CS 165: Foundations of Machine Learning **Winter 2019**

Lecture 8: 01/31/2019

Lecturer: Anima Anandkumar *Scribes: Yu Wu, Sha Sha*

8.1 Generalization

Goal: Give foundations on what is the theory of **complexity**, how well a certain class can **generalize** and a good way to **model** what is happening in real experiments.

Introduction: In machine learning, it is not just optimization. If we optimize on the training set, there is no guarantee it will do well on test set. As the ultimate goal of machine learning is not only to fit the training data, but also the test data (which we have no access to), the out-of-sample performance distinguishes machine learning from standard optimization.

Notations:

Hypothesis class:	$h \in H$; $h : X \to Y$
Loss function:	$L(Y, h(x))$
Expected risk:	$L(h) := E_P[L(Y, h(x))]$
Expected risk minimizer:	$h^* \in \underset{h \in H}{argmin}\ L(h)$
Empirical risk:	$\hat{L}(h) := \frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i))$
Empirical risk minimizer:	$\hat{h} \in \underset{h \in H}{argmin}\ \hat{L}(h)$
Excess risk:	$\hat{L}(\hat{h}) - L(h^*)$

Question: If we have a sub-sample of data from the distribution, how is the empirical risk $\hat{L}(h)$ different from the expected risk $L(h)$, can we bound this?

The difference between empirical risk and the expected risk of any estimators: $\underset{h \in H}{sup} |\hat{L}(h) - L(h)|$. It is called **uniform convergence** because it considers every hypothesis in the hypothesis set. It bounds the excess risk, but is a loose upper bound. It can be bad as it is just a loose bound, but it is convenient as it does not use the nature of the algorithm, like any specialty of the algorithm to get the estimator, the expected risk minimizer and etc. If consider the worst case, we can use union bound, a simple Hoeffding bound (generalization bound based on).

PAC framework: a statement holds in an approximate way, w.p. $1 - \delta$
In the case of excess risk, excess risk $\geq \epsilon$ (approximate), w.p. $1 - \delta$.
if the number of hypotheses is finite: just do union bound.
if the number of hypotheses is not finite: use more sophisticated tool.

8.2 Measures of Complexity

1. Rademacher Complexity

$R_n(H, L) = E[\sup_{h \in H} \sum_{i=1}^{n} \sigma_i L(y_i, h(x_i))]$, where σ_i is Rademcher random variable $\{-1, 1\}$ (for generalization). The intuition behind Rademacher Complexity is to **measure how much we are over-fitting with our training data in a certain way**, meaning how much it fits to random noise.

2. VC-Dimension

The intuition behind VC-Dimension is **the number of points we can fit to if we really want the estimator to memorize a certain number of training points**. VC-Dimension gives the maximum value the estimator can do. If a small number of training points is memorized, the task will generalize better as it cannot memorize all the training data. Therefore, **limiting complexity can help prevent over-fitting and generalize better**.

8.3 Linear Class

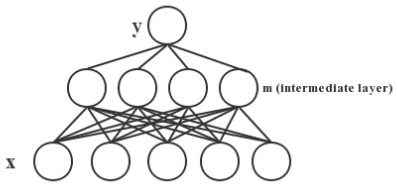
For Linear Class, we care about dimension, and if it is bounded, we can replace it with bound. By limiting function class (sparsity, low rank or different norms), we can replace the ambient dimension with the number of parameters. This is not just parameter-counting, but gives information on how many parameters to learn from data and how many data points we have.

The rule is to have more data points than the number of parameters, which is verbose problems in statistics, meaning enough data to learn better estimator. If with more parameters, which is the high dimension statistics, we need to assume something more. For example, we can assume even though with many parameters, the complexity is bounded. The principle here is to have enough samples to ensure that the complexity is bounded. Therefore, having the number of parameters bigger than d is good for generalization. We should really select function class based on how many samples we have as larger x meaning harder to generalize.

8.4 Neural Networks

1. Complexity

For linear class, the complexity is similar to one-layer neural network. For a two-layer neural network, $||\alpha||_2 \leq B_2^{'}$, ℓ_2 norm is bounded, meaning not all the weights can be very large. $||w_i||_2 \leq B_2$, ℓ_2 norm on each classifier is bounded. Rademacher complexity is the product of these two.



$$y = \sum \alpha_i h(< \mathbf{w_i}, \mathbf{x} > + \mathbf{b_i})$$

Question: How to limit complexity?

