Learning theory

September 28, 2020

Abstract

Topics: empirical error, generalization error, empirical risk minimization (ERM), Uniform convergence, Union bound.
Readings: Chapter 2-7 of UML.

1 Motivation and notation

We have learned several supervised learning algorithms, such as linear regression, logistic regression, naive Bayes, and SVM. In practice, different learning models generate different performance during testing, and even with the same machine learning model, different training methods and training data can lead to varied test performance. To understand what makes the difference, we need a formal framework to analyze the factors that contribute to a model's performance.

Example 1.1. *Figure 1 shows an example where three trained linear regression models can perform well or poorly, depends on what data patterns the models find.*

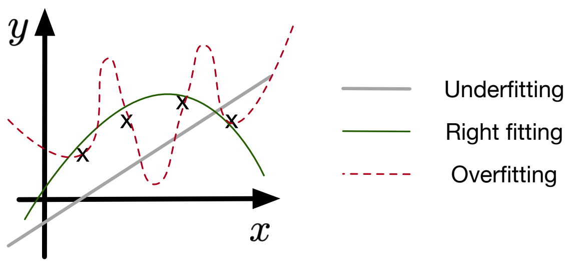


Figure 1: The four training points (black crosses) are generated from a quadratic function that needs to be learned. Depending on what hypothesis spaces we choose to fit a linear regression model, the learned model can underfit (when linear functions $h_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x$ are used) or overfit (e.g., when cubic functions $h_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$ are used) the training data so that the underlying true quadratic function is not learned. Only when the right hypothesis space (the class of quadratic functions $h_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$) is used will the training algorithm (e.g., gradient descent minimizing MSE) find the underlying true quadratic function.

Example 1.2. *The following example comes from the UML book. In Figure 2, assuming that the data points $\mathbf{x} = (x_1, x_2)$ are uniformly sampled from interior of the outer rectangle, and a data point is labeled as 1 if and only if the point lies within the inner rectangle. The outer rectangle has area 2 and the inner one has area 1. Given any m training examples $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$, the classifier*

$$h(\mathbf{x}) = \begin{cases} y^{(i)} & \text{if } \mathbf{x} = \mathbf{x}^{(i)} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

has zero training error but has 1/2 error over random samples of unseen test data points.

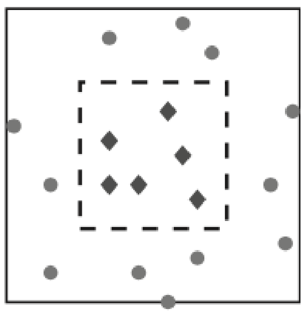


Figure 2: In $\mathbb{R}^2$, the solid dots represent negative examples and diamonds represent positive examples. The underlying labeling function f is represented by the inner rectangle.

1.1 Notation and Definitions

Let $S = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$ be a set of m I.I.D. training samples from some data generation distribution $\mathcal{D}$, which is unknown. Let f be a labeling function that maps $\mathbf{x}$ to $\{0, 1\}$ so that f defines the ground truth labels of the data. For example, $\mathbf{x}^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ and $y^{(i)} = f(\mathbf{x}^{(i)}) = \mathbb{1}[\boldsymbol{\theta}^\top \mathbf{x}^{(i)} > 0]$ for some fixed but unknown parameters $\boldsymbol{\mu}$, Σ, and $\boldsymbol{\theta}$. We let the distribution $\mathcal{D}^m$ be the distribution from which any m I.I.D. data points S are sampled (denoted by $S \sim \mathcal{D}^m$).

A learning algorithm assumes a fixed hypothesis space $\mathcal{H} = \{h : y = h(\mathbf{x})\}$, which contains functions that map from $\mathbf{x}$ to $y \in \{0, 1\}$. The function f may or may not be in $\mathcal{H}$. $|\mathcal{H}|$ is the number of functions in $\mathcal{H}$ and can be finite or infinite.

Example 1.3. *$\mathcal{H}$ can contain $h = \mathbb{1}[h(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x} > 0]$ and there are infinitely many such h's identified by $\boldsymbol{\theta}$. $\mathcal{H} = \{h : h(\mathbf{x}) = \mathbb{1}[x_1 > \theta], \theta \in \{-1, 0, 1\}\}$ has only three hypotheses.*

The learning algorithm needs to select a hypothesis that achieves good prediction performance on the test data $(\mathbf{x}, y)$ that is sampled in the same way as the training data. The performance is measured by the generalization error (test error, or risk):

$$L_{\mathcal{D},f}(h) = \Pr_{\mathbf{x}\sim\mathcal{D}}(h(\mathbf{x}) \neq f(\mathbf{x})) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbb{1}[h(\mathbf{x}) \neq f(\mathbf{x})]]. \tag{2}$$

That is, the generalization error of h is the probability that h will make a mistake on any data sampled from $\mathcal{D}$ and labeled by f. We drop the subscript $\mathcal{D}$ and f from $L_{\mathcal{D},f}$ if there is no ambiguity. The generalization error of a hypothesis is unknown since the distribution $\mathcal{D}$ and labeling function f are unknown. The error can be estimated using the empirical error (training error), a proxy to the generalization error based on S:

$$L_S(h) = \frac{1}{m} \sum_{i=1}^m \mathbb{1}[h(\mathbf{x}^{(i)}) \neq y^{(i)}], \tag{3}$$

where the subscript S emphasizes that the training error depends on S.

2 ERM and PAC learning

Ideally, we want to find a hypothesis $h^* \in \mathcal{H}$ such that

$$h^* = \arg\min_{h\in\mathcal{H}} L_{\mathcal{D},f}(h), \tag{4}$$

for any data distribution $\mathcal{D}$ and labeling function f so that the generalization error is minimized. This is the ultimate goal of supervised machine learning algorithms. However, $\mathcal{D}$ and f are not accessible during model training. and only m training examples S are accessible. The best a learning algorithm can do is to minimize the empirical error $L_S(h)$. Empirical risk minimization (ERM) refers to any algorithms that select a hypothesis from $\mathcal{H}$ to minimize the empirical error:

$$\hat{h}_S = \arg\min_{h\in\mathcal{H}} L_S(h). \tag{5}$$

We may drop the subscript S in $\hat{h}_S$ that represents the dependency of h on S when it is clear from the context. If the true error of $\hat{h}_S$ is much worse than the empirical error of $\hat{h}_S$ on S, then we say the hypothesis $\hat{h}_S$ overfits or underfits the training data S.

$\mathcal{H}$ restricts or biases the selection of predictors to be within a subset of all functions mapping from $\mathbf{x}$ to y, and the bias is called "inductive bias". By choosing an inductive bias for a model, one may overfit or underfit the training data, or find the underlying f. A fundamental question in learning theory is over which hypothesis class $\mathcal{H}$ that ERM will not overfit or underfit the training data (i.e., the empirical error and the true error are close).

