving.

the need/application humans express[3] the **unknown-concept** through some **input-output** pairs. For e.g., in the grouping/clustering of data example, the input is the datum and the output is the group-id etc. This clarifies what we mean by **input** and **output**.

Also, how well or fast will a person learn will definitely depend on his current state/capacity/bias/background. We will refer to this as **background** hence-forth. Given this terminology, one could say that the objective in human learning is to determine the **unknown-concept**, but it seems enough to say that the goal is to be able to provide the "right" output for *any* input (because this is how usually humans express their ability/unknown-concept that has been learnt). Summarizing, here are the five important aspects in human learning:

1. **Input** and **Output**
2. **Unknown-concept**
3. **Experience**
4. **Background**
5. **Objective of learning**

1.2 Machine Learning

Though humans possess very many abilities, they are currently far from understanding how they learn/acquire/improve these abilities. So the idea in machine learning is to develop mathematical models and algorithms that mimic human learning rather than understanding the phenomenon of human learning and replicating it. Hence, we begin by writing down the mathematical concepts by which we represent each of the five aspects in human learning mentioned above.

1. **Input** by $x \in \mathcal{X} \subset \mathbb{R}^n$ and[4] **Output** by $y \in \mathcal{Y} \subset \mathbb{R}$.
2. **Unknown-concept** by a Probability distribution [Ross, 2002] over the inputs (hence-forth denoted by U_X) or over the input-output pairs (hence-forth denoted by U_{XY}). For e.g., in case of support/mean/density estimation or

[3]If humans could express the unknown-concept directly, rather than in terms of input-output pairs, then perhaps, humans would also understand how they learn!

[4]Euclidean model for input space and real space for output is what we employ in CS419. Machine learning, in general, is not restricted to these.

sequence filling problems the probability distribution is over the inputs alone and in case of grouping/clustering data and speech/object recognition problems the probability distribution is over the input-output pairs.

3. **Experience** by:

   (a) set of input-output pairs. In this case learning is said to be **supervised**.

   (b) set of inputs. In this case learning is said[5] to be **unsupervised**.

   In either case, we call this set as the **training set**. Most importantly, the training set and the unknown distribution have a relation: we assume that the training set is an instantiation of a set of iid random samples from the unknown distribution. We denote the set of iid random variables generating the training set as $D = \{X_1, \ldots, X_m\}$ in un-supervised case and as $D = \{(X_1, Y_1), \ldots, (X_m, Y_m)\}$ in the supervised case. The training set is an instantiation of D (hence-forth denoted by $\mathcal{D}$).

4. **Background** by (mathematical) Model. Models are of two types:

   (a) Deterministic Models: Linear models (set of all linear functions [Sheldon Axler, 1997] over the input space), Quadratic models (set of all quadratic functions over the input space), Analytic models (set of all analytic functions over the input space), Convex models (set of all convex functions [Boyd and Vandenberghe, 2004] over the input space) etc.

   (b) Probabilistic Models [Ross, 2006][6]: Bernoulli models (set of all Bernoulli distributions over the input space), Gaussian models (set of all Gaussian distributions over the input space).

   Note that the members of each model differ only in terms of their parameters. These parameters are hence called as **model parameters**. For eg. for linear models $\theta \in \mathbb{R}^n$ are the parameters and for Gaussian models the mean vector and covariance matrix are the parameters. We will reserve $\mathcal{M}$ for denoting model and θ for denoting the vector of model parameters.

5. **Objective of learning** by some mathematical statement (discussed below).

[5]In lecture, we attempted to categorize some eg. of human learning into supervised and unsupervised and realized that in some cases this is clear and some cases it is not clear. This means that one should consider other paradigms too in machine learning than these two. However in CS419 we will only focus on these two. Other paradigms are semi-supervised learning, active learning, reinforcement learning etc.

[6]With these models, one can give answers like it is very likely to rain today, which humans do.

Broadly, there are two schools of thought when it comes to formalizing the objective of learning. The first is to view learning as the process of selecting the "best" model parameter after observing the training set (given the model). In this case the output for a given input is predicted/estimated/inferred only using this best model parameter. The second is to view learning as the process of estimating the uncertainty of each model parameter being the "right" one and then the output is estimated/predicted/inferred (for a given input) by taking an aggregate opinion of each model parameter[7]. The latter approach is called as Bayesian learning. We begin with the former set-up of non-Bayesian learning and then provide a description of Bayesian learning.

1.2.1 Non-Bayesian Set-up for Learning

Recall that here, the goal is to pick the "best" model parameter that gives good "performance". Here is a natural ways of writing down this goal mathematically in the case of deterministic models for supervised learning:

$$\min_{\theta\in\vartheta} \quad R[g_\theta], \tag{1.1}$$

where ϑ is the set of possible values the model parameter, θ, could take, and g_θ is the particular member of the deterministic model (i.e., member of the set of functions) with θ as the parameter[8]. $R[g_\theta]$ is the **risk** incurred by employing g_θ as the "best" hypothesis for prediction. Further, we define risk as the expected **loss**, where loss is any non-negative function, $l : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$ that measures how far the predicted and true output are for the given input with the given function i.e., $R[g_\theta] \equiv \mathbb{E}\left[l(g_\theta(X), Y)\right]$. Examples of loss function[9] are: i) 0-1 loss: $l(g_\theta(X), Y) \equiv 1_{g_\theta(X)=Y}$ ii) square-loss: $l(g_\theta(X), Y) \equiv (g_\theta(X) - Y)^2$. Needless to say, one cannot even compute this expectation as the distribution (U_{XY}) is unknown and hence minimizing it is impossible. Later on we will see how to approximate this goal using the training set.

In this case, given a x, the corresponding y is predicted using $y = g_{\theta^*}(x)$, where θ^* is the minimizer of the objective in (1.1).

In case of probabilistic models for unsupervised learning with U_X as the unknown distribution, here is a natural way of writing down the goal:

$$\min_{\theta\in\vartheta} \int_{\mathcal{X}} \left(F_\theta(x) - U_X(x)\right)^2 \, \mathrm{dx}, \tag{1.2}$$

[7]To better understand this statement, you may want to view a model as a set of hypothesis and each model parameter corresponds to a particular hypothesis in this set.

[8]In other words, the model is the set of all g_θ for various $\theta \in \vartheta$.

[9]Later on we will provide more examples of loss functions.

where F_θ is the particular distribution in the model with the parameter as θ. In plain words, this simply picks that θ that induces the closest distribution to the unknown one. Further, assuming that for all the distributions in the model and U_X the moment-generating-function (mgf) exists, here is another way of writing the goal.

$$\min_{\theta \in \vartheta} \sum_{i=1}^{\infty} \left(\mathbb{E}_{U_X}[X^i] - \mathbb{E}_{F_\theta}[X^i] \right)^2 \tag{1.3}$$

In plain words, this simply picks that θ that induces the distribution whose moments are closest to the unknown one.

