
Theory of Deep Learning

Manik Bhandari

Department of Computational Data Science
Indian Institute of Science
Bangalore, India
mbbhandarimanik2@gmail.com

Abstract

Notes of Theory of Deep Learning mainly from lectures at IISC.

1 Introduction

Define error of a classifier (aka *true error*) as probability of it making a mistake given a random data point.

$$L_{D,f}(h) = \mathbb{P}_{x \sim D}[h(x) \neq f(x)] = D(\{x : h(x) \neq f(x)\})$$

where D is the distribution from where a data point x is drawn, f is a known *correct* function which always gives the correct labels to a data point. By this definition $D(A)$ is the probability of observing a random point x from A .

Define training error or *empirical risk* as

$$L_S(h) = \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m}$$

where S is the training *set* (it is actually a sequence since points can repeat and classifiers often take into account their order) of the form $\{(x_i, y_i)\}$. If you naively minimize this empirical risk then you are likely to overfit. To avoid it, you use some prior knowledge about the *kind of classifier* that can possibly fit to the data and restrict your hypothesis search space to those types of classifiers.

This kind of restriction induces a *bias* in the model (aka *inductive bias*). In this setting, define

$$h_S = \text{ERM}_h(S) \in \underset{h \in \mathcal{H}}{\text{argmin}} L_S(h).$$

This is a tradeoff – choosing a restricted \mathcal{H} can add too much bias but choosing a large \mathcal{H} may lead to overfitting.

Finite hypothesis class If we restrict \mathcal{H} to have an upper bound on its size then ERM_h will not overfit if we have *large* training data (how large will depend on size of \mathcal{H}).

Realizability Assumption There exists $h^* \in \mathcal{H}$ such that $L_{D,f}(h^*) = 0$ i.e. it never makes a mistake which means that $L_S(h^*) = 0$. Since this is the least possible error, this means that for every ERM hypothesis $L_S(h_S) = 0$. We are however interested in true error of h_S i.e. $L_{D,f}(h_S)$.

iid assumption Assume that elements of S are identically and independently distributed according to D denoted by $S \sim D^m$.

Now, we would like to have an h_S such that $L_{D,f}(h_S)$ is not too large. Let's say h_S *fails* if $L_{D,f}(h_S) > \epsilon$.

We want to upper bound the probability of sampling a training set that leads to a failure i.e. $D^m(S : L_{D,f}(h_S) > \epsilon)$. Define bad hypothesis as $\mathcal{H}_B = \{h \in \mathcal{H} : L_{D,f}(h_S) > \epsilon\}$ and misleading training sets as $M = \{S : \exists h \in \mathcal{H}_B, L_S(h) = 0\}$. So, all the training sets for which h_S fails must be misleading (there can be other misleading sets also). So

$$\{S : L_{D,f}(h_S) > \epsilon\} \subseteq M = \bigcup_{h \in \mathcal{H}_B} \{S : L_S(h) = 0\}.$$

This means that

$$D^m(\{S : L_{D,f}(h_S) > \epsilon\}) \leq D^m(M) = D^m\left(\bigcup_{h \in \mathcal{H}_B} \{S : L_S(h) = 0\}\right).$$

Take union bound of RHS to get

$$D^m(\{S : L_{D,f}(h_S) > \epsilon\}) \leq \sum_{h \in \mathcal{H}_B} D^m(\{S : L_S(h) = 0\}) = \sum_{h \in \mathcal{H}_B} \left(\prod_{i=1}^m D(\{x_i : h(x_i) = f(x_i)\}) \right).$$

and since $h \in \mathcal{H}_B$, $D(\{x_i : h(x_i) = f(x_i)\}) \leq 1 - \epsilon$. But each x_i is iid over D so $D^m(\{S : L_S(h) = 0\}) \leq (1 - \epsilon)^m \leq e^{-\epsilon m}$. As m goes large, the probability of finding a misleading set reduces. Therefore

$$D^m(\{S : L_{D,f}(h_S) > \epsilon\}) \leq |\mathcal{H}_B| e^{-\epsilon m} \leq |\mathcal{H}| e^{-\epsilon m}.$$

Take log both sides to get

$$\log D^m(\{S : L_{D,f}(h_S) > \epsilon\}) \leq \log |\mathcal{H}| - \epsilon m \implies m \leq \frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$$

where $\delta = D^m(\{S : L_{D,f}(h_S) > \epsilon\})$. This also implies that if m is large enough i.e. $m \geq \frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$ then $L_{D,f}(h_S) \leq \epsilon$ with probability $1 - \delta$ of choosing the iid samples S .