Since the training data S is a random sample from the underlying distribution $\mathcal{D}^m$, the resulting $\hat{h}_S$ and $L_S(\hat{h}_S)$ depend on S and are thus random. We model $L_S(\hat{h})$ as a random variable taking values in the domain $[0,1]$ (the empirical error is indeed in this range). Probably Approximately Correct (PAC) learning is theoretical framework that analyzes the gap between $L(\hat{h})$ and $L_S(\hat{h})$ with respect to random sampling of the training data S. We define two quantities.

- The accuracy parameter ϵ measures the upper bound of the gap between the empirical error and generalization error of any hypothesis $h \in \mathcal{H}$:

$$|L(h) - L_S(h)| \leq \epsilon. \tag{6}$$

  This parameter corresponds to the term "approximately correct".

- The confidence parameter δ: since the "bad" event $|L(h) - L_S(h)| > \epsilon$ happens with some probability due to random sampling of S, δ upper-bounds the probability that such an event happens:

$$\Pr(|L(h) - L_S(h)| > \epsilon) \leq \delta. \tag{7}$$

  This parameter corresponds to the term "probably".

A picture relating the two parameters are shown in Figure 3.

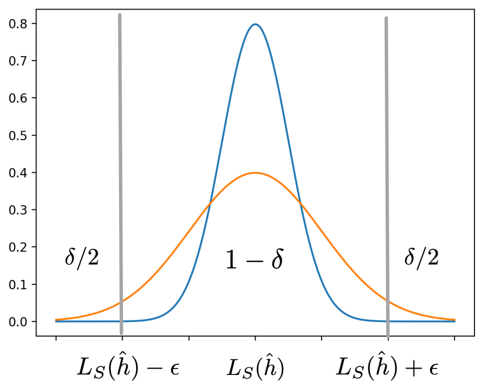


Figure 3: On the x-axis, we indicate the empirical error $L_S(\hat{h})$. Due to the randomness in S, $L_S(\hat{h})$ is a random variable that is the sum of m terms. By large number theory, $L_S(\hat{j})$ should distributed like a Gaussian. The two vertical bars indicate lower and upper bounds of ($L_S(\hat{h}) - \epsilon$ and $L_S(\hat{h}) + \epsilon$) of a region that the generalization error $L(\hat{h})$ should fall in with high confidence $(1-\delta)$.

PAC learning aims to find an integer m for a hypothesis class $\mathcal{H}$, so that for any ϵ and δ, and for any training set S I.I.D. sampled from $\mathcal{D}^m$, one has

$$\mathcal{D}^m\{L(\hat{h}_S) \leq L(h^*) + 2\epsilon\} \geq 1 - \delta. \tag{8}$$

That is, with probability at least $1-\delta$, the "desirable" event $L(\hat{h}_S) \leq L(h^*) + 2\epsilon$ (the generalization error of the hypothesis h_S selected by ERM on any training set S of size m not exceeding the best generalization error $L(h)$ by ϵ) will happen [1]. You may find this statement hard to interpret. So imagine that there is a random mechanism that sends out the m training examples, leading to an random event that may or may not not happen. Then you ask: over many rounds of such

[1]The integer m is the minimal in the sense that any other $m' > m$ will work too.

simulation, what's the ratio of trials that such event does happen, and to make this ratio greater than $1-\delta$, how large m should be. The intuition is that with a larger m, ERM will select h_S more accurately so that the probability of the desirable event will increase.

Given an finite-size hypothesis class $\mathcal{H}$, accuracy δ, and confidence δ, how can one compute such an m, even without the knowledge of data distribution $\mathcal{D}$ (since the PAC learning statement holds true for any $\mathcal{D}$)?

2.1 Mathematical tools for analyzing generalization errors

Now we introduce three mathematical tools that will formally define PAC learning.

2.1.1 Union bound

Given any two events (or sets) A and B from a sample space Ω (the universal set) and a probability distribution Pr defined on Ω, we have

$$\Pr(A \cup B) \leq \Pr(A) + \Pr(B). \tag{9}$$

This inequality can be generalized to more than two or even countably infinitely many events $A_1, A_2, \ldots$

$$\Pr(\cup_{k=1}^{\infty} A_k) \leq \sum_{k=1}^{\infty} \Pr(A_k). \tag{10}$$

2.1.2 Hoeffding's inequality

We state the theorem without proof.

Theorem 2.1. *Let $v_1, \ldots, v_m$ be m I.I.D. random variables, all bounded in $[0,1]$ and with mean $\mu = \mathbb{E}[v_i]$, $i = 1, \ldots, m$. Then for any $\epsilon > 0$,*

$$\Pr\left(\left| \frac{1}{m} \sum_{i=1}^{m} v_i - \mu \right| > \epsilon \right) \leq 2\exp\{-2m\epsilon^2\}. \tag{11}$$

It says that the empirical mean of m I.I.D. samples with the same mean will converge to the mean with high probability that increases exponentially in m (the right hand side converges to 0). This type of convergence is called "convergence in probaility" due to randomness in v_i and is different from the convergence $\frac{1}{m}\sum_{i=1}^{m} v_i \to \mu$.

2.1.3 Uniform convergence

Uniform convergence refers to a "nice" property of a hypothesis space $\mathcal{H}$: there is an integer $m(\epsilon, \delta)$ for any accuracy parameter ϵ, confidence δ, and data distribution $\mathcal{D}$, whenever a training set $S \sim \mathcal{D}^m$ has size $m \geq m(\epsilon, \delta)$, it is guaranteed that for *any* $h \in \mathcal{H}$, $|L(h) - L_S(h)| \leq \epsilon$ with probability $1 - \delta$ over random sampling of S.

Note: The term "uniform" refers to the fact that the integer $m(\epsilon, \delta)$ works for "any" hypothesis $h \in \mathcal{H}$ and does not depend on which h is being discussed.

In the definition of uniform convergence, any such S is said to be ϵ-representative with respect to the hypothesis space $\mathcal{H}$ and the distribution $\mathcal{D}$: if one evaluates any $h \in \mathcal{H}$ on S regarding the performance of h ($L_S(h)$), one can be sure that the true error of h ($L(h)$) will only be ϵ away from $L_S(h)$.