1.2.2 Bayesian Set-up for Learning

Recall that here, the goal is to get the "right" distribution over the model paramters given the training set such that the distribution reflects the probability that a model parameter is "correct". In other words, obtaining the "right" $F(\theta/\mathcal{D})$ is the objective. Ofcourse now one can again resort to the set-up in non-Bayesian and assume a model over θ etc. and then try to pick the "best". However, such a method would not come under Bayesian set-up. In the subsequent chapter, we will show that Bayes rule can be employed to compute this **posterior distribution**[10] of θ.

In the subsequent chapter we will present a summary of well celebrated algorithms that approximate/achieve the objective of learning in the various settings illustrated above.

[10]Posterior distribution because it is the distribution *after* observing something .... (here it is the distribution of θ after seeing the training set).

Chapter 2

Overview of Machine Learning Algorithms

Note that the objective in case of Bayesian set-up is achievable/realizable (exactly), ofcourse assuming that the prior distribution over the model parameters is known. However the objective in the non-Bayesian set-up is not achievable (impossible to achieve). Hence the non-Bayesian algorithms need a special introduction.

2.1 Supervised Learning with Deterministic Models

Here we present algorithms for supervised learning with deterministic models. In this case the non-Bayesian set-up is natural[1].

Lets start with the problem (1.1). One way to devise a algorithm is by looking at a quantity computable from the training set and approximates the objective in (1.1). From weak law of large numbers it is clear that the so-called **empirical risk**, which is defined as the average loss over the training set i.e., $\hat{R}_m[g_\theta] \equiv \frac{1}{m}\sum_{i=1}^{m} l(g_\theta(x_i), y_i)$, is a probably-approximately-correct (PAC) approximation of the risk i.e., for a given θ the sequence of random variables (with increasing m) $\left\{\frac{1}{m}\sum_{i=1}^{m} l(g_\theta(X_i), Y_i)\right\}$ converge in probability to $R[g_\theta]$. This observation motivates the so-called **Empirical Risk Minimization (ERM)**:

$$\min_{\theta\in\vartheta} \quad \hat{R}_m[g_\theta], \tag{2.1}$$

[1]Bayesian set-up is natural with probabilistic models only.

Note that, unlike (1.1), this minimization is not impossible to compute[2].

While this is encouraging, the immediate question is whether the minimum of empirical risk is PAC version of minimum of risk ? We say ERM is **statistically consistent** iff this happens. Fortunately, for the loss functions and models we use in this course, this can be shown [Schölkopf and Smola, 2002, Vapnik, 1998].

2.2 Unsupervised Learning with Probabilistic Models

We begin with a discussion for non-Bayesian set-up and study Bayesian set-up sub-subsequently.

2.2.1 Non-Bayesian Set-up

Now lets consider the problem (1.3). Again motivated by law of large numbers the following algorithm, often called as the **method of moments**, is immediate:

$$\min_{\theta\in\vartheta}\sum_{i=1}^{\infty}\left(\frac{1}{m}\sum_{j=1}^{m}[x_j^i]-\mathbb{E}_{F_\theta}[X^i]\right)^2 \tag{2.2}$$

Under some mild regularity conditions one can show that the method of moments is statistically consistent [Hansen, 1982].

Infact, one can easily generalize this algorithm: suppose $U_X \in \mathcal{M}$ i.e., there exists $\theta^* \in \vartheta$ such that $F_{\theta^*} = U_X$. Lets also assume no two distributions in the model are the same. And, suppose there exist n functions $g_i : \mathcal{X} \times \vartheta \mapsto \mathbb{R},\ i = 1,\ldots,n$ and $g = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix}$ such that $\mathbb{E}_{U_X}[g(X,\theta)] = 0$ if and[3] only if $\theta = \theta^*$. We define the **method of generalized moments** as:

$$\min_{\theta\in\vartheta}\sum_{i=1}^{n}\left(\frac{1}{m}\sum_{j=1}^{m}g_i(x_j,\theta)\right)^2 \tag{2.3}$$

[2]This does not mean that (2.1) can be solved in polynomial time. Infact even with 0-1 loss and linear model, this problem is computationally hard [Feldman et al., 2009].

[3]Note that we define expectation of a vector to be the vector of expectations.

Note that if we choose $g_i(X,\theta) \equiv X^i - \mathbb{E}_{F_\theta}[X^i], i = 1, 2, \ldots$, then we will get back the method of moments. Hence this method indeed generalized the earlier one.

From the literature of information theory, we know that $g(X,\bar{\theta}) = \nabla_\theta \ln(f_\theta(X))|_{\theta=\bar{\theta}}$, known as the Fisher information[4], satisfies[5] the required zero expectation criteria mentioned above i.e., the expectation of gradient of log-likelihood function is zero only with the correct model parameters (θ^*). Here, and hence-forth, we denote the pmf/pdf of F_θ by f_θ and call it the **likelihood**[6]. Note that with this choice of g, the method of generalized moments is:

$$\min_{\theta\in\vartheta} \|\nabla_\theta \ln(f_\theta(\mathcal{D}))\|^2 \tag{2.4}$$

where $f_\theta(\mathcal{D})$ is the likelihood of the training set, $\mathcal{D}$, wrt. the F_θ distribution.

Motivated by the above, we write the following algorithm, popular as the maximum likelihood algorithm:

$$\max_{\theta\in\vartheta} \ln(f_\theta(\mathcal{D})) \tag{2.5}$$

While the above algorithm[7] is very intuitive, it simply says the best distribution is the one with which the likelihood of seeing the training data is the most, it is interesting that under some sufficient conditions (2.4), the generalized method of moments, and (2.5), the maximum likelihood , are the same: i) log-likelihood as function of θ is concave and ii) ϑ is an open set.

We finish this section with another algorithm that again is closely related to the maximum likelihood algorithm. Given two likelihood functions, f_1, f_2, one way of computing inner-product Sheldon Axler [1997] is: $\langle f_1, f_2\rangle \equiv \mathbb{E}_{f_1}[f_2(X)] = \mathbb{E}_{f_2}[f_1(X)]$, the expected likelihood Jebara et al. [2004]. Needless to say, given an inner-product, a distance/norm function is induced: $\|f_1 - f_2\|^2 \equiv \langle f_1, f_1\rangle + \langle f_2, f_2\rangle - 2\langle f_1, f_2\rangle$. This motivates the following objective for learning: $\min_{\theta\in\vartheta} \|f_\theta - U_\theta\|^2 = \min_{\theta\in\vartheta} \mathbb{E}_{f_\theta}[f_\theta(X)] - 2\mathbb{E}_{U_X}[f_\theta(X)]$. Again, motivated by law of large numbers, we propose the following algorithm for minimizing the above objective:

$$\min_{\theta\in\vartheta} \mathbb{E}_{f_\theta}[f_\theta(X)] - 2\sum_{i=1}^{m} f_\theta(x_i). \tag{2.6}$$

Interestingly, the first term is a term independent of training data and is fixed given the model. For e.g., for exponential distribution, $f(x) = \lambda \exp -\lambda x, \lambda > 0$, this

[4]For this course the Wikipedia article is enough: `http://en.Wikipedia.org/wiki/Fisher_information`.

[5]$\nabla_x g(x)$ denotes the gradient of g wrt x i.e., the vector of partial derivatives of g wrt. x_is.