So with the ERM_h rule, your hypothesis will be *probably* $(1 - \delta)$ *approximately* (ϵ) *correct* (PAC). Note that the size m does not depend upon the underlying distribution or labeling function.

PAC Learnability A hypothesis class \mathcal{H} is PAC learnable if $\exists m_{\mathcal{H}} : (0, 1)^2 \rightarrow \mathbb{N}$ and a learning algorithm such that

For every $(\epsilon, \delta) \in (0, 1)$, for every distribution \mathcal{D} over \mathcal{X} and for every labeling function $f : \mathcal{X} \rightarrow (0, 1)$

If the realizability assumption holds over $\mathcal{H}, \mathcal{D}, f$

then running the algorithm on $m > m_{\mathcal{H}}(\epsilon, \delta)$ samples generated iid from \mathcal{D} and labeled by f gives a hypothesis h such that

with probability at least $1 - \delta$ over the choice of examples, $L_{\mathcal{D},f}(h) \leq \epsilon$.

Sample complexity $m_{\mathcal{H}} : (0, 1)^2 \rightarrow \mathbb{N}$ defines the *sample complexity* of learning \mathcal{H} i.e. how many samples are required to get a PAC solution. Let it be the *minimum function* that satisfies the criteria of PAC learnability.

Sample complexity of finite hypothesis class Every finite hypothesis class is PAC learnable with sample complexity $m \leq \lceil \frac{\log(|\mathcal{H}|/\delta)}{\epsilon} \rceil$

Removing realizability assumption Assuming that such an h^* exists such that

$\mathbb{P}_{x \sim \mathcal{D}}[h^*(x) = f(x)] = 1$ is too strong. Not only might such an h^* not exist, your features might not be discriminative enough. Instead assume that \mathcal{D} is a joint distribution over domain points \mathcal{X} and labels \mathcal{Y} . Now, true error

$$L_D(h) = \mathbb{P}_{(x,y) \sim \mathcal{D}}[h(x) \neq y] = \mathcal{D}(\{(x, y) : h(x) \neq y\})$$

Bayes optimal predictor is the best labeling function defined as

$$f_{\mathcal{D}}(x) = \begin{cases} 1 & \text{if } \mathbb{P}[y = 1|x] \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

i.e. there is no other labeling function with a lower true error rate.

Agnostic PAC Learnability A hypothesis class \mathcal{H} is agnostic PAC learnable w.r.t. a set \mathcal{Z} and a loss function $l : \mathcal{Z} \rightarrow \mathbb{R}_+$ if there exists a function $m_{\mathcal{H}} : (0, 1)^2 \rightarrow \mathbb{N}$ and a learning algorithm such that

for every $\epsilon, \delta \in (0, 1)$ and for every \mathcal{D} over \mathcal{Z} when the algorithm is run $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ samples iid from \mathcal{D} , the algo returns a hypothesis $h \in \mathcal{H}$ such that with probability $1 - \delta$ over the training samples

$$L_{\mathcal{D}}(h) = \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon$$

where $L_{\mathcal{D}}(h) = \mathbb{E}_{z \sim \mathcal{D}}[l(h, z)]$.

2 Logistic Regression

Used for binary classification and equivalent to 1 layer Neural Network [show how](#). Let set of classes $= \{0, 1\}$ and training set be $S = (x^1, y^1), (x^2, y^2), \dots, (x^m, y^m)$ where each $x^i \in \mathbb{R}^d$ and $y^i \in \{0, 1\}$. Classification function will predict the probability of label being 1 (or 0 since their sum is 1). So, we'll have

$$p_m(y = 0|x) + p_m(y = 1|x) = 1.$$

To make predictions using this, we'll use

$$y = \begin{cases} 1 & \text{if } p_m[y = 1|x] > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

The function to model probabilities will be

$$p_{w,b}(x) = \sigma(< w, x > + b)$$

where $W \in \mathbb{R}^d$ and $b \in \mathbb{R}$. To get a probability, we'll pass the linear model through a *sigmoid* function i.e. $\sigma(t) = \frac{1}{1+e^{-t}} = 1 - \sigma(-t)$.