Lemma 2.2. *Let S be a ϵ-representative training set with respect to the hypothesis space $\mathcal{H}$ and the distribution $\mathcal{D}$ and $\hat{h} = \arg\min_{h \in \mathcal{H}} L_S(h)$, then for any $h \in \mathcal{H}$,*

$$L(\hat{h}) \leq L(h) + 2\epsilon. \tag{12}$$

In particular,

$$L(\hat{h}) \leq L(h^*) + 2\epsilon = \min_{h \in \mathcal{H}} L(h) + 2\epsilon. \tag{13}$$

Proof.

$$
\begin{aligned}
L(\hat{h}) &\le L_S(\hat{h}) + \epsilon && (14)\\
&\le L_S(h) + \epsilon && (15)\\
&\le L(h) + \epsilon + \epsilon && (16)\\
&= L(h) + 2\epsilon. && (17)
\end{aligned}
$$

The first and the third inequality is based on the assumption that S is ϵ-representative (and thus the concentration of $L_S(h)$ around $L(h)$), while the second inequality is because of $\hat{h} = \arg\min_{h\in\mathcal{H}} L_S(h)$ (any h can't outperform $\hat{h}$ on the training set S).

Since h^* is just one specific $h \in \mathcal{H}$, so the inequality $L(\hat{h}) \le \min_{h\in\mathcal{H}} L(h) + 2\epsilon$ holds. □

2.1.4 PAC learning for finite $\mathcal{H}$

Given these tools, we are ready to introduce PAC learning formally. We want the event $L(\hat{h}) \le L(h^*) + 2\epsilon$ to happen with probability at least 1-δ with respect to the sampling distribution $S \sim \mathcal{D}^m$. Based on Lemma 2.2, the event will happen with respect to a particular $S \sim \mathcal{D}^m$ if S is ϵ-representative (is the other direction also true?). Therefore, we just need to make sure that with probability at least 1-δ, S is sampled to be ϵ-representative. Recall the definition of ϵ-representativeness, we need

$$\mathcal{D}^m(\{S : \forall h \in \mathcal{H} \text{ such that } |L_S(h) - L(h)| \le \epsilon\}) \ge 1 - \delta, \tag{18}$$

or equivalently,

$$\mathcal{D}^m(\{S : \exists h \in \mathcal{H} \text{ such that } |L_S(h) - L(h)| > \epsilon\}) \le \delta. \tag{19}$$

Here we are using the axiom of probability that $\Pr(A) = 1 - \Pr(\neq A)$, where $\neg A$ is the complementary of an event A, and the negation of the statement "for any $h \in \mathcal{H}$ such that some property of h holds" is "there is a certain $h \in \mathcal{H}$ such that the property of h does not hold".

$$
\begin{aligned}
& \mathcal{D}^m(\{S : \exists h \in \mathcal{H} \text{ such that } |L_S(h) - L(h)| > \epsilon\}) && (20)\\
= \;& \mathcal{D}^m(\cup_{h\in\mathcal{H}}\{S : |L_S(h) - L(h)| > \epsilon\}) && (21)\\
\le \;& \sum_{h\in\mathcal{H}} \mathcal{D}^m(\{S : |L_S(h) - L(h)| > \epsilon\}). && (22)
\end{aligned}
$$

The inequality is based on union bound. The equality may be hard to recognize if you don't have real analysis or measure theory background, but it is intuitive to understand based on Figure 4 that reproduces Figure 2.1 from the UML book. In the figure, one point represents a training set $S \sim \mathcal{D}^m$ and a circle represents a collection of "bad" training sets, each of which leads to $|L_S(h) - L(h)| > \epsilon$ for a certain $h \in \mathcal{H}$. The left hand side of the above equality computes the probability of all points falling inside the four circles (so long as there exists one h that $|L_S(h) - L(h)| > \epsilon$), while the right hand side first identifies the four circles ($\{S : |L_S(h) - L(h)| > \epsilon\}$), each associated with a "bad" $h \in \mathcal{H}$ and then computes the probability of the union of the circles.

Lastly, we can use the Hoeffding's inequality to bound the $\mathcal{D}^m(\{S : |L_S(h) - L(h)| > \epsilon\})$. Due to I.I.D of the training data and linearity of expectation,

$$
\begin{aligned}
\mathbb{E}[L_S(h)] &= \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} \mathbb{1}[h(\mathbf{x}^{(i)}) \neq y^{(i)}]\right] && (23)\\
&= \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}[\mathbb{1}[h(\mathbf{x}^{(i)}) \neq y^{(i)}] && (24)\\
&= \frac{1}{m}\sum_{i=1}^{m} \Pr(h(\mathbf{x}) \neq f(\mathbf{x})) && (25)\\
&= L(h). && (26)
\end{aligned}
$$

$L(h)$ is the mean of each of the I.I.D. random variable $\mathbb{1}[h(\mathbf{x}^{(i)}) \neq y^{(i)}] \in [0, 1]$. By Hoeffding, we have

$$\mathcal{D}^m(\{S : |L_S(h) - L(h)| > \epsilon\}) \le 2\exp\left\{-2m\epsilon^2\right\}. \tag{27}$$

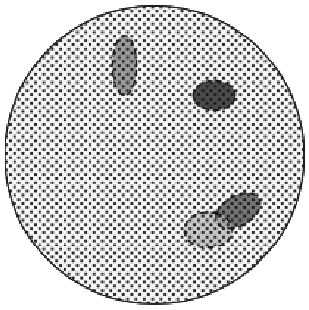


Figure 2.1. Each point in the large circle represents a possible m-tuple of instances. Each colored oval represents the set of "misleading" m-tuple of instances for some "bad" predictor $h \in \mathcal{H}_B$. The ERM can potentially overfit whenever it gets a misleading training set S. That is, for some $h \in \mathcal{H}_B$ we have $L_S(h) = 0$. Equation (2.9) guarantees that for each individual bad hypothesis, $h \in \mathcal{H}_B$, at most $(1-\epsilon)^m$-fraction of the training sets would be misleading. In particular, the larger m is, the smaller each of these colored ovals becomes. The union bound formalizes the fact that the area representing the training sets that are misleading with respect to some $h \in \mathcal{H}_B$ (that is, the training sets in M) is at most the sum of the areas of the colored ovals. Therefore, it is bounded by $|\mathcal{H}_B|$ times the maximum size of a colored oval. Any sample S outside the colored ovals cannot cause the ERM rule to overfit.

Figure 4: This figure is a reproduction of Figure 2.1 of the UML book.

$$\mathcal{D}^m(\{S : \exists h \in \mathcal{H} \text{ such that } |L_S(h) - L(h)| > \epsilon\}) \leq 2|\mathcal{H}| \exp\left\{-2m\epsilon^2\right\}. \tag{28}$$

Letting

$$\delta = 2|\mathcal{H}| \exp\left\{-2m\epsilon^2\right\}, \tag{29}$$

The factor 2 before $|\mathcal{H}|$ is interesting: if the distribution of $L_S(h)$ is symmetric, then each side of the mean of $L_S(h)$ will have $|\mathcal{H}| \exp\left\{-2m\epsilon^2\right\}$. Solving for m, we obtain the so-called "sample complexity":

$$m = \frac{\ln(2|\mathcal{H}|/\delta)}{2\epsilon^2}, \tag{30}$$

so that when there are at least m (rounded up) training examples, the ERM can produce a hypothesis that has generalization error no ϵ more than the minimal error with probability $1 - \delta$.