[6]Instead of saying probability with discrete RVs and density with conts. RV, machine learning/statistics community uses likelihood term for both kinds of RVs.

[7]Note that the θ^* that maximizes likelihood also maximizes log-likelihood and vice-versa.

term is simply $\lambda/2$. While this first term is minimized, the second term (without the minus sign) is the sum of likelihoods of training examples, which is maximized. Hence the objective is not only performing some kind of maximum likelihood, but also preferring low-energy[8] models. Such terms in the objective that do not depend on training set are called as **regularizers**. We also, note that, without the first term, this algorithm maximizes the arithmetic mean of likelihoods, whereas the maximum likelihood algorithm maximizes the geometric mean of the likelihoods (and AM$\geq$GM). This is another way to motivate the popular maximum likelihood algorithm. In the subsequent section, we shift our focus to the Bayesian set-up.

Note that all methods/algorithms mentioned above approximate the true expectation/mean/risk wrt. the unknown distribution with the sample average/mean/empirical-risk. And then hope[9] for statistical consistency. Such algorithms that depend on the notion of probability as frequency as called as **Frequentist** algorithms[10].

2.2.2 Bayesian Set-up

As mentioned earlier, unlike Frequentists, here we will employ Bayes rule[11] to obtain $F(\theta/\mathcal{D})$:

$$f_{\Theta/D}(\theta/\mathcal{D}) = \frac{f_{D/\Theta}(\mathcal{D}/\theta) f_{\Theta}(\theta)}{f_D(\mathcal{D})}. \tag{2.7}$$

Observe that given f_Θ, usually called as **prior**[12], $f_{\Theta/D}$ can be computed. Also, for prediction, we aggregate over all the model parameters using the total probability rule: consider the U_X case, then $f_{X/D}(x/\mathcal{D}) = \sum_{\theta\in\vartheta} f_{X/\Theta}(x/\theta) f_{\Theta/D}(\theta/\mathcal{D})$ if the posterior distribution over θ is discrete and $f_{X/D}(x/\mathcal{D}) = \int_\vartheta f_{X/\Theta}(x/\theta) f_{\Theta/D}(\theta/\mathcal{D})\, \mathrm{d}\theta$ if the posterior distribution over θ is continuous. Hence-forth we will refer to $f_{X/D}$ as the **Bayesian Average Model (BAM)**.

2.2.3 Other Hybrids

Once the posterior distribution for the model parameters is determined, instead of using it for computing the posterior of x given $\mathcal{D}$, sometimes it used to determined

[8]Energy is simply integral of the function-squared, the first term. For exponential distribution, the energy is simply $\lambda/2$. The algorithm thus prefers exponential distributions that decay slowly!

[9]Ofcourse, the goal is to prove consistency.

[10]Note that ERM is also a frequentist algorithm.

[11]Recall that Bayes rule can be applied to *any* joint distribution where all the conditionals and marginals are either discrete or continuous distributions.

[12]The distribution prior to seeing the training data.

the “best” parameter and used for prediction (like in non-Bayesian set-up). For e.g., compute the mode of the posterior $f_{\Theta/D}$ and use it as the “best” parameter, popularly known as the **Maximum Aposteriori Algorithm (MAP)**; or compute the mean or median etc. Such half-way Bayesian approaches are often called as posterior plug-in approximations.

If one now asks the question which of these methods MLE, MAP, BAM may have better predictive performance, then my answer would be: simply because MAP and BAM utilize domain knowledge, which is a superior form of knowledge/wisdom, they may perform better (ofcourse their performance is doubtful when there is no such explicit domain knowledge) than MLE. In between MAP and BAM, I would intuitively say, if the model has the distribution that is equal or close to the unknown distribution being modeled, then MAP or some other such plug-in approximation based on the posterior is perhaps better. In anycase, we will be able to answer this question better only in section 2.4.

2.3 Supervised Learning with Probabilistic Models

Algorithms for this case are exactly same as those in the previous section. However, the modeling is different and infact multiple choices for modeling exist when dealing with supervised learning. The following text explains these choices.

In supervised learning, U_{XY} is the unknown distribution, $\mathcal{D} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, and the usual goal is to estimate the conditional distribution $U_{Y/X}$. Now this can be estimated by two fundamentally different types of models:

Discriminative Models: Model $U_{Y/X}$ directly[13]. Needless to say, this is very natural and intuitive.

Generative Models: Model U_{XY}, the unknown distribution and then use Bayes rule for obtaining $U_{Y/X}$ from it[14]. At first this seems an overkill, but as summarized in section 8.6 in Murphy [2012], both generative and discriminative models have their own trade-offs. Further, there are two different choices in modeling U_{XY}:

1. Model U_{XY} directly.
2. Model two things: $U_{X/Y}$ and U_Y. Needless to say, given these two, U_{XY} is fixed and vice-versa.

[13]Model for $U_{Y/X}$ is a set of distributions for Y/X for all $X = x$.

[14]Note that from U_{XY} one can obtain $U_{Y/X}$ however the converse is impossible.

Ofcourse, any of Frequentist, Bayesian and other plug-in approximations may be employed for learning with any of the above types of models. Through various examples we will later realize the conveniences with each of these three different types of supervised models.

We end this section with some terminology: the supervised problem where Y is a discrete set[15] is called as **classification** problem. And more specifically, if Y is a discrete set with exactly two members, then it is called as a **binary classification** problem. Typically, when employing a probabilistic model for predicting the label/output for a given x with a learnt model, one resorts to what is popularly known as **Maximum Aposteriori Inference/Prediction (MAP Inference/Prediction)** i.e. the predicted label of x is given by $\text{argmax}_{y \in \mathcal{Y}} f(y/x)$, where f is the estimate for $U_{Y/X=x}$. This MAP derived function that provides the prediction $g(x) = \hat{y}$ is called the **classifier**. If in a classification problem one employs the third type of model i.e., model both $U_{X/Y}$ and U_Y, then the resulting classifier is called as the **Bayes classifier**. Further, if one employs the Bayesian average model for both $U_{X/Y}$ and U_Y, then we qualify the classifier as **Bayesian Bayes Classifier**. Sometimes, in addition to making the assumption that the examples in the training set are independent, one makes the assumption that the features/dimensions of the data $X \in \mathbb{R}^n$ are independent. If a Bayes classifier makes such an assumption i.e., assume $U_{X/Y} = U_{X_1/Y} \dots U_{X_n/Y}$, then the resulting classifier is called a **Naive Bayes classifier**.

If $Y = \mathbb{R}$ (with the usual Euclidean space for $\mathbb{R}$), then the supervised problem is called as a **regression** problem.

2.4 Model Selection

Till now we assumed that a model is given (and we were mimicking the learning in a particular human being). The obvious question to be answered is what to do if multiple models are available or if the model under consideration does not contain the unkown distribution? This section is dedicated to a discussion on this question. In the following text, we use model and hyperparameter interchangeably.

