1 Subjects

- VC Dimension
- Bias Variance
- Regularization
- Validation

2 Notes

2.1 Preceding discussion

Chapter 2 in Learning from data.
What we want to minimize, is the out-of-sample error:

$$E_{out}(g) = \mathbb{E}_{x,y}\left(e\left(g(x), y\right)\right)$$

We can't minimize this, however what we can minimize is the in-sample-error:

$$E_{in}(g) = \frac{1}{\|D\|} \sum_{(x,y)\in D} e\left(g(x), y\right)$$

So we are hoping, or aiming for, that the generalization error $E_{in} - E_{out}$ should be small, so when we minimize E_{in} it benefits E_{out}.
Recall that the Hoeffding inequality provides a way to bound the generalization error:

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$$

We can rephrase this, by introducing a tolerance level δ and assert with probability at least $1 - \delta$ that:

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$$

We notice here, that the bound depends on M, which is the size of the hypothesis set. Unfortunately, most problems has infinite hypotheses, and thus the bound will become meaningless as it goes towards infinity. So we want to replace M by something that stays meaningful as M goes to infinity.

To this end we introduce the growth function, which will formalize the effective number of hypotheses. Furthermore, a *dichotomy* is an N-tuple, generated by a hypothesis, which splits the data into two groups.

The set of dichotomies generated by the hypothesis set $\mathcal{H}$ on the points $x_1, \ldots, x_n$ is defined by:

$$\mathcal{H}(x_1, \ldots, x_N) = \{(h(x_1), \ldots, h(x_N)) \,|\, h \in \mathcal{H})\} \tag{1}$$

One can think of the dichotomies as being the hypothesis set as seen through the eyes of just N points. We can then define the growth function as:

$$m_{\mathcal{H}}(N) = \max_{x_1,\ldots,x_n \in X} |\mathcal{H}(x_1, \ldots, x_N)| \tag{2}$$

I.e. the maximum number of dichotomies that can be generated by $\mathcal{H}$ on any N points. If we just look at binary classification, then the upper limit on the amount of dichotomies for a data-set of size N is:

$$m_{\mathcal{H}}(N) \leq 2^N \tag{3}$$

If $\mathcal{H}$ is capable of generating all possible dichotomies on the data-set, then $m_{\mathcal{H}}(n) = 2^N$ and we say that $\mathcal{H}$ shatter $x_1, \ldots, x_n$. If there is no such data-set of size k that can be shattered by $\mathcal{H}$ then we say that k is a breakpoint for $\mathcal{H}$.

If there is such a breakpoint k, then we know that $m_{\mathcal{H}}(k) < 2^N$. We define $B(N, k)$ as the maximum number of dichotomies on N points, such that no subset of size k can be shattered by these dichotomies. We can then see that:

$$m\mathcal{H}(N) \leq B(N, k) \text{ if } k \text{ is a break point for } \mathcal{H}$$

Sauer's lemma then states that:

$$B(N, k) \leq \sum_{i=0}^{k-1} \binom{N}{i}$$

Which means that:

$$m_{\mathcal{H}}(N) \leq \sum_{i=0}^{k-1} \binom{N}{i}$$

We then see that, if $\mathcal{H}$ has a breakpoint then $m_{\mathcal{H}}(N)$ has a polynomial bound. This is important because when $m_{\mathcal{H}}(N)$ has a polynomial bound, the generalization error will go to zero as $N \to \infty$

2.2 VC Dimension

The Vapnik-Chervonenkis dimension of a hypothesis set $\mathcal{H}$, denoted by $d_{vc}(\mathcal{H})$ or simply d_{vc} is the largest value of N for which $m_{\mathcal{H}}(N) = 2^N$. If $m_{\mathcal{H}}(N) = 2^N$ for all N, then $d_{vc}(\mathcal{H}) = \infty$.

It follows then, that if d_{vc} is the VC dimension for $\mathcal{H}$, then $k = d_{vc} + 1$ is a breakpoint and there are no smaller breakpoints. We can therefore rewrite the previous sum in terms of the VC dimension:

$$m_{\mathcal{H}}(N) \leq \sum_{i=0}^{d_{vc}} \binom{N}{i}$$

We then arrive at the VC Generalization Bound:

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}} \tag{4}$$

with probability $\geq 1 - \delta$

VC-Dimension captures expressiveness/capacity of hypothesis spaces and relate them to generalization. Leads to out-of-sample error equals in-sample-error + model complexity:

$$E_{out}(h) \leq E_{in}(h) + \Omega(N, \mathcal{H}, \delta)$$

$$\Omega(N, \mathcal{H}, \delta) = \sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}}$$