2.1.5 Error bound

If we are given δ and m, solving for ϵ, we will have for any $h \in \mathcal{H}$,

$$|L_S(h) - L(h)| \leq \epsilon = \sqrt{\frac{\ln(2|\mathcal{H}|/\delta)}{2m}}. \tag{31}$$

If we plug this ϵ in Eq. (32), we obtain the follow

$$L(\hat{h}) \leq L(h^*) + 2\sqrt{\frac{\ln(2|\mathcal{H}|/\delta)}{2m}}. \tag{32}$$

This bound of the generalization error is commonly seen in machine learning research papers.

To interpret this inequality, we can regard $L(h^*)$ as a bias term and $2\sqrt{\frac{\ln(2|\mathcal{H}|/\delta)}{2m}}$ as the variance term, leading to a bias-variance decomposition for binary classification. The bias represents how well the best hypothsis from the selected hypothesis class $\mathcal{H}$ can fit the training data S, while the variance is an upper-bound of the difference $L(\hat{h}) - L(h^*)$ and represents how hard for the ERM solution $\hat{h}$ to approach the optimal hypothesis h^* that has the lowest generalization error.

For example, the class of linear classifiers $\mathcal{H} = \{h : h(x) = \theta_0 + \theta_1 x\}$ is a subset of the class $\mathcal{H}' = \{h : h(x) = \theta_0 + \theta_1 x + \theta_2 x^2\}$, since one can take $\theta_2 = 0$ to obtain classifiers in $\mathcal{H}$. As a result, $L(h^*) \leq L(h'^*)$ for the generalization error minimizers $h^* \in \mathcal{H}$ and $h'^* \in \mathcal{H}$. We then say that the class $\mathcal{H}'$ has a smaller bias. The price it pays is an increased variance, as $2\sqrt{\frac{\ln(2|\mathcal{H}|/\delta)}{2m}}$ has the term $|\mathcal{H}|$ and $|\mathcal{H}| < |\mathcal{H}'|$. It is typically unknown before hand which hypothesis class one should take. Figure 5 demonstrates this trade-off between bias and variance.

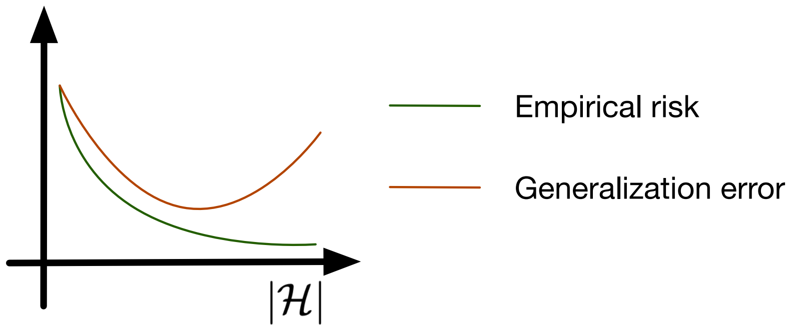


Figure 5: Given a training set $S \sim \mathcal{D}^m$, when $|\mathcal{H}|$ is small, $\mathcal{H}$ is not expressive enough to fit the data. As one increases the size of the hypothesis class (such as fitting adding more features in the linear classifiers), the empirical error and generalization error both go down, due to more reduction in the bias than increase in the variance. Lastly, a large $\mathcal{H}$ can fit S very well and leads to almost zero empirical error, but the generalization error starts to go up since the variance becomes more prominent than the reduction in bias (overfitting).

3 The no-free-lunch theorem

3.1 Motivation

From Figure 4, we can see that the smaller circles are the collections of "misleading" size-m training samples that lead to the picking of bad hypotheses that have a large (ϵ) generalization error. A nice practice to have is to pick the hypothsis class $\mathcal{H}$ carefully to avoid the small circles before running a learning algorithm (such as ERM). The selection of such an $\mathcal{H}$ represents some kind of prior knowledge of the underlying data distribution $\mathcal{D}$ that is unknown to the learner.

For example, although we don't know how spams are generated, when building a spam email classifier, commonsense tells us that if keywords such as "free", "cash" appear in the email contents, then the likelihood of spam becomes higher. The algorithm designers can then incorporate such prior knowledge in ruling out hypotheses that are insensitive to such keywords.

So one question is, is there a universal learning algorithm with a pre-determined hypothesis space $\mathcal{H}$ and can learn well on *any* data distribution $\mathcal{D}$. The point here is that $\mathcal{H}$ is picked before $\mathcal{D}$ so that any prior knowledge about $\mathcal{D}$ cannot guide the selection of $\mathcal{H}$. It turns out that this is impossible.

3.2 The no-free-lunch theorem

Theorem 3.1. *Let A be any learning algorithm for the task of binary classification with respect to the 0-1 loss over a domain $\mathcal{X}$. Let m be any number smaller than $|\mathcal{X}|/2$, representing a training set size. Then there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ such that:*

- *There exists a function $f : \mathcal{X} \to \{0,1\}$ with 0 generalization error: $L_{\mathcal{D}}(f) = 0$.*
- *With probability of at least 1/7 over the choice of $S \sim \mathcal{D}^m$ we have that $L_{\mathcal{D}}(A(S)) \geq 1/8$.*

For any learner, there is a learning task, identified by a specific distribution and training set size, so that the returned hypothesis has high error rate ($> 1/8$) with a high probability ($> 1/7$). On the other hand, if the same learner can take f as a prior knowledge and uses a hypothesis class $\mathcal{H}$ that contains f, then ERM will be able to identify f with zero generalization error.

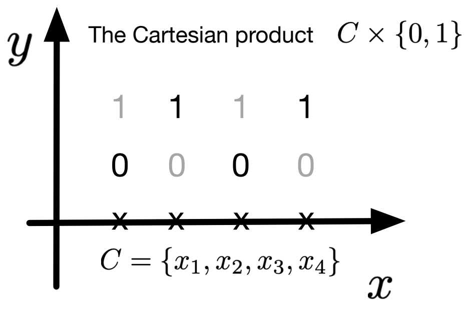
Proof. The proof[2] is constructive: we will construct a learning task so that any learner can fail.

[2]The proof is based on the proof of Theorem 5.1 in UML.

Let C be a subset of $\mathcal{X}$ of size $2m$. There are $T = 2^{2m}$ possible ways to label the training instances in C with 0 or 1. Each way of labeling is a function mapping from C to $\{0,1\}$ and we let these T functions be denoted as $f_1, \ldots, f_T$.