We begin with the scenario where multiple models $m_1, \dots, m_r$ are given or equivalently, hyperparameters $\alpha_1, \dots, \alpha_r$ are given. In this case, you may already observe the following fact: previously we assumed hyperparameter/model was given i.e., the set of parameters to choose from was given and we discussed how to choose the parameters from this fixed set. Now we are fixing the set of models or equivalently assuming that the set of hyperparameters to choose from is known and asking

[15]It is assumed that the members of this discrete set are NOT comparable.

how to choose the hyperparameter (and parameter) given this fixed set. So essentially, whatever ideas we used for parameter selection or Bayesian averaging, can be used again; with the only difference being that the hyperparameters now play the role of parameters and perhaps we should call those playing the role of hyperparameters are hyperhyperparameters. Needless to say, this process can be done recursively. In the following we will begin with some methods that are essentially equivalent to the ideas in parameter selection. However, later we will present a statistical learning theory result that introduces a new component to the usual objective of training-data fit. This will motivate us to devise a new class of algorithms (please refer to structural risk minimization in the following for that).

Let's first start with simple algorithms for model/hyperparameter selection/averaging that are similar in spirit to parameter selection/averaging: a Bayesian's answer would be to simply take a weighted opinion of all the models under consideration. For this you may use domain knowledge that gives a prior[16] over models/hyperparameters. We can write an expression[17] for the Bayesian average model: $p(x/\mathcal{D}) = \sum_{i=1}^{r} p(x/m_i, \mathcal{D})p(m_i/\mathcal{D})$ or equivalently, $p(x/\mathcal{D}) = \sum_{i=1}^{r} p(x/\alpha_i, \mathcal{D})p(\alpha_i/\mathcal{D})$, where $p(\alpha_i/\mathcal{D})$ is the posterior distribution of the hyperparameters that $\propto p(\mathcal{D}/\alpha_i)p(\alpha_i)$, i.e., proportional to the product of the **marginal likelihood** of the data under a model/hyperparameter and the prior of the hyperparameter/model. Further, $p(x/\alpha_i, \mathcal{D})$ is essentially the BAM for a given hyperparameter/model (already familiar to you) and the expression for marginal likelihood is ofcourse: $p(\mathcal{D}/\alpha_i) = \int_{\theta \in m_i} p(\mathcal{D}/\theta)p(\theta)\ \mathrm{d}\theta$ (average opinion of all distributions in the model m_i). Hence this algorithm can be called Bayesian average of Bayesian average models.

While this is the case with a Bayesian, a non-Bayesian would argue for selecting the "best" hyperparameter. Here are some obvious alternatives:

- An algorithm that is analogous to the MAP estimation of parameter selection: the idea is to use the posterior distribution of hyperparameters: $p(\alpha_i/\mathcal{D})$ and pick the one that maximizes it, say $\hat{\alpha}$. Now given this "best" hyperparameter there are again options to predict using this model: we can do the usual BAM or MAP or MLE based parameter selection and subsequent prediction.

- An algorithm that is analogous to the MLE estimation of parameter selection: the idea is to use the marginal likelihood of hyperparameters: $p(\mathcal{D}/\alpha_i)$ and pick the one that maximizes it, say $\hat{\alpha}$. Now given this "best" hyperparameter there are again options to predict using this model: we can do the usual BAM or MAP or MLE based parameter selection and subsequent prediction. Such a

[16]The parameters of this prior distribution over hyperparameters may be called as hyperhyperparameters.

[17]We can also write the expression for the case where the set of hyperparameters is an interval etc.; the sum needs to be replaced by an integral.

model selection algorithm is known as the **maximum marginal likelihood** algorithm.

- Another simple algorithm is to pick that parameter from all (the union of) the models that maximizes the likelihood of the training data. This is equivalent to doing MLE in the union of all models.

Note that each of the four types of algorithms stated above may give different predictive functions. In particular, the method of maximum marginal likelihood is not equivalent to the last one (simple MLE over the union). This is because the marginal likelihood is the likelihood's average over the entire model and hence the model for which it is maximum need not be same as that model which contains the parameter/distribution that maximizes the likelihood of the data. Infact, it is easy to see that the maximum marginal likelihood and the algorithm mentioned above it inherently penalize big/complex models! Please read section 5.3 in Murphy [2012] for a detailed discussion regarding this phenomenon.

This is a key observation that merits a discussion: the simple MLE algorithm achieves the best fit to the training data because it essentially searches a bigger model; whereas the others penalize big models; in the sense that bigger the model, it is more probable that the likelihood, with the corresponding predictive distribution obtained by the other three algorithms, is less. While this is the case, at the first look, one might argue that since bigger models have a better chance to contain the unknown distribution/function being modeled, than smaller models, the simple MLE algorithm is superior. However, results such as the following[18] hint that bigger models need not necessarily be better:

Theorem 2.4.1. *Let $\mathcal{G}$ be a set of functions/distributions parametrized by $\theta \in \vartheta$. Denote a typical member of this model by g_θ. Recall that $R[g_\theta]$ and $\hat{R}_m[g_\theta]$ denote the true and empirical risks with g_θ (computed using a loss function l). Then with probability $1-\delta$, we have:*

$$R[g_\theta] \leq \hat{R}_m[g_\theta] + 2\frac{\mathcal{C}(\mathcal{G},l)}{\sqrt{m}} + 3\sqrt{\frac{1}{2m}\log\frac{2}{\delta}} \ \forall\ \theta \in \vartheta. \tag{2.8}$$

where, $\mathcal{C}(\mathcal{G},l)$ denotes the complexity of the model and loss combination that satisfies the property that $\mathcal{C}(\mathcal{G}_1,l) \leq \mathcal{C}(\mathcal{G}_2,l)$ whenever $\mathcal{G}_1 \subset \mathcal{G}_2$.

Later on we will give an expression for this complexity for a particular model, loss combination.

At a first look this theorem's result may not seem great because after all $\hat{R}_m$ is simply the sample average, and R is the expectation, so it should be easy to derive

[18]Refer `http://www.cse.iitb.ac.in/saketh/teaching/cs729Notes.pdf` for details.

such a bound (for example, the proof of weak law of large numbers will itself provide a bound). Moreover, such a bound will not have anything to do with $\mathcal{G}$. If this is your thought process, then you should note that this is true if the g_θ if fixed apriori. However, note that if one needs to ascertain how good the ERM candidate, denoted by θ^{ERM}, is i.e., how close $\hat{R}_m[g_{\theta^{\text{ERM}}}]$ and $R[g_{\theta^{\text{ERM}}}]$ are, then such simple bounding techniques will not work simply because the θ^{ERM} itself is a function of the training data and hence the $\hat{R}_m$ is no more a simple average of iid random variables. The coolest thing about the theorem's result it that it holds uniformly for every $\theta \in \vartheta$ and hence can be used to compare $\hat{R}_m[g_{\theta^{\text{ERM}}}]$ and $R[g_{\theta^{\text{ERM}}}]$.

Also, in general, this bound cannot be improved. From this theorem it is very clear that big models are not necessarily better if we desire that the empirical risk of our candidate is close to its true risk. On the other hand, with smaller models, even though the empirical risk is perhaps close to the true risk, just because the model is perhaps too small to contain the unknown distribution, the predictive function that ERM/MLE/MAP/BAM chooses may be rendered useless. Hence an optimum balance needs to be striken where perhaps the bound in (2.8), which is usually called as the **guaranteed risk**, itself is minimized.

