# learning theory

- “For me what separates the people that really understand and get machine learning from the people that read a textbook and sort of worked thru the math is the part that I’m going to talk about now” - Andrew Ng.
- for an intuitive feel of bias/variance tradeoff, see the infographic here.
- bias/variance tradeoff: bias is roughly means underfitting → where the model you’ve chosen is just not the right one for the thing you’re trying to learn so no matter how much data you have, you can’t get it to work. variance is the opposite → its overfitting, learning the intricacies of the training set, bad at predicting things outside the training set.
  - The *bias* is error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).
  - The *variance* is error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).
- the learning algorithms we’ve seen can all be captured as minimizing some training error $\epsilon$ (also called empirical risk or empirical error in learning theory). when we train, we want to minimize this error for the training set. but really what we care about at the end of the day is how well our hypothesis makes predictions on examples it hasn’t seen before (called generalization error: $P(h_\theta(x) \neq y)$). so when does empirical risk minimization give us good generalization error?
- Andrew then goes on to state the hoeffding inequality, which we’ve already seen in our statistics notes!
- Lets say we are choosing from a finite set of hypothesis during training. lets see how empirical risk minimization does on generalization error. we will define training error for now as the number of training examples incorrectly classified over the number of training examples.
- Assuming our training examples are i.i.d, we can use hoeffding’s inequality to show that the probability that the difference between generalization error and training error is big is pretty small. note that this is true for any hypothesis and says nothing about whether the hypothesis is good or bad.
- From the lecture notes:

$$\begin{aligned} P(\exists\, h \in \mathcal{H}. |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(A_1 \cup \cdots \cup A_k) \\ &\leq \sum_{i=1}^{k} P(A_i) \\ &\leq \sum_{i=1}^{k} 2\exp(-2\gamma^2 m) \\ &= 2k\exp(-2\gamma^2 m) \end{aligned}$$

If we subtract both sides from 1, we find that

$$\begin{aligned} P(\neg\exists\, h \in \mathcal{H}. |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(\forall h \in \mathcal{H}. |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| \leq \gamma) \\ &\geq 1 - 2k\exp(-2\gamma^2 m) \end{aligned}$$

(The "¬" symbol means "not.") So, with probability at least $1-2k\exp(-2\gamma^2 m)$, we have that $\varepsilon(h)$ will be within $\gamma$ of $\hat{\varepsilon}(h)$ for all $h \in \mathcal{H}$. This is called a *uniform convergence* result, because this is a bound that holds simultaneously for all (as opposed to just one) $h \in \mathcal{H}$.

- We can use this result, that's based on hoeffding's inequality to work out a minimum training set size to get the desired difference between training error and generalization error with given probability. this is called "sample complexity" → how large a sample do we need to accomplish a given probability of error for **any** hypothesis.