For each function f_i, we induce a discrete data distribution $\mathcal{D}_i$ over the Cartesian product $C \times \{0,1\}$:

$$\mathcal{D}_i(\mathbf{x}, y) = \begin{cases} 1/|C| & \text{if } y = f_i(\mathbf{x}), \\ 0 & \text{otherwise.} \end{cases}$$



y
The Cartesian product $C \times \{0,1\}$
1 1 1 1
0 0 0 0
$C = \{x_1, x_2, x_3, x_4\}$
x

Figure 6: This figure shows an example with $m = 2$ and $C = \{x_1, x_2, x_3, x_4\}$. An example labeling function $f_i : C \to \{0,1\}$ is indicated by the dark-colored 0's and 1's on top of each training instance. With this f_i, the discrete distribution $\mathcal{D}_i$ will be: $\mathcal{D}_i(x_1, 0) = \mathcal{D}_i(x_2, 1) = \mathcal{D}_i(x_3, 0) = \mathcal{D}_i(x_4, 1) = 1/4$. Note that there are other $2^4 - 1 = 15$ labeling functions and data distributions.

The generalization error of f_i on the distribution $\mathcal{D}_i$ is zero: $L_{\mathcal{D}_i}(f_i) = 0$. We will show that any learning algorithm A, which takes m training examples S in the space $C \times \{0,1\}$ and return a hypothsis (or function) $A(S) : C \to \{0,1\}$, it holds that:

$$\max_{1 \le i \le T} \mathbb{E}_{S \sim \mathcal{D}_i^m}[L_{\mathcal{D}_i}(A(S))] \ge 1/4. \tag{33}$$

The max operator is essentially saying that there is at least one i such that the corresponding labeling function f_i and the induced distribution $\mathcal{D}_i$ lead to a large average generalization error of the hypothesis $A(S)$ returned by the learning algorithm A on training set S sampled from $\mathcal{D}_i$.

The function f_i can be extended to a function f defined over the entire set $\mathcal{X}$ by assigning arbitrary values to $\mathbf{x} \notin C$. The distribution $\mathcal{D}_i$ can be extended to $\mathcal{X}$ by assigning 0 probability to $\mathcal{X} \setminus C$. As a result, it holds true that

$$\mathbb{E}_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}(A(S))] \ge 1/4, \tag{34}$$

for any learning algorithm A.

To prove Eq. (33), let's list all possible training sets $S_1, \ldots, S_k$ that can be generated from C, where $k = 2^{2m}$. For the set $S_j = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$, when the instances are labeled by f_i, we denoted the labeled set by $S_j^i = ((\mathbf{x}_1, f(\mathbf{x}_1)), \ldots, (\mathbf{x}_m, f(\mathbf{x}_m)))$. If the distribution over C is $\mathcal{D}_i$, then there are k labeled training sets $S_1^i, \ldots, S_k^i$ and all of them have the same probability of being sampled (since after f_i labels all $2m$ points in C, all sets of size m have equal probability of being sampled).

$$\mathbb{E}_{S \sim \mathcal{D}_i}[L_{\mathcal{D}_i}(A(S))] = \frac{1}{k} \sum_{j=1}^{k} L_{\mathcal{D}_i}(A(S_j^i)). \tag{35}$$

$$\begin{aligned} \max_{1 \le i \le T} \mathbb{E}_{S \sim \mathcal{D}_i}[L_{\mathcal{D}_i}(A(S))] &\ge \frac{1}{T} \sum_{i=1}^{T} \left[\frac{1}{k} \sum_{j=1}^{k} L_{\mathcal{D}_i}(A(S_j^i)) \right] \\ &= \frac{1}{k} \sum_{j=1}^{k} \left[\frac{1}{T} \sum_{i=1}^{T} L_{\mathcal{D}_i}(A(S_j^i)) \right] \\ &\ge \min_{1 \le j \le k} \frac{1}{T} \sum_{i=1}^{T} L_{\mathcal{D}_i}(A(S_j^i)) \end{aligned}$$

Fix any j and let $S_j = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ be a specific training set (without being labeled). Let $\mathbf{z}_1, \ldots, \mathbf{x}_p$ be the data points that are in C but not in S_j. Since S_j may have duplicated points from C, $p \geq m$. For any function $h : C \to \{0, 1\}$ and any $i \in \{1, \ldots, k\}$ we have

$$\begin{aligned} L_{\mathcal{D}_i}(h) &= \frac{1}{2m} \sum_{\mathbf{x} \in C} \mathbb{1}[h(\mathbf{x}) \neq f_i(\mathbf{x})] \\ &\geq \frac{1}{2m} \sum_{r=1}^{p} \mathbb{1}[h(\mathbf{z}_r) \neq f_i(\mathbf{z}_r)] \\ &\geq \frac{1}{2p} \sum_{r=1}^{p} \mathbb{1}[h(\mathbf{z}_r) \neq f_i(\mathbf{z}_r)] \end{aligned}$$

Therefore

$$\begin{aligned} \frac{1}{T} \sum_{i=1}^{T} L_{\mathcal{D}_i}(A(S_j^i)) &\geq \frac{1}{T} \sum_{i=1}^{T} \left[\frac{1}{2p} \sum_{r=1}^{p} \mathbb{1}[A(S_j^i)(\mathbf{z}_r) \neq f_i(\mathbf{z}_r) \right] \\ &= \frac{1}{2p} \sum_{r=1}^{p} \left[\frac{1}{T} \sum_{i=1}^{T} \mathbb{1}[A(S_j^i)(\mathbf{z}_r) \neq f_i(\mathbf{z}_r) \right] \\ &\geq \frac{1}{2} \min_{1 \leq r \leq p} \left[\frac{1}{T} \sum_{i=1}^{T} \mathbb{1}[A(S_j^i)(\mathbf{z}_r) \neq f_i(\mathbf{z}_r) \right] \end{aligned}$$

The last term within the brackets represents, over all possible labeling function f_i, the chance that the hypothesis learned from S_j^i (S_j labeled by f_i) will make a mistake on an unseen point $\mathbf{z}_r$ labeled by the same function f_i. So what's an upper bound of this chance? Fix any $\mathbf{z}_r$, let f_i and f_i' be two of the T labeling functions that label $\mathbf{z}_r$ with different labels. Since $\mathbf{z}_r$ is not in the training set S_j, the hypothesis returned by the learning algorithm A should be the same so that the labels $A(S_j^i)$ labels all points outside C the same as well. One and only one of f_i and f_i' will label $\mathbf{z}_r$ differently than $A(S_j^i)$. There are $T/2$ such pairs and so

$$\frac{1}{T} \sum_{i=1}^{T} \mathbb{1}[A(S_j^i)(\mathbf{z}_r) \neq f_i(\mathbf{z}_r) = \frac{1}{T} \sum_{i=1}^{T/2} \left\{ \mathbb{1}[A(S_j^i)(\mathbf{z}_r) \neq f_i(\mathbf{z}_r)] + \mathbb{1}[A(S_j^i)(\mathbf{z}_r) \neq f_i'(\mathbf{z}_r)] \right\} = 1/4.$$

To complete the proof, Exercise 5.1 of UML will prove the second conclusion of the theorem. □

4 The VC-dimension

The VC-dimension is a complexity measure of a hypothesis class $\mathcal{H}$ that more fundamentally characterizes the difficulty in learning from data with $\mathcal{H}$, compared to the number of hypothesis in $\mathcal{H}$. The following example shows that even $\mathcal{H}$ has infinitely many hypotheses, $\mathcal{H}$ is learnable (that is, there is a learning algorithm so that for any ϵ and δ, and for any distribution $\mathcal{D}$, there is an integer m, that algorithm returns a hypothesis with generalization error upper bounded by ϵ with probability at least $1 - \delta$ over random sampling of size m training sets). See Lemma 6.1 of UML for an example where $\mathcal{H}$ is of infinite size but is learnable.