Based on the above discussion, one might hurry to write down the following algorithm for model selection and parameter selection, which is usually called as **structural risk minimization (SRM)**. For reasons of convenience, usually while talking about this algorithm, one assumes that the given models are nicely arranged in a structure[19] such as $m_1 \subset m_2 \subset m_3 \ldots$. SRM simply selects the model/hyperparameter and the parameter in it that minimizes the guaranteed risk. While this seems flawless at first sight, one has to note that the theorem holds when the model is fixed/given and perhaps not when the model is to be selected. One may go ahead and re-derive a theorem for this case where the structure/hierarchy of models is given (and not the model itself). You can believe me, this will simply lead to an additive term to (2.8) that is a function of complexity of this hierarchy/structure of models. Hence SRM is indeed a "safe" algorithm for model selection; however not so if one further wants to choose the hierarchy/structure itself. It should now be clear that one can keep on recursively keep minimizing additive variants of the bound in (2.8). In practice, users usually stop at the level of hyperparameter/model selection, but somehow ensure that they use some (finite) structure/hierarchy of models that will certainly contain or approximate the unknown distribution.

We will end this discussion with a slight variant of the simple MLE based model selection algorithm (the fourth one in the above list). It so happens that practitioners

[19]It so happens that this hierarchy/structure is a way to encode prior knowledge about which models or which parameters are expected to be "good" candidates. The idea is to simply put the most likely candidates in/as the smallest subset/model so on. etc. since SRM will prefer the smaller models this is like using prior knowledge that they are good.

more commonly employ different subsets of training data to perform parameter selection and hyperparameter selection using the same criteria of ERM/MLE/fit-to-the-data. Such algorithms as known as validation based algorithms. Here are some popular variants:

- The method of Validation for hyperparameter selection. Here, the idea is to randomly partition the training set into two parts, the validation set and the "new" training set. For each hyperparameter, using MLE/ERM etc., the parameter/candidate selection is performed using the "new" training set alone. Then each hyperparameter's performance is approximated by the performance on the validation set[20]. The hyperparameter/model that achieves highest validation set accuracy is chosen.
- The extreme case of the above called as leave-one-out cross-validation. Here, m iterations are performed: at i^{th} iteration, the i^{th} training example alone is used as the validation set and the rest are used for parameter selection and the validation accuracy on the left out example is computed for every hyperparameter. After all rounds, these validation accuracies are averaged (this average is called as the leave-one-out cross-validation accuracy). The hyperparameter/model that achieves the highest leave-one-out cross-validation accuracy is chosen.
- The k-fold cross-validation algorithm, which can be viewed as a hybrid of the above two: the initial dataset is randomly partitioned into k subsets, henceforth called as folds. Here, k iterations are performed: at i^{th} iteration, the i^{th} fold alone is used as the validation set and the rest are used for parameter selection and the validation accuracy on the left out fold is computed for every hyperparameter. After all rounds, these validation accuracies are averaged (this average is called as the k-fold cross-validation accuracy). The hyperparameter/model that achieves the highest k-fold cross-validation accuracy is chosen.

The important point to note however, owing to the discussion earlier, is that with any of these model selection algorithms, one pre-fixes the (finite) set of models/hyperparameters. And it is not necessarily better if more and more hyperparameters/models are considered. The only conscious choice that can be made for choosing the set of models/hyperparameters is to somehow ensure that the unknown distribution can be well approximated by this set.

[20]Again, this is intuitive from law of large numbers. However, one should still prove statistical consistency. Since in practice always one takes a finite set of hyperparameters, it turns out that statistically consistent is guaranteed.

Chapter 3

Examples of Various Models

This this chapter we work out the various algorithms presented earlier for various models. We begin with unsupervised learning examples.

3.1 Unsupervised Learning

We begin with the case where U_X is the unknown distribution, $\mathcal{D} = \{x_1, \ldots, x_m\}$, and the goal is to estimate the entire unknown distribution U_X.

3.1.1 Bernoulli and Beta-Bernoulli Models

Refer section 3.3 in Murphy [2012].

3.1.2 Multinoulli and Dirichlet-Multinoulli Models

Refer section 3.4 in Murphy [2012].

3.1.3 Gaussian and Gaussian-inverse-Gamma-Gaussian Models

Refer sections 4.1 and 4.6 in Murphy [2012].

3.1.4 Multivariate Gaussian and Gaussian-inverse-Wishart-Gaussian Models

Refer sections 4.1 and 4.6 in Murphy [2012].

3.1.5 Mixture Models

Refer sections 11.2.1-11.2.3, 11.3, 11.4.1-11.4.2.5, 11.4.7 in Murphy [2012] and Appendix 1 appearing near the end of this notes.

Here are some further comments:

- Please refer Wu [1983] for details on sufficiency conditions for convergence to stationary point by EM algorithm and proof details.
- Note that from the theorems in the above paper, it is clear that EM algorithm directly implements the generalized method of moments algorithm (2.4), where the goal is infact to find stationary points of log-likelihood. Hence, the EM algorithm is better motivated by the generalized method of moments algorithm rather than by the maximum likelihood algorithm. Somehow strangely, many books introduce EM algorithm for solving the maximum likelihood problem instead.
- For a novice there seems to be the following paradox: since it is known that (under some mild conditions) the generalized method of moments is statistically consistent, you may think that just by providing millions and millions of samples from U_X, through EM algorithm, you are able to recover the joint U_{XY} ! However a closer look will convince you that the mixture model itself never involves Y and it is we who are interpreting[1] the component densities are class conditionals and the mixing probabilities as the class prior. Hence the paradox is resolved.
- Moreover, two different mixing probabilities of component distributions may give exactly the same mixture distribution[2].
- However, the above never happens when the components are Gaussians with distinct means. A more comprehensive result is that Gaussian mixture model,

[1] We could arrive at the same distribution with a totally different re-parameterization.

[2] It is very easy to see this if the component distributions are discrete and hence equivalent to Euclidean vectors. In this case this essentially is the same as realizing that two different convex combinations of a set of vectors may give the same vector.

which is the set of all Gaussian mixture models with various number of components, is **universal** i.e., given any continuous distribution (over $\mathcal{X}$), there is a Gaussian mixture distribution (over $\mathcal{X}$) with finite number of components that closely approximates the given distribution. Please refer McLachlan and Peel [2000] for details. In other words, the Gaussian mixture model is rich enough for any learning problem (over $\mathcal{X}$).

- A Gaussian mixture model hence need not only be used for a clustering problem but also for example to model class conditionals in a classification problem (which is supervised learning).

3.1.6 Hidden Markov Models

Refer sections 17.3.1-17.5.2.3 in Murphy [2012] and Appendix 2 appearing at the end of this notes.