If we define our hypothesis h as above i.e.

$$h_{w,b}(x) = \begin{cases} 1 & \text{if } p_{w,b}[x] > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

then the set $\mathcal{H} = \{h_{w,b} : w \in \mathbb{R}^d, b \in \mathbb{R}\}$ defines the *half spaces* hypothesis class, which we know has a VC-dimension of $d + 1$. [prove this](#).

Now, since $L_{\mathcal{D}}(\mathcal{H}) \leq L_S(\mathcal{H}) + O(\sqrt{\frac{VC(\mathcal{H}) + \log(1/\delta)}{m}})$ for $m \gg d$ ERM should give a small generalization error.

Solving it We need to find $\operatorname{argmin}_{w,b} L_S(h_{w,b})$ which is

$$L_S(h_{w,b}) = \frac{1}{m} \sum_i L(y^i \neq h_{w,b}(x^i))$$

such that for *all* i

$$wx^i + b > 0 \quad \text{if } y^i = 1 \quad \text{and} \quad wx^i + b < 0 \quad \text{if } y^i = 0,$$

This is a linear program which is solvable in polynomial time in m, d if realizable. If not, then finding the argmin is NP-hard. It is also not obvious how to find an approximate solution to this expression.

You can try converting this into a *least-squares* problem i.e.

$$\min_{w,b} \sum_{i \in [m]} (< w, x^i > + b - y^i)^2$$

There is an efficient way to solve this but this is solving the wrong problem! We don't want to value of $< w, x^i > + b$ to be exactly y^i .

MLE Approach Another approach may be to maximize the MLE i.e.

$$\text{Likelihood}(s) = \prod_{i \in [m]} p_{w,b}(y^i | x^i)$$

where $p_{w,b}(y = 1|x) = \sigma(< w, x > + b)$ and $p_{w,b}(y = 0|x) = 1 - \sigma(< w, x > + b)$. This implies that

$$p_{w,b}(y|x) = \hat{y}^y (1 - \hat{y})^{(1-y)} \implies \log(p_{w,b}(y|x)) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$$

So we need to minimize the *cross entropy loss*

$$L_{CE}(w, b; S) = - \left(\sum_{i \in [m]} y^i \log(\sigma(< w, x^i > + b)) + (1 - y^i) \log(1 - \sigma(< w, x^i > + b)) \right)$$

This minima might not always exist. Let's say we keep $y = 1$, then the first term can be taken to $-\infty$ by making the weights arbitrarily large.

3 Multinomial Logistic Regression

Let set of classes = $\{1, 2 \dots k\}$ and training set be $S = (x^1, y^1), (x^2, y^2), \dots (x^m, y^m)$ where each $x^i \in \mathbb{R}^d$ and $y^i \in [k]$.

There are multiple ways to model this, one way is to use *one vs all* approach. The other is to use a logistic regression approach. Here, we take the latter. **Work out the expression for one vs all case.** Use k linear functions as

$$< w_1, x > + b_1, \dots, < w_k, x > + b_k.$$

We'll also make use of the softmax function which takes a vector of inputs (t_1, t_2, \dots) and transforms them to $(\frac{e^{t_1}}{\sum e^{t_i}}, \dots)$. This has the nice property that its sum to one. **add imp props of softmax.**

We then use the *Maximum Likelihood Approach*. Define Likelihood as

$$\prod_{i=1}^m p(y^i | x^i) = p(1|x)^{\mathbb{1}(y=1)} \dots p(k|x)^{\mathbb{1}(y=k)}$$

Maximizing this equivalent to minimizing the *Cross Entropy Loss* **Why is it called cross entropy.**

$$L_{CE}(w, b; S) = - \sum_{i \in [k]} \mathbb{1}(y = 1) \log \left(\frac{e^{< w_i, x > + b_i}}{\sum e^{< w_i, x > + b_i}} \right) = \log \left(\frac{e^{< w_{y^i}, x > + b_{y^i}}}{\sum e^{< w_{y^i}, x > + b_{y^i}}} \right)$$

4 Feed forward Neural Networks

Assume 2 layers - hidden layer and output layer. We never count input layer.

Let $f = (f_1, \dots, f_k)$ such that $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$.

Let $f_i(x) = \sum_{r \in [m]} a_{ir} \rho(< w_r, x > + b_r)$ where $w_r \in \mathbb{R}^d, a_{ir}, b_r \in \mathbb{R}$ and $\rho : \mathbb{R} \rightarrow \mathbb{R}$ is the *activation function*. **Note different types of activation functions and their properties. When to use which.**

This can be written in a compact form as $f(x) = A\rho(Wx + b)$ where each row of A is a_i and each row of W is w_r .