Definition 4.1. *A hypothesis class $\mathcal{H}$ shatters a finite set C if all $2^{|C|}$ possible ways of labeling elements of C to 0 and 1 can be implemented by any functions in $\mathcal{H}$.*

Since there are $2^{|C|}$ possible ways of labeling, there should be at least $2^{|C|}$ hypotheses in $\mathcal{H}$.

Example 4.2. *The set $C = \{x_1\} \subset \mathbb{R}$ can be shattered by the class of all threshold functions $h_a(x) = \mathbb{1}[x < a]$, since by choosing $a_1 < x_1$ and $a_2 > x_1$, x_1 is labeled in both of the two ways: $h_{a_1}(x_1) = 0$ and $h_{a_2}(x_1) = 1$. Now let $C = \{x_1, x_2\} \subset \mathbb{R}$ and $x_1 < x_2$. The labeling of x_1 to 0 and x_2 to 1 can't be implemented by any threshold function defined above.*

Definition 4.3. *The VC-dimension of a hypothesis class $\mathcal{H}$ is the maximal size of a set $C \subset \mathcal{X}$ that can be shattered by $\mathcal{H}$. If $\mathcal{H}$ can shatter sets of arbitrary size, then the VC-dimension of $\mathcal{H}$ is infinite.*

To show that $\mathcal{H}$ has VC-dimension d, we need to prove

- There is a set C of size d can be shattered by $\mathcal{H}$, and
- any set C of size $d+1$ can not be shattered by $\mathcal{H}$.

Example 4.4. *When $\mathcal{H}$ is the set of threshold functions defined above, then its VC-dimension is 1.*

Example 4.5. *Let $\mathcal{H}$ be the set of indicator function of intervals: $h_{[a,b]}(x) = \mathbb{1}[x \in [a,b]]$. For the set $C = \{x_1, x_2\}$, any of the four ways of labeling x_1 and x_2 can be implemented by hypotheses in $\mathcal{H}$. For any set $C = \{x_1, x_2, x_3\}$, there is a labeling ($x_1 \to 1$, $x_2 \to 0$, and $x_3 \to 1$) can be implemented by any function in $\mathcal{H}$.*

Example 4.6. *Axis-aligned rectangles in $\mathbb{R}^2$ has VC-dimension 4.*

While VC-dimension measures the complexity of a set of functions (hypotheses), the number of parameters of the functions may not be directly related to the VC-dimension. One example is that $h_\theta(x) = \lceil 0.5 \sin(\theta x) \rceil$ has infinite VC-dimension (for any integer d, there is a set of d points on the real line that can be shattered by $\mathcal{H}$.

Theorem 4.7. *The fundamental theorem of PAC learning: Let $\mathcal{H}$ be a hypothesis class of functions from $\mathcal{X}$ to $\{0,1\}$. With 0-1 loss function, $\mathcal{H}$ has the uniform convergence property $\iff$ $\mathcal{H}$ has a finite VC-dimension.*

With uniform convergence, one can prove PAC-learnability of $\mathcal{H}$ similarly to the finite $|\mathcal{H}|$ case.

5 Model selection

There are many machine learning models and each can have hyperparameters that can take different values. For example, one can train an SVM or a logistic regression model on a given training set, and for SVM, one can choose different kernel functions, or different values for bandwith of Gaussian kernel. The collection of models, along with their hyperparameter values, compose the hypothesis space $\mathcal{H}$. For example, one can set $\mathcal{H}$ to be all SVM models with the Gaussian kernel with bandwidth $\sigma \in \{1, 10, 20\}$.

During training time, without the test data to evaluate the generalization errors, it is hard to tell which model and hyperparameter values are best[3].

Model selection refers to the techniques that use available training data to guide selection of learning models and hyperparameters. There are several ways.

5.1 Hold-out validation

A portion, say 50%, of the training data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$ can be held out for validation after a hypothesis from $\mathcal{H}$ is learned by a learning algorithm (such as ERM) on the remaining portion of the training data. Since the hold-out portion is independent of the selected hypothesis, the generalization error estimated on the hold-out portion converges to the true generalization error in probability, according to Hoeffding's inequality.

5.2 Cross validation

k fold cross validation partitions the training data into k folds with equal size. To evaluate a learning algorithm, each of the k folds are used as validation set while the remaining $k-1$ folds are used to run the learning algorithm and select a hypothesis The generalization errors on the k folds are then averaged to obtain an estimation of the true generalization error.

[3]Generalization error is what machine learning cares about and can only be computed on test data unseen during training

6 Regularization

Recall that when fitting a polynomial regression model $h(x; \mathbf{w}) = \mathbf{w}^\top \mathbf{x} = \sum_{j=0}^n w_j x^j$, one needs to choose a hypothesis space $\mathcal{H}$ that depends on the degree of the polynomials n. Based on the No-Free-Lunch Theorem, it is desirable to select a hypothesis space (selecting an n, or basis functions $1, x, x^2, \ldots, x^n$ for data representation) using prior knowledge to avoid ERM selecting a hypothesis h with low empirical error and high generalization error. Such as example is given in Figure 1. The example shows that with a larger n, the hypothesis space $\mathcal{H}$ has more polynomials to fit the data and overfitting can be a problem. One needs to restrict the size or complexity of the hypothesis space $\mathcal{H}$ by selecting a smaller n. Selecting n is done by cross-validation.