Here are some further comments:

- For a tutorial on speech recognition, which was the motivating application for HMMs in the lectures, please refer Rabiner [1989].
- We defined a HMM as a family of distributions over $\mathcal{X}, \mathcal{Y}$ ($calX$ is a space of (finite) sequences of vectors[3] and $\mathcal{Y}$ is a space of (finite) sequences of objects in a discrete set[4]) that satisfies the following conditional independence and homogeneity assumptions: $f_{X_1,\ldots,X_t,Y_1,\ldots,Y_t}(x_1,\ldots,x_t,y_1,\ldots,y_t) =$

$$
\begin{aligned}
& f_{X_1,\ldots,X_t/Y_1,\ldots,Y_t}(x_1,\ldots,x_t/y_1,\ldots,y_t) f_{Y_1,\ldots,Y_t}(y_1,\ldots,y_t) \\
& = f_{X_1/Y_1}(x_1/y_1)\ldots f_{X_t/Y_t}(x_t/y_t) f_{Y_1}(y_1) f_{Y_2/Y_1}(y_2/y_1)\ldots f_{Y_t/Y_{t-1}}(y_t/y_{t-1}) \\
& (\because \text{ the conditional independence assumptions. The first set say given the } Y_t, X_t \text{is conditionally independent of other } X_i) \\
& (\because \text{ while the second set is simply the Markovian assumption described in section 17.2.}) \\
& = f(x_1/y_1)\ldots f(x_t/y_t) f(y_1) f(y_2/y_1)\ldots f(y_t/y_{t-1}) \\
& (\because \text{ Homogenity assumption that says the (conditional) distribution does not depend on the position in the sequence.})
\end{aligned}
$$

- We also noted two graphical ways of representing HMMs:
  - One is by using Directed Graphical Models (DGMs). Here the conditional independences only can be represented (not the homogeneity assumptions). Please refer sections 10.1-10.2.2 in Murphy [2012]. The section

[3]In the speech recognition example, a member of $\mathcal{X}$ is a sequence of 12-dimensional MFCC cepstral coefficients computed for every frame in a speech signal.

[4]In the speech recognition example, a member of $\mathcal{Y}$ is a sequence of phonemes, one for every frame in a speech signal.

  10.2.2 provides the representation for HMMs. By this it should be clear how HMMs are to be generalized.

  – Another is by representing the Markov chain part in the HMM (i.e., the distribution over Y) through state transition diagrams (example figure 17.1 in Murphy [2012]).

- We next commented that there are other conditional independences that follow from those made above in a HMM. One of them was eqn. (17.50) in the book. We noted that this implied conditional independence is crucial for the EM algorithm.

- Though we motivated HMMs through the application of speech recognition, now we can use it model say, the class conditional densities in any classification problem involving inputs that are sequences. For e.g., speaker recognition where the training set is pairs of speech signals labeled with the speaker's name etc. (see also section 17.5.4 in Murphy [2012]).

- In lectures we also considered training HMMs in a supervised fashion. Though the maximum likelihood algorithm turns out to be simple, we commented that collecting such training data, where each time frame in a speech signal has to be labelled with the phoneme, is extremely tedious and hence usually we talk about HMMs trained in an unsupervised fashion.

- Infact, the most popular way of using HMMs is with loose supervision where for each speech signal the label is simply given as the word uttered. It turns out that one can tweak the unsupervised HMM a bit to utilize such weak supervision.

3.2 Supervised Learning

We begin with the case where U_{XY} is the unknown distribution, Y is discrete, $\mathcal{D} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, and the goal is to estimate the conditional distribution $U_{Y/X}$. We begin with probabilistic models and then finish with a section on deterministic models.

3.2.1 Probabilistic Models

3.2.1.1 Multivariate Bernoulli Naive Bayes Classifier

Refer sections 3.5.1-3.5.2 in Murphy [2012].

3.2.1.2 Gaussian Discriminant Analysis

Refer section 4.2 in Murphy [2012].

3.2.1.3 Logistic Regression

Refer sections 8.1-8.3.1 in Murphy [2012].

3.2.1.4 Linear Regression

Refer sections 7.1-7.3 and 7.5.1 in Murphy [2012].

3.2.2 Deterministic Models

The basic model in the deterministic case is the linear model (set of all linear functions over the input space= $\mathbb{R}^n$), denoted by $\mathcal{L} = \{f \mid \exists w \in \mathbb{R}^n \ \ni \ f(x) = w^\top x \, \forall \, x \in \mathbb{R}^n\}$. It is evident that $w \in \mathbb{R}^n$ is the parameter for this model. The following loss functions are popularly employed:

- Binary Classification:
  - Hinge-loss: $l(w^\top x, y) \equiv \max(0, 1 - yw^\top x)$. We commented that this loss is piece-wise linear, convex (wrt.w) and hence attractive. The zero-one loss on the other hand is non-convex and hence rarely employed[5].
  - Squared-hinge-loss: $l(w^\top x, y) \equiv \max(0, 1 - yw^\top x)^2$. This loss is further differentiable wrt. w, hence preferred over hinge-loss sometimes.
  - Logistic-loss: $l(w^\top x, y) \equiv \log(1 + \exp(-yw^\top x))$. This is also convex and differentiable (wrt. w).
- Regression:
  - Square-loss: $l(w^\top x, y) \equiv (y - w^\top x)^2$.
  - ϵ-insensitive loss: $l(w^\top x, y) \equiv \max(0, |y - w^\top x| - \epsilon)$.

With each of these one can implement ERM given by (2.1). It is easy to see that the logistic-loss ERM problem is exactly same as the MLE problem with logistic

[5]Refer Feldman et al. [2009] to know why the non-convexity in the zero-one loss makes ERM a computationally hard problem.

regression. Also, the square-loss ERM problem is exactly same as the MLE problem with linear regression.

While the above provide examples for equivalence with MLE, it so happens that there are examples where there is equivalence with MAP in logistic and linear regression. For this one usually employs the following subset of the linear model: $\mathcal{F}_W \equiv \{f \mid \exists w \in \mathbb{R}^n, \|w\| \leq W \ \ni \ f(x) = w^\top x \ \forall \ x \in \mathbb{R}^n\}$. Here W is the hyperparameter.

This model together with logistic-loss is equivalent to MAP based logistic regression. Here is why: the ERM problem in this case is

$$\min_{w \in \mathbb{R}^n} \quad \textstyle\sum_{i=1}^m \log(1 + \exp(-y_i w^\top x_i)) \tag{3.1}$$

$$\text{s.t.} \qquad \|w\| \leq W \tag{3.2}$$

From optimization theory (Lagrange duality), one can show that for every W there exists some $C > 0$ such that the above problem and the following have the same optimal solution set[6]:

$$\min_{w \in \mathbb{R}^n} \quad \frac{1}{2}\|w\|^2 + C \sum_{i=1}^m \log(1 + \exp(-y_i w^\top x_i)) \tag{3.3}$$

It is now easy to see that the above problem is exactly the same form of MAP based logistic regression with a zero-mean Gaussian prior over w.

Similarly, (after using Lagrange duality) it is easy to see that $\mathcal{F}_W$ together with square-loss is equivalent to MAP based linear regression with a zero-mean Gaussian prior over w (in other words, ridge regression).