Note that, A takes the role of W in the next layer. So, to generalize to further layers, you'll find:

For l layers: $f(x) = \rho(W^l(\dots \rho(W^2 \rho(W^1 x))))$.

Survey of techniques

1. [Blum and Rivest, 1992] n input units, 2 hidden units, 1 output unit, $|S| = O(d)$ where $x \in \mathbb{R}^d$, Activation function ρ is *threshold function*. It's NP-Hard to fit this network to an S .

2. [Bartlett and Ben-David] Similar architecture as above except k hidden units. For a realizable S , finding weights that fit at least $(1 - \frac{c}{k})$ fraction of examples is NP-hard, where c is some constant.
3. [Sima 2002] One neural, sigmoid activation, NP-hard to train.
4. [BLR 18, MR18] Same results as above but for ReLU networks.
5. [Jones, VN 1998] 1 Hidden layer of sigmoid with units polynomial in d . Output linear with non-negativity. Also NP-hard.

Over-parametrization case Most of the previous work rely on *small networks* or put some restrictions on activation functions. Let's now consider the case when there is overparametrization i.e. hidden layer size $k >$ data points m and the activation used is the one that is practically used - *ReLU*.

Solving ERM Assume some data points in 2-d of the form $S = \{(x^i, y^i), \dots, (x^m, y^m)\}$. Assume 2 layer Neural Network with k hidden units and 1 output unit.

$$\text{So } f(x_i) = \sum_{i=1}^k a_i \text{ReLU}((\langle w, x \rangle + b_i))$$

We need to find parameters w_i, b_i, a_i such that $f(x_i) = y_i$ for all $i \in [m]$.

Proof by construction Let all w_i be in a *special direction* with unit norm. This means that $w_i = w$. The special direction is such that if you take a line perpendicular to that direction, you'll encounter at most one data point over the whole 2-d space. **Prove that such a line always exists.** Let's pass our x through the Neural Network.

$$f(x^1) = a_1 \text{ReLU}((\langle w, x^1 \rangle + b_1)) + a_2 \text{ReLU}((\langle w, x^1 \rangle + b_2)) + \dots \quad (1)$$

$$f(x^2) = a_1 \text{ReLU}((\langle w, x^2 \rangle + b_1)) + a_2 \text{ReLU}((\langle w, x^2 \rangle + b_2)) + \dots \quad (2)$$

$$f(x^3) = a_1 \text{ReLU}((\langle w, x^3 \rangle + b_1)) + a_2 \text{ReLU}((\langle w, x^3 \rangle + b_2)) + a_3 \text{ReLU}((\langle w, x^3 \rangle + b_3)) \dots \quad (3)$$

Let's say I choose b_1 such that $\text{ReLU}((\langle w, x_1 \rangle + b_1)) = \epsilon$ and choose $a_i = y^i/\epsilon$. Then $\text{ReLU}((\langle w, x_1 \rangle + b_1)) = 0$ for x lying beyond ϵ left of x^1 since $\langle w, x \rangle$ is just the projection of x on w . Also value of the NN at x^1 will then be y^1 .

Now consider x^2 . Choose b_2 in similar fashion i.e. $\text{ReLU}((\langle w, x_2 \rangle + b_2)) = 0$ for all x beyond ϵ left of x_2 . The first term in (2) will contribute let's say p . Let output of the second term in (2) be $a_2 p$ then you can choose a_2 such that $a_2 p + q = y^2$. No other term will contribute to this equation since we'll choose their b_i so that they are zero to the ϵ left of respective x_i .

Note that, for a successful construction, you assumed that there is at least some small ϵ distance along w between two points.

Need figure.

Solving ERM proves nothing We showed that we can solve for ERM for $k \geq m$. Now, since $L_{\mathcal{D}}(\mathcal{H}) \leq L_S(\mathcal{H}) + O\left(\sqrt{\frac{VC(\mathcal{H}) + \log(1/\delta)}{m}}\right)$, the second term is greater than 1. **How?** But we knew that $L_{\mathcal{D}}(\mathcal{H})$ can be at most 1 so, this is a *vacuous bound*. This is just an example of the larger observation that traditional bounds fail for the overparametrization case. And if you were to use such an ERM hypothesis in practice, you'll observe a large generalization error.