6.1 Regularized linear regression

The prior knowledge about the degree n can be hard to specify, and typically the features of data are given and n is fixed. Rather, it can be easier to work with an n and let "regularization" to control the complexity of the hypothesis space $\mathcal{H} = \{\mathbf{w} \in \mathbb{R}^n : h(x; \mathbf{w}) = \sum_{j=0}^n w_j x_j\}$, where x_j is the j-th feature and we ignore the bias term so there is no constant 1 appended to $\mathbf{x}$. We focus on $\ell 2$ regularized linear regression, that adds the regularization term

$$R(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 = \frac{\lambda}{2}\sum_{j=1}^n w_j^2. \tag{36}$$

to the MSE loss function

$$J(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^m \frac{1}{2}\left(\mathbf{w}^\top \mathbf{x}^{(i)} - y^{(i)}\right)^2. \tag{37}$$

The regularized linear regression problem becomes

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^n} J(\mathbf{w}) + R(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{m}\sum_{i=1}^m \frac{1}{2}\left(\mathbf{w}^\top \mathbf{x}^{(i)} - y^{(i)}\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2. \tag{38}$$

The MSE loss function and the regularization term are both convex functions in $\mathbf{w}$ so that a local optimum is a global optimum. By taking the partial derivative of the objective function $J(\mathbf{w}) + R(\mathbf{w})$ with respect to $\mathbf{w}$, we find

$$\frac{\partial}{\partial \mathbf{w}}(J(\mathbf{w}) + R(\mathbf{w})) = \frac{1}{m}(X^\top X\mathbf{w} - X^\top \mathbf{y}) + \lambda \mathbf{w}, \tag{39}$$

where $\mathbf{X} = \begin{bmatrix} —x^{(1)\top}— \\ —x^{(2)\top}— \\ \vdots \\ —x^{(m)\top}— \end{bmatrix}$ is the design matrix and $\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \cdots \\ y^{(m)} \end{bmatrix}$ is the vector of m target values. Setting the partial derivative to zero, we obtain

$$\mathbf{w}^* = (X^\top X + m\lambda I)^{-1} X^\top \mathbf{y}, \tag{40}$$

where I is the $n \times n$ identity matrix. Since the matrix $X^\top X$ is positive semi-definite, the matrix $X^\top X + m\lambda I$ is positive definite with eigenvalues greater than 0 and the matrix inversion exists.

6.2 Bayesian linear regression

The above regularization can be motivated from a Bayesian treatment of linear regression. Using a generative model, we can generate the target values $y^{(i)}$ in the training data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$, given the feature vectors $\mathbf{w}^{(i)}$ as follows:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \tau^2 I),$$
$$y^{(i)} \sim \mathcal{N}(\mathbf{w}^\top \mathbf{x}^{(i)}, \sigma^2)$$

Note that the two Gaussian distributions are of different dimensions: the first one is of the dimension of $\mathbf{w}$ (n-dim) and the second one is of one dimension since $y^{(i)}$ is a scalar. The hyper-parameters τ^2 and σ^2 are the variances of the corresponding random variables $\mathbf{w}$ and $y^{(i)}$.

The distribution of $\mathbf{w}$ models the prior knowledge about $\mathbf{w}$: each element of $\mathbf{w}$, w_j, should have mean 0 and variance τ^2. The way we specify a prior of $\mathbf{w}$ is not by taking a hypothesis space for $\mathbf{w}$, but by specifying a probability distribution of $\mathbf{w}$. By taking a different mean or covariance matrix for $\mathbf{w}$, one can incorporate different prior knowledge of $\mathbf{w}$. For example, if one knows that some element w_j should not be zero, then the corresponding mean for w_j should be set to the known mean, and if w_j and w_k are known to have strong positive interaction, then the covariance Σ_{jk} should be set to a positive value rather than zero as in $\tau^2 I$. The distribution of $y^{(i)}$ is the likelihood of $y^{(i)}$ conditioned on $\mathbf{w}$ and $\mathbf{x}^{(i)}$, indicating that $y^{(i)}$ should have mean equal to the predicted value $\mathbf{w}^\top \mathbf{x}$ with variance σ^2.

According to the Bayes rule, the posterior distribution of $\mathbf{w}$ is

$$\Pr(\mathbf{w}|\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m; \tau, \sigma) = \frac{\Pr(\mathbf{w}; \tau) \prod_{i=1}^m \Pr(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}; \sigma)}{\Pr(\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m)}. \tag{41}$$

Here we are assuming that the training data are I.I.D. sampled from some underlying distribution $\mathcal{D}$ so the product in the numerator. Note that the notation of the vertical bar and the semi-colon in the probability distributions mean different things: the vertical bar means conditioning, while anything after a semi-colon means those are given hyper-parameters.

We can see that the Bayesian linear regression will absorb the prior information about $\mathbf{w}$ and the training data into the posterior distribution of $\mathbf{w}$, rather than estimating a single point $\mathbf{w}^*$ as in MLE.

The denominator is the probability of see the training target values using the above generative model and is just the integration of the numerator over all possible $\mathbf{w}$:

$$\Pr(\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m) = \int_{\mathbf{w}} \Pr(\mathbf{w}; \tau) \prod_{i=1}^m \Pr(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}; \sigma) d\mathbf{w}. \tag{42}$$

The integration can be difficult to evaluate since $\mathbf{w}$ is in an n dimensional space. Fortunately, for linear regression with $\Pr(\mathbf{w}; \tau) = \mathcal{N}(\mathbf{0}, \tau^2 I)$, the posterior is also a Gaussian distribution and can be evaluated in a closed-form. Seem PRML Section 3.3 for more details.

During testing, given a test example $\mathbf{x}$, the Bayesian linear regression uses the predictive distribution:

$$\Pr(y|\mathbf{w}, \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m; \tau, \sigma) = \int_{\mathbf{w}} \Pr(\mathbf{w}|\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m; \tau, \sigma) \Pr(y|\mathbf{x}, \mathbf{w}; \sigma). \tag{43}$$

The posterior of $\mathbf{w}$ is used to calculate the likelihood of every single $\mathbf{w}$ value ($\Pr(\mathbf{w}|\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m; \tau, \sigma)$), multiplying that with the likelihood of a specific y value for that given $\mathbf{w}$ ($\Pr(y|\mathbf{x}, \mathbf{w}; \sigma)$). The integration takes into account of all possible $\mathbf{w}$ values and the uncertainty $\mathbf{w}$. Since both factors in the integrand are Gaussian distributions, the left-hand-side is also a Gaussian distribution. See PRML Section 3.3 for more details.

Since the predictive distribution does not give a specific y value as a prediction, the expected value of y under the predictive distribution is used to predict y:

$$\mathbb{E}[y] = \int_t t \times \Pr(t|\mathbf{w}, \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m; \tau, \sigma) \mathrm{d}t. \tag{44}$$

In this integration, one considers all possible values of t and the probability of taking each of those values. But this is just the mean of the distribution of y with density function $\Pr(y|\mathbf{w}, \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m; \tau, \sigma)$, which is Gaussian.