Sampling Complexity

2.3 Bias-Variance decomposition

Page 62

The out-of-sample error for hypothesis $g^{(D)}$ learned on data D is:

$$E_{out}(g^{(D)}) = \mathbb{E}_x \left[(g^{(D)}(x) - f(x))^2 \right]$$

Where $\mathbb{E}_x$ denotes the expected value with respect to x (based on the probability distribution on the input space X)

We can generalize this, to remove the dependence on a specific data-set by taking the expectation with respect to all data-sets:

$$\begin{aligned}
\mathbb{E}_D \left[E_{out}(g^{(D)}) \right] &= \mathbb{E}_D \left[\mathbb{E}_x \left[(g^{(D)}(x) - f(x))^2 \right] \right] \\
&= \mathbb{E}_x \left[\mathbb{E}_D \left[(g^{(D)}(x) - f(x))^2 \right] \right] \\
&= \mathbb{E}_x \left[\mathbb{E}_D \left[g^{(D)}(x)^2 \right] - 2\mathbb{E}_D \left[g^{(D)}(x) \right] f(x) + f(x)^2) \right]
\end{aligned}$$

The term $\mathbb{E}_D \left[g^{(D)}(x) \right]$ gives an 'average function', which we denote by $\bar{g}(x)$ it can be seen as an average function as a result of many data-sets $D_1, \ldots, D_K$ where:

$$\bar{g}(x) \approx \frac{1}{K} \sum_{k=1}^{K} g_k(x) \tag{5}$$

We can now rewrite the expected out-of-sample error in terms of $\bar{g}$:

$$\begin{aligned}
&\mathbb{E}_D \left[E_{out}(g^{(D)}) \right] \\
&= \mathbb{E}_x \left[\mathbb{E}_D \left[g^{(D)}(x)^2 \right] - 2\bar{g}(x) f(x) + f(x)^2 \right] \\
&= \mathbb{E}_x \left[\mathbb{E}_D \left[g^{(D)}(x)^2 \right] - \bar{g}(x)^2 + \bar{g}(x)^2 - 2\bar{g}(x) f(x) + f(x)^2 \right] \\
&= \mathbb{E}_x \left[\mathbb{E}_D \left[(g^{(D)}(x) - \bar{g}(x))^2 \right] + (\bar{g}(x) - f(x))^2 \right]
\end{aligned}$$

The term to the right $(\bar{g}(x) - f(x))^2$ measure how much the average function we would learn using the D different data-sets deviates from the target function, we call this term the bias:

$$\text{bias}(x) = (\bar{g}(x) - f(x))^2$$

The other term, $\mathbb{E}_D\left[(g^{(D)}(x) - \bar{g}(x))^2\right]$ is what we call the variance, which measures the variation in the final hypothesis:

$$\text{var}(x) = \mathbb{E}_D\left[(g^{(D)}(x) - \bar{g}(x))^2\right] \tag{6}$$

We thus arrive at the bias-variance decomposition of out-of-sample error:

$$\begin{aligned}\mathbb{E}_D\left[E_{out}(g^{(D)})\right] &= \mathbb{E}_x\left[\text{bias}(x) + \text{var}(x)\right] \\ &= \text{bias} + \text{var} \\ \text{bias} &= \mathbb{E}_x\left[\text{bias(x)}\right] \\ \text{var} &= \mathbb{E}_x\left[\text{var(x)}\right]\end{aligned}$$

Bias: How well can we actually fit - on average
Variance: How much will data samples lead me astray - on average

We cannot compute actual bias and variance in practice, since they depend on the target function and input probability distribution. So it is a conceptual tool, which is helpful when it comes to developing a model.

There are two typical goals: we want to reduce the variance without significantly increasing bias et vice versa. These goals are achieved through heuristics, regularization being one them.

Intuition The bias term is big if the model has too little capacity to fit the data.

The variance term is big if the model has so much capacity, that it is good at fitting noise (sampling errors) in any training set.

2.4 Regularization

In learning, we want to achieve two things:

1. Ensure that out-of-sample error is close to in-sample-error
   $\mathbb{P}\left[|E_{in} - E_{out}| > \epsilon\right] \leq 2Me^{-2\epsilon^2 N}$ or:
   $E_{out}(h) \leq E_{in}(h) + \Omega(N, \mathcal{H}, \delta)$

2. Minimize in-sample-error E_{in}

So far we have looked at how to prevent underfitting, i.e. how do we fit our data as well as possible. Regularization is about preventing over-fitting, i.e. if we fit our data really well, then our in-sample-error might be really low $E_{in} = 0$, but it's no longer close to the out-of-sample-error $E_{out} >> E_{in}$

This happens because we fit the noise rather than the signal, if we look at the target complexity, e.g. for some polynomial, then if we try to fit it with a hypothesis of h_{10}, then as the target complexity increases there will be more and more data that our hypothesis cannot capture, so even though it is not stochastic noise, it wil look like noise to our hypothesis. This is reflected in the fact that:

- Data increases $\rightarrow$ Overfitting Decreases

- Noise increases → Overfitting Increases
- Target Complexity Increases → Overfitting Increases

So increasing the target complexity seems to introduce noise similar to increasing the noise. In order to avoid this, we want to fit it in a "simpler" way, so we avoid the precise fitting we get from our current techniques.