The predictive function that results from $\mathcal{F}_W$ together with hinge-loss is called as the (soft-margin) Support Vector Machine (refer section 3.2.2.1). And that with $\mathcal{F}$ and ϵ-insensitive loss is called as Support Vector Regression (refer section 3.2.2.2).

3.2.2.1 Support Vector Machine

Refer section 14.5.2 in Murphy [2012].

3.2.2.2 Support Vector Regression

Refer section 14.5.1 in Murphy [2012].

Now one can similarly talk about various combinations of quadratic or higher-order non-linear models with the various losses. It so happens that there is a very

[6] C is now the hyperparameter that determines the model.

convenient way of generalizing these non-linear models, which is discussed in the subsequent section:

3.2.2.3 Kernel Machines

Consider the model $\mathcal{F}_{W,\phi} \equiv \{f \mid \exists w \in \mathcal{H}, \|w\| \leq W \ni f(x) = \langle w, \phi(x)\rangle \; \forall\, x \in \mathbb{R}^n\}$, where $\mathcal{H}$ is some Euclidean space or its infinite-dimensional generalization, called Hilbert space, $\langle\cdot,\cdot\rangle$ denotes the inner-product[7] in that space and $\phi : \mathbb{R}^n \mapsto \mathcal{H}$.

For example, let $\phi(x)$ be defined as the vector of all possible monomials with components of x upto degree two, let the inner-product be the dot-product and let $W \to \infty$. Then $\mathcal{F}_{W,\phi}$ in this case is nothing but the set of all quadratic functions (i.e., the quadratic model). The advantage with this representation however is that w is still the parameter. Needless to say, W, ϕ are the hyperparameters that determine the model.

Now we claim that with such a model and any loss function, the corresponding ERM problem will have an optimal solution of the form $w^* = \sum_{i=1}^m \alpha_i \phi(x_i)$. In other words, there will exist $\alpha_i \in \mathbb{R} \; \forall\; i = 1, \ldots, m$ such that $w^* = \sum_{i=1}^m \alpha_i \phi(x_i)$. The proof is simple:

- any w can be written as $\sum_{i=1}^m \alpha_i \phi(x_i)$ plus some vector in the orthogonal complement of the space spanned by $\phi(x_1), \ldots, \phi(x_m)$. Denote this orthogonal complement by $w_\perp$.

- Re-write the ERM problem in terms of αs and $w_\perp$:
  (3.4)
  $$\min_{\alpha\in\mathbb{R}^m, w_\perp\in\mathcal{H}} \;\; \frac{1}{2}\left\|\left(\sum_{i=1}^m \alpha_i\phi(x_i)\right) + w_\perp\right\|^2 + C\sum_{i=1}^m l\left(\left\langle\left(\sum_{j=1}^m \alpha_j\phi(x_j)\right) + w_\perp, \phi(x_i)\right\rangle, y_i\right),$$

  which is same as ($\because \langle w_\perp, \phi(x_i)\rangle = 0$ for any $i = 1, \ldots, m$ by the very definition of orthogonal complement):
  (3.5)
  $$\min_{\alpha\in\mathbb{R}^m, w_\perp\in\mathcal{H}} \;\; \frac{1}{2}\left\|\left(\sum_{i=1}^m \alpha_i\phi(x_i)\right) + w_\perp\right\|^2 + C\sum_{i=1}^m l\left(\left\langle\sum_{j=1}^m \alpha_j\phi(x_j), \phi(x_i)\right\rangle, y_i\right),$$

[7] For the purposes of this course it is enough to understand that Hilbert space and inner-product are simply the generalizations of the Euclidean space and dot-product. Interested students may refer `http://www.cse.iitb.ac.in/saketh/teaching/cs723Scans3.pdf` for further details.

which is same as ($\because$ Pythogorus theorem holds in $\mathcal{H}$):

$$(3.6) \qquad \min_{\alpha\in\mathbb{R}^m} \quad \frac{1}{2}\left\|\sum_{i=1}^m \alpha_i\phi(x_i)\right\|^2 + C\sum_{i=1}^m l\left(\left\langle\sum_{j=1}^m \alpha_j\phi(x_j), \phi(x_i)\right\rangle, y_i\right),$$

which is same as ($\because$ using properties of inner product):

$$(3.7) \qquad \min_{\alpha\in\mathbb{R}^m} \quad \frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m \alpha_i\alpha_j\langle\phi(x_i),\phi(x_j)\rangle + C\sum_{i=1}^m l\left(\sum_{j=1}^m \alpha_j\left\langle\phi(x_j),\phi(x_i)\right\rangle, y_i\right),$$

The above result is known as the representer theorem. In addition to the result, from the proof it is clear that ERM, which is originally an optimization problem in $\mathcal{H}$ (that potentially could be infinite dimensional), is essentially a problem in $\mathbb{R}^m$. Secondly, from the proof it is clear that neither solving ERM (training phase) nor for computing the prediction function $\langle w, \phi(x)\rangle$ (inference phase) the hyperparameter ϕ is not explicitly needed. What is needed is only inner-product between $\phi(x_i), \phi(x)$.

It so happens that mathematicians (in 1950s) have well-understood functions that may represent inner products in some Hilbert space. Here is the result:

Suppose $k : \mathcal{X}\times\mathcal{X} \mapsto \mathbb{R}$ is a function such that $G = \begin{bmatrix} k(x_1,x_1) & \dots & k(x_1,x_m) \\ k(x_2,x_1) & \dots & k(x_2,x_m) \\ \vdots & \vdots & \vdots \\ k(x_m,x_1) & \dots & k(x_m,x_m) \end{bmatrix}$ is a symmetric positive semi-definite matrix for any $\{x_1, x_2, \dots, x_m\} \subset \mathcal{X}$ and for any $m \in \mathbb{N}$, then there exists a Hilbert space $\mathcal{H}_k$ and a map $\phi_k : \mathcal{X} \mapsto \mathcal{H}_k$ such that $k(x_1,x_2) = \langle\phi(x_1),\phi(x_2)\rangle \ \forall\ x_1, x_2 \in \mathcal{X}$. Such functions k are called as **kernels**.

Some examples of kernels are **linear kernel** $k(x_1,x_2) \equiv x_1^\top x_2$, **polynomial kernel** $k(x_1,x_2) \equiv \left(1 + x_1^\top x_2\right)^d$ (here d is the hyperparameter that controls the degree of the polynomial), **Gaussian kernel** or Radial Basis Function (RBF) kernel $k(x_1,x_2) \equiv \exp\left(-\frac{\|x_1-x_2\|^2}{2\sigma^2}\right)$ (here $\sigma > 0$ is the hyperparameter). Please verify if the definition is satisfied for all these examples. Your book Murphy [2012], in section 14.2, provides examples of kernels on non-Euclidean spaces too!

This discussion makes it clear that, instead of using the model $\mathcal{F}_{W,\phi_k}$, we may now use the equivalent $\mathcal{F}_{W,k}$ and the ERM problem is of size m. The later model is more convenient because of the following reasons:

- In many applications, it is easier for domain experts to say how similar two inputs are rather than to provide a Euclidean representation. For e.g., if the inputs are graphs, it is easy to say how similar two graphs are rather than to give a Euclidean vector representation for a graph. Such similarity functions can be directly used as kernels (provide the mathematical conditions are satisfied); whereas the former model will explicitly require ϕ i.e., the representation of the input.