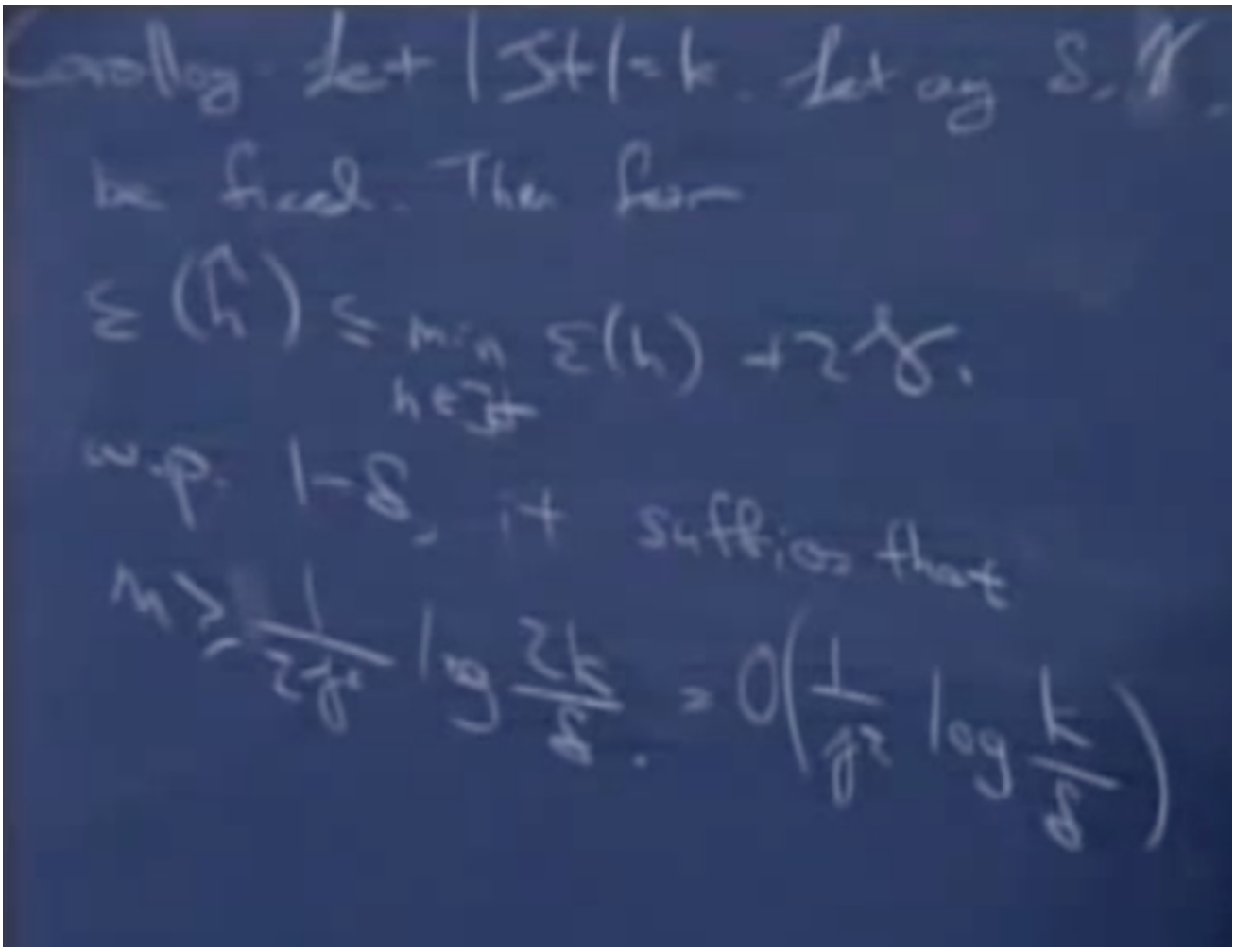
$$m \geq \frac{1}{2\gamma^2} \log \frac{2k}{\delta},$$

- Notice where this is going → given enough data of the right mix, hoeffding's is basically telling us that training error is a good proxy for generalization error, which is some justification for why we choose to minimize training error when what we really care about is generalization error.
- Using hoeffding's inequality as described above and doing the manipulations, we can show that with probability $1-\delta$, where $k$ is the size of the hypothesis set:

$$\varepsilon(\hat{h}) \leq \left(\min_{h \in \mathcal{H}} \varepsilon(h)\right) + 2\sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}.$$

- where the min indicates the best hypothesis (in terms of generalization error).

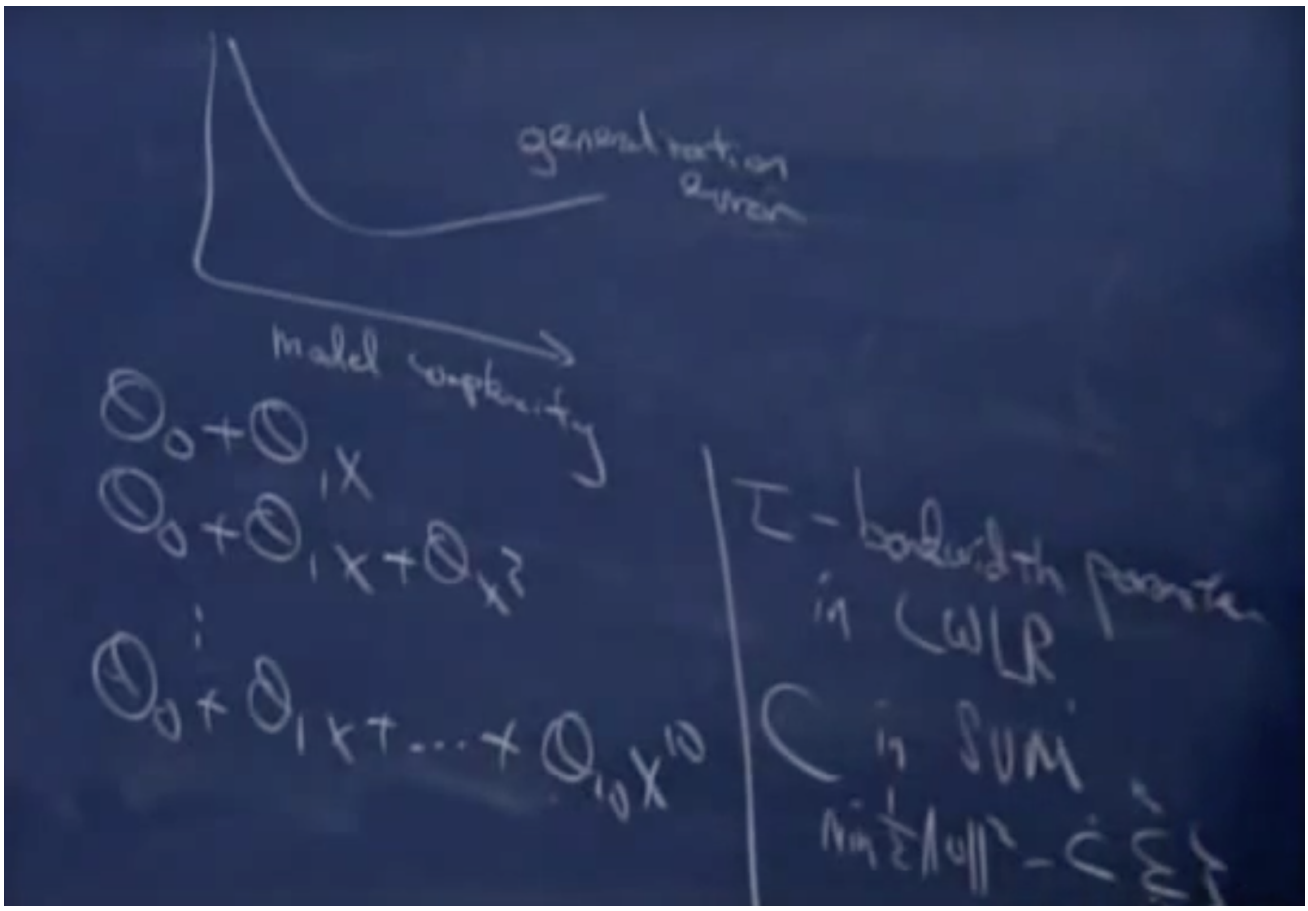
- The above is a more formal way to state the bias/variance tradeoff. if we have a smaller hypothesis class, $k$ is smaller but the best generalization error is higher than if we had a bigger hypothesis class to pick from.