In linear regression, this is what we currently minimize:

$$E_{in} = \frac{1}{|D|} \sum_{x,y \in D} (w^T x - y)^2$$

If we instead, for different values of α, minimize:

$$E_{in} + \Omega(h) = \frac{1}{|D|} \sum_{x,y \in D} (w^T x - y)^2 + \alpha \|w\|_2^2$$

Which gives us the constrained optimization:

$$\begin{aligned} \text{Minimize: } & \frac{1}{|D|} \sum_{x,y \in D} (w^T x - y)^2 \\ \text{Subject to: } & \lambda \|w\|_2^2 \leq C \end{aligned}$$

Then we get the constrained hypothesis set $\mathcal{H}_{\text{Constrained}}$ where:

$$\mathcal{H}_{\text{Constrained}} \subseteq \mathcal{H} \implies d_{vc}(\mathcal{H}_{\text{Constrained}}) \leq d_{vc}(\mathcal{H})$$

So we get a simpler hypothesis set (even though d_{vc} might be the same), which also means the variance should go down (as the regularization favors h's that look alike, and rules out the ones that are specialized to a single data-set) although the bias should go up. why?

Then for e.g. linear regression we get that:

$$\nabla_w = \frac{1}{n}(2X^T X w - 2X^T y) + \frac{\lambda w}{2n}$$

Which gives us that:

$$w = (X^T X + \lambda I)^{-1} X^T y$$

For gradient descent, we get that when we minimize $E_{in} + \lambda \|w\|_2^2$, the gradient descent step that previously looked like:

$$w_{t+1} = w_t - \alpha \nabla_w f(w_t)$$

will now become:

$$\begin{aligned} w_{t+1} &= w_t - \alpha \nabla_w E_{in}(w_t) - 2\alpha\lambda w_t \\ &= w_t(1 - 2\alpha\lambda) - \alpha \nabla_w E_{in}(w_t) \end{aligned}$$

Thus, every round the $(1 - 2\alpha\lambda)$ factor will make w decay towards the zero vector.

2.5 Validation

Regularization introduces the need for validation in order to be able to train on the data again and again with different values of λ. The need for validation becomes apparent without regularization in more advanced models (like Neural Networks) but regularization is the first technique we encounter.

Validation simply means to take out K points from the data-set and use them to validate the result of the training on the remaining $N - K$ points.

Validation has the issue that as you increase K, the E_{val} estimate tightens but it also increases, as you has less data to train on so E_{out} increases with smaller $N - K$. Analytically:

$$E_{val} = E_{out} \pm \mathcal{O}\left(\frac{\sigma}{\sqrt{K}}\right)$$

As a rule of thumb: $K = \frac{N}{5}$ (20% of data). If we validate and try out different models with different $E_{val}(m_i)$, then we can pick the best and retrain on all the data, hoping that the model we chose is still the best.

Usually if we have few hyperparameters, (like λ) then we can just perform grid search (exhaustively search through a range of parameters). If we have many hyperparameters, we should just do "random sampling" and repeatedly search on promising areas.

We do, however risk that we now overfit on the validation parameters. So what we really want is a training set, a validation set *and a test set*! We will usually split the data-set by 50/25/25% or 60/20/20% depending on the application and the data available.

For small K we get that $E_{out}(h_{all}) \approx E_{out}(h_{train})$ and for large K we get $E_{out}(h_{train}) \approx E_{val}(h_{train})$, but we would like to have both of these.

It turns out that setting $K = 1$ and computing:

$$\begin{aligned} D_n &= D - \{(x_n, y_n)\} \\ e_n &= E_{val}(h_{D_n}) = e(h_{D_n}, (x_n), y_n) \\ E_{cv} &= \frac{1}{N}\sum_{i=1}^{N} e_n \end{aligned}$$

This is hard to analyze, since errors are correlated but it has been used successfully in practice. However, doing this cross-validation is slow, as we have to train N times on $N - 1$ points. Therefore we use K-Fold Cross Validation:

- Split data in K parts of size $\frac{N}{K}$
- Run K rounds of validation with different data-sets of size $N - K$ and take the mean error.

Usually, we will run for $K = 10$ or $K = 5$, as we are impatient beings with deadlines.