Instead of the above fully Bayesian treatment, we can take a simpler approach that can still incorporate prior distribution of $\mathbf{w}$ in the linear regression, leading to the $\ell 2$ regularized linear regression. In particular, we can maximize the posterior of $\mathbf{w}$ Eq. (41):

$$\mathbf{w}^*_{\text{MAP}} = \arg\max_{\mathbf{w}} \Pr(\mathbf{w}|\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m; \tau, \sigma) = \Pr(\mathbf{w}; \tau) \prod_{i=1}^m \Pr(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}; \sigma). \tag{45}$$

This is called the maximum a posterior (MAP) estimation. Compared to the MLE estimation of $\mathbf{w}$, one only maximize the likelihood

$$\mathbf{w}^*_{\text{MLE}} = \arg\max_{\mathbf{w}} \prod_{i=1}^m \Pr(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}; \sigma). \tag{46}$$

You can prove that $\mathbf{w}^*_{\text{MAP}}$ leads to the same problem in Eq. (38).

7 Feature selection

Given the training data with n features, there can be features x_j that are not relevant to the target variable y. One can select only the relevant ones to build a model and the selection is called "feature selection". Feature selection can be considered as a special case of model selection, trying to identify the best hypothesis space that involves the selected feature only.

Formally, given the training data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$ and n features, we aim to find a subset of k features that gives the best generalization error. This is a subset selection problem and is thus NP-hard. Heuristics have been developed to approximately find a good set of k features.

7.1 Forward and backward selection

The first kind of feature selection algorithms are called "wrapper" selection: a selection algorithm wraps around a learning algorithm as a component. Therefore, the selected features are dependent on the given learning algorithm.

Algorithm 1: Forward feature selection

```
Given: training data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$, $\mathbf{x}^{(i)} \in \mathbb{R}^n$, the number of features to be selected $k$,
  the learning algorithm to be used;
Initialization: $\mathcal{F} = \emptyset$.;
for $t = 1$ to $k$ do
    for $j \in \{1, \ldots, n\} \setminus \mathcal{F}$ do
        Train or cross-validate a model using the learning algorithm on the st of feature(s)
          $\mathcal{F} \cup \{j\}$;
        Record the training error or the cross-validation error of feature $j$;
    end
    Let $j_t$ be the feature that has the lowest training error or cross-validation error;
    $\mathcal{F} = \mathcal{F} \cup \{j_t\}$.
end
return $\mathcal{F}$
```

Backward selection works in a similar way but starts from the full set of features $\{1, \ldots, n\}$ and eliminates one feature at a time until there is only k features left.

The advantage of wrapper selection is that the selected subset of features fit the given learning algorithm so that the performance of the model trained on $\mathcal{F}$ can be optimized (locally). One of the disadvantages is that the selection procedure requires training many models and is costly. The other disadvantage is that the selected features may be suboptimal for a different learning algorithm.

7.2 Filter selection

To address the two disadvantages of wrapper selection, one can use the filter feature selection algorithms. First, each feature is evaluated by some metric that is generally useful for many learning algorithms, such as feature-target correlation. The top k features with the best performance are selected to train any models. We discuss two metrics that are useful for filter feature selection.

7.2.1 Mutual information

An information theoretical metric for feature evaluation is mutual information. Given a feature X_j and the target variable Y (both assumed to be discrete random variables). Then we can calculate

$$\mathrm{MI}(X_j, Y) = \sum_{x_j} \sum_{y} \Pr(x_j, y) \log \frac{\Pr(x_j, y)}{\Pr(x_j)\Pr(y)}, \tag{47}$$

where the two summations are over all the possible values that X_j and Y can take, and the probabilities $\Pr(x_j, y)$, $\Pr(x_j)$, and $\Pr(y)$ are estimated from the training data. For example, the joint probability $\Pr(x_j, y)$ is the ratio of the number of training examples having values $X_j = x_j$ and $Y = y$ to the number of all training examples (m).

If the two random variables X_j and Y are independent, then $\Pr(x_j, y) = \Pr(x_j)\Pr(y)$ and the mutual information will be 0. Mutual information is the KL divergence applied to the two distributions $P(X_j, Y) = \Pr(x_j, y)$ and $Q(X_j, Y) = \Pr(x_j)\Pr(y)$:

$$\mathrm{KL}(P||Q) = \sum_{x_j} \sum_{y} P(x_j, y) \log \frac{P(x_j, y)}{Q(x_j, y)}. \tag{48}$$

It can be proved that the KL divergence is non-negative, so that 0 is the lowest value that the mutual information can take.

7.2.2 Correlation

If a feature x_j is highly correlated with the target y, then x_j is predictive of y (note: it does not mean x_j will be predictive of y along with other features). Formally, assume the features and target are all continuous. Define the Pearson correlation coefficient as

$$r(X_j, Y) = \frac{\langle X_j - \bar{X}_j, \mathbf{y} - \bar{y} \rangle}{\|X_j - \bar{X}_j\| \times \|\mathbf{y} - \bar{y}\|}. \tag{49}$$

That is, we subtract the mean of the j-th column of the design matrix X from each element of the j-th column, and the mean of the target from each element of $\mathbf{y}$. This step is called "centering". Then we take the inner product of the centered X_j and $\mathbf{y}$ and divide the result by the product of the $\ell 2$ norms of the two centered vectors.

7.3 Feature selection via regularization

As we aim to find k out of n features, we can add a constraint on the upper bound of the number of non-zeros in the parameter vector $\mathbf{w}$ to the objective function of linear regression:

$$\begin{aligned} \min\nolimits_{\mathbf{w}} \quad & J(\mathbf{w}) = \tfrac{1}{m} \textstyle\sum_{i=1}^{m} \tfrac{1}{2} \left(\mathbf{w}^\top \mathbf{x}^{(i)} - y^{(i)}\right)^2, \\ \text{s.t.} \quad & \|\mathbf{w}\|_0 \le k, \end{aligned} \tag{50}$$

where the $\ell 0$ norm of $\mathbf{w}$ is defined as the number of non-zeros in $\mathbf{w}$. However, $\ell 0$ norm is not convex and non-differentiable and the optimization problem is quite difficult.

The $\ell 1$ regularized linear regression is a relaxed version of the $\ell 0$ regularized linear regression so that the constraint becomes convex:

$$\begin{aligned} \min\nolimits_{\mathbf{w}} \quad & J(\mathbf{w}) = \tfrac{1}{m} \textstyle\sum_{i=1}^{m} \tfrac{1}{2} \left(\mathbf{w}^\top \mathbf{x}^{(i)} - y^{(i)}\right)^2, \\ \text{s.t.} \quad & \|\mathbf{w}\|_1 \le k, \end{aligned} \tag{51}$$

Using the Lagrangian multiplier, it can be proved that there is a corresponding unconstrained optimization problem to the above $\ell 1$ constrained optimization:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left(\mathbf{w}^\top \mathbf{x}^{(i)} - y^{(i)}\right)^2 + \lambda \|\mathbf{w}\|_1. \tag{52}$$