- If the dimensionality of $\mathcal{H}$ is high (or say ∞), then $\mathcal{F}_{W,k}$ is an efficient representation. Moreover, the ERM in the form (3.7) is of manageable size.

- Useful for building non-linear models: for example, with the Gaussian kernel, $\langle w, \phi_k(x)\rangle = \sum_{i=1}^{m} \alpha_i \exp\left(-\frac{\|x_i - x\|^2}{2\sigma^2}\right)$, which is a non-linear function of x.

- One can show that the (infinite) hierarchy $\mathcal{F}_{10^{-10},k} \subset \mathcal{F}_{10^{-9},k} \subset \ldots \subset \mathcal{F}_{1,k} \subset \mathcal{F}_{10,k} \subset \ldots$, where k is the Gaussian kernel, is universal in the sense that the union of all these can approximate any (measurable) function over $\mathbb{R}^n$. For practitioners, this simply means that they can start with Gaussian kernel and SVM, tune hyperparameters using some model selection, then the classifier will perform well (given enough training data). This is important because no such guarantee, even with lots of training examples, can be given for say SVM with linear or polynomial kernel or for most of the models discussed in the course.

Bibliography

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

V. Feldman, V. Guruswami, P. Raghavendra, and Yi Wu. Agnostic learning of monomials by halfspaces is hard. *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 385–394, 2009.

Lars Peter Hansen. Large sample properties of generalized method of moments estimators. *Econometrica*, 50:1029–1054, 1982.

Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004. ISSN 1532-4435.

G. McLachlan and D. Peel. *Finite Mixture Models.* John Wiley and Sons, New York, 2000.

Kevin Murphy. *Machine Learning: A Probabilistic Perspective.* MIT Press, 1 edition, 2012.

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

S. M. Ross. *A First Course in Probability.* Pearson Education, 6 edition, 2002.

S. M. Ross. *Introduction to Probability Models.* Academic Press, 9 edition, 2006.

Bernhard Schölkopf and Alex Smola. *Learning with Kernels.* MIT press, Cambridge, 2002.

Sheldon Axler. *Linear Algebra Done Right.* Springer-Verlag, 1997.

V. N. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, Inc., 1998.

C. F. Jeff Wu. On the Convergence Properties of the EM Algorithm. *Annals of Statistics*, 11(1):95–103, 1983.

Appendix 1

(EM algorithms for mixture models)

We have $U_{XY} \longrightarrow \mathcal{D} = \{x_1, x_2, \ldots, x_m\}$

$$\log\left(p_\theta(\mathcal{D})\right) = \sum_{i=1}^{m} \log\left(p_\theta(x_i)\right)$$

$$= \sum_{i=1}^{m} \log\left(\sum_{j=1}^{\boxed{k}} \underbrace{p_\theta(x_i/j)}_{\substack{\text{component distributions} \\ \text{let parameters be } \theta_j}} \underbrace{p_\theta(j)}_{\substack{\text{mixing probabilities} \\ \text{let parameters be } \pi}} \right)$$

$k \rightarrow$ no. of comp (given)

model parameters $\theta \equiv (\theta_1, \ldots, \theta_k, \pi)$

Since, in general, the above log-likelihood need not be concave in θ_j & π, we resort to an iterative algorithm where a concave lower bounding function that is exact at the iterate is chosen and maximized to update the iterate. This is illustrated in figure 11.16 in Murphy's book and is a standard idea for maximizing non-concave functions.

Since the expression involves a logarithm of a convex combination, using the definition of concave functions a lower bound is immediate. However, it seems unless the coefficients of the convex combination are known, the resulting lower bound need not be concave. Hence we thought of artificially bringing in the coefficients through: $q_i(j) \ni q_i(j) \geq 0 \ \forall i, j$ &

$$\sum_{j=1}^{k} q_i(j) = 1 \quad \forall i$$

$$\log p_\theta(\mathcal{D}) = \sum_{i=1}^{m} \log\left(\sum_{j=1}^{k} \left[\frac{p_\theta(x_i/j)\, p_\theta(j)}{q_i(j)}\right] q_i(j)\right)$$

$$\geq \sum_{i=1}^{m} \sum_{j=1}^{k} q_i(j) \log\left(\frac{p_\theta(x_i/j)\, p_\theta(j)}{q_i(j)}\right)$$

$\left(\because \log \text{ is a concave function } \& \ q_i \text{ is a valid pmf}\right)$

As mentioned earlier, we wish to choose $q_i(j)$ such that the inequality becomes an equality at $\theta = \theta^t$, the current iterate.

Towards this end:

$$\log p_\theta(\mathcal{D}) \geq \sum_{i=1}^{m} \sum_{j=1}^{k} q_i(j) \log\left(\frac{p_\theta(j/x_i)\, p_\theta(x_i)}{q_i(j)}\right)$$

(∵ Bayes rule)

$$= \sum_{i=1}^{m} \underbrace{\boxed{\sum_{j=1}^{m} q_i(j) \log\left(\frac{p_\theta(j/x_i)}{q_i(j)}\right)}}_{\leq 0 \ \forall i} + \underbrace{\boxed{\sum_{i=1}^{m} \log p_\theta(x_i)}}_{\log p_\theta(\mathcal{D})}$$

From the above it is clear that our choice for

$$\boxed{q_i(j) \text{ should be } p_{\theta^t}(j/x_i)}.$$

So with this choice, the idea is to maximize the lower bound:

$$\theta^{t+1} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{m} \underbrace{\boxed{\sum_{j=1}^{k} p_{\theta^t}(j/x_i) \log\left(p_\theta(x_i/j)\, p_\theta(j)\right)}}_{E_{J\sim q_i}\left[\log\left(p_\theta(x_i/J)\, p_\theta(J)\right)\right]}$$

EQN I

i.e. maximization of expectation

The objective function in optimization prob. in EQN $\textcircled{I}$ is called as "auxiliary function" (denoted by Q) in Kevin's book.

With this notation in mind, please follow section 11.4.2 for details of solving EQN $\textcircled{I}$ in the special case of GMM.

Appendix 2

The only change to EQN $\textcircled{I}$ in case of HMMs, is that the number of possible y_i for an x_i is s^{T_i}, where s is the no. states and T_i is the sequence length of x_i.

Hence 'k' in EQN $\textcircled{I}$ is to be replaced by $k_i = s^{T_i}$

$$\boxed{\Theta^{t+1} = \underset{\boxed{\theta = (\pi, A, B)}}{\operatorname{argmax}} \sum_{i=1}^{m} \sum_{j=1}^{\boxed{k_i}} p_{\theta^t}(j/x_i) \log\left(p_\theta(x_i/j)\, p_\theta(j)\right)}$$

HMM's parameters

EQN $\textcircled{II}$

The details of solving EQN $\textcircled{II}$ for the special case where emission distributions are Gaussian is given in section 17.5.2 in the Kevin's book.

Note that (17.96) simply gives the expression for the objective in EQN $\textcircled{II}$.