Have the norm bounded less than 1. By doing so, the bound won't explode. To make this bound better, collect all the weights in a matrix, which is weight matrix $W = [w_1|w_2|...|w_n]$, $||w|| \leq B_2\sqrt{m}$, $||w|| = B_3$, then we have better bound:

$$looserbound : \frac{B_2^{'}B2\sqrt{m}}{\sqrt{n}}$$

$$betterbound : \frac{B_2^{'}B_3}{\sqrt{n}}$$

For looser bound, we consider each neuron level and takes the number of neurons on each level into the bound. For better bound, we directly get the spectral norm of the weight matrix bounded and the complexity independent of the number of neurons.

Question: How to design a good network with generalization with better bound?

From the matrix level, normalize it with top eigenvalue, which can give sharper generalization bound. Actually, spectral normalization gives properties beyond generalization, as good generalization bound can have other benefits like stability. Having this norm bound will help we ensure regularization and thus good generalization.

2. Approximation

We need to balance both estimation error and approximation error. For linear classifier with good Rademacher Complexity, it will generalize well, the estimation error can be limited, but the approximation error can be large, so we really need to balance both of them and not just look at one of them. It can be complicated because we don't know how one will change based on the other.

Decompose errors for linear regression:

Loss function:

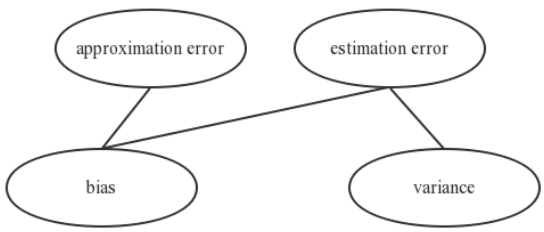
$$L(\mathbf{w}) = E_{\mathbf{x},y}[||y- < \mathbf{w}, \mathbf{x} > ||]$$

$$L(\mathbf{w}) = E_{\mathbf{x}}[< \mathbf{w}, \mathbf{x} > -E[y|\mathbf{x}]]^2 + E[Var(y|\mathbf{x})]$$

$$\hat{L}(\hat{\mathbf{w}}) = E[(E[y|\mathbf{x}]- < \mathbf{w}^*, \mathbf{x} >)^2]+E[< \mathbf{w}^*, \mathbf{x} > - < E(\hat{\mathbf{w}}), \mathbf{x} >]^2+E[< \mathbf{w}^*, \mathbf{x} > - < E[\hat{\mathbf{w}}], \mathbf{x} >]^2+E[Var(y|\mathbf{x})]$$

$$\mathbf{w}^* = \min_{\mathbf{w}} L(\mathbf{w}), \hat{\mathbf{w}} = \min_{\mathbf{w}} \hat{L}(\mathbf{w})$$

We decompose the empirical loss into these parts: $E[(E[y|\mathbf{x}]- < \mathbf{w}^*, \mathbf{x} >)^2]$ is the approximation error inherent. $E[< \mathbf{w}^*, \mathbf{x} > - < E(\hat{\mathbf{w}}), \mathbf{x} >]^2$ is the estimation bias. $E[< \mathbf{w}^*, \mathbf{x} > - < E[\hat{\mathbf{w}}], \mathbf{x} >]^2$ is the estimation variance. $E[Var(y|\mathbf{x})]$ is the inherent noise variance, which is independent of the estimator. Therefore, for a neural network, we decompose the error as follows:



From linear regression to neural network (non-linear): discretize

one-dimension: if choose neurons as step function $\rightarrow$ interpolate to fit the function.
d-dimension: the number of neurons $m \sim e^d$.

Alternative basis to represent functions:

-Polynomial
-Fourier
-Wavelets
-...

If we assume our function has low complexity or is sparse in the certain basis, then our neural network will approximate to that. For the class of smooth functions, we can do the Fourier series, if we assume only need a small number of Fourier units, rather few frequencies to represent the function, then we can avoid the exponential dependencies. As $C = \int ||w|||F(w)|dw$, if bounded, this can give good approximation.

The approximation is roughly $\leq \frac{C}{m}$. We can see that if smoothness is satisfied, it can avoid exponential dependencies. Therefore, different Basis may compact representation, meaning doesn't need too many units to approximate it.

Question: Why deep is better than shallow in practical scenarios?

For polynomial function: $f(x) = (x_1 + x_2)(x_3 + x_4)...$, DEEP

It can be rewritten as: $x_1x_2 + ...$, SHALLOW, which needs exponential number of neurons to represent it.

8.5 Next

Next lecture: bounds in practice.

References