Closing Remarks

compiled by Alvin Wan from Professor Benjamin Recht's lecture

1 Empirical Risk Minimization

Risk is the expected loss of our prediction function, or in colloquial terms, the guess on how well we'll predict:

$$R[f] = E[\text{loss}(f(x), y)]$$

Machine learning is only as good as your data, because that's the only way we can access the model. Samples expose the actual distribution of our data, but only partially. Our empirical risk, or training error, is the following:

$$R_T[f] = \frac{1}{n} \sum_{i=1}^{n} \text{loss}(f(x_i), y_i)$$

We can measure and optimize this empirical risk. However, we can guess how well this estimator serves as a proxy for our actual risk. Let us consider the **Fundamental Theorem of Machine Learning**:

$$R[f] = (R[f] - R_T[f]) + R_T[f]$$

where the last term $R_T[f]$ is the training error, and $R[f] - R_T[f]$ is our generalization error. Only our training error is guaranteed to be observable. We can measure generalization only assuming that we have some holdout data that is representative of all possible data. In effect, our assumption is the following:

$$R_V[f] - R_T[f] \approx R[f] - R_T[f]$$

If your data is i.i.d., $(x_1, y_1) \dots (x_n, y_n)$, then the scale of our error is roughly $R_T[f] - R[f] \approx O(\sqrt{\frac{d}{n}})$. If $d > n$, then we need to regularize.

It's much more important that $R[f] - R_V[f] \approx O(\sqrt{\frac{\log K}{M}})$ is small, wjere K is the number of models used for cross validation. The ideal validation is run on newly-generated data in exactly the same way old data was generated, independent of the training set.

2 Pipeline

- Data
- Features
- Train Model

We have some rules of thumb for features:

- Text: Bag of words, n-grams, histograms
- Vision: Pixels, gradients, histograms of gradients, wavelets
- Medicine: age, gender, family histories, blood tests

Here is a survey of methods:

- linear predictors $w^T x$
- Lifting: $x \to [x_1, x_2, x_1 x_2 \dots]^T$ (nonlinear)
- Kernel trick: $x \to \phi(x)$, $\phi_x(z) = k(x, z)$
- Neural networks: $f(x, \theta) \to$ optimize w.r.t. θ

Another way to build features is to use binning, histograms, or trees. With binning, we split data per ranges of values, such as $[a > 0, a < 0]$. Always compare against nearest neighbors. In general, you should not be hyper-sensitive to continuous-valued features e.g., necessitating the third or fourth decimal place. A model is **stable** (i.e., robustness) if $f_T \approx f_{T\ \{x_i, y_i\}}$ means a good model. One theorem states that if the previous statement holds, we know that generalization error is fairly low.

3 Unsupervised Learning

We can either reduce dimensions (PCA) or cluster (k-means, agglomerative, spectral).

Stochastic gradient descent is one algorithm that is well-suited for all the methods presented in this course. n should be fairly large, and pick a learning α just large enough so that you do not diverge, and decrease α as the algorithm runs. 200 epochs is a good upper bound for overfitting.

After this course, you can take the following to further your knowledge in this domain:

- **Optimization** EE127A, EE227C
- **Probability** EE126, Stat 134, Stat 210A, 210B
- **Applications** NLP, Vision, Robotics

Learning theory teaches active learning and experiment design. Scalable machine learning is one field to explore. Safety, reliability, robustness are also possible fields to explore.