- Note that $m \geq O(log(k))$. this will come up later. In this result, the error ($\gamma$) and probability of doing worse than the error $(1-\delta)$ are held fixed.
- Lets use the above result. If we have a linear hypothesis with $d$ parameters, and each is a 64 bit floating point number, then the size of the hypothesis space is $k = 2^{64d}$, which means that $m \geq O(d)$ (plugging $k = 2^{64d}$ above.
- Definition: Given a training set with feature values $S = x_1, ..x_n$, we say that a hypothesis class $\mathcal{H}$ shatters $S$ if for all possible assignments of labels to $S$, there is some $h \, \epsilon \, \mathcal{H}$ that can realize that assignment. Note that this need not be the same $h$.
- We see that shattering means something like "can this set of hypotheses draw a decision boundary for **any** assignment of labels given a set $S$".
- The Vapnik–Chervonenkis dimension (VC dimension) of a set $S$ is the size of the largest set shattered by $\mathcal{H}$. If the hypothesis class can shatter arbitrarily large sets, then the VC dimension is $\infty$. Note that if the VC dimension is 3, it just means that there exists a set of size 3 that can be shattered. It does not mean that all sets of size 3 can be shattered.
- the best known result in all of learning theory → ng does not prove this:

Let $\mathcal{H}$ be given and let $VC(\mathcal{H}) = d$. Then with probability $1 - \delta$, we have that $|\epsilon(h) - \hat{\epsilon}(h)| \leq O(\sqrt{\frac{d}{m}log(\frac{m}{d}) + \frac{1}{m}log(\frac{1}{\delta})})$. the thing on the left is the difference between the generalization error and the training error.

- ng does not prove this or give any intuition 😓:
  - "The take away is that the number of training examples needed is linear in the VC dimension of the hypothesis class"
  - "for most reasonable hypothesis classes, the VC dimension is basically the same as the number of parameters"
- to recap, the bias/variance tradeoff is basically saying don't choose a model that is too simple (underfit, high variance) or too complex (overfit, high variance)
- model selection:




*the graph shows that high model complexity overfits (bad generalization error) and very low complexity underfits (again, bad generalization error)*

- we will now come up with an algorithm to choose a model (number of degrees in polynomial, the $C$ in SVMs etc). we will assume we are now choosing between a finite set of models. we can't just go by training error → this will lead us to overfit. some techniques:
  - hold out validation: split training set to training set (70%) and cross validation set (30%). train each model on thus picked training set, then test on validation set and pick the model with the least error on validation set. we could then go and train the chosen model on the whole available training data.
  - another variation on this is k-fold cross validation where we break up training set into $k$pieces, train on $k - 1$ pieces, and test on remaining piece. and then do this with each piece as hold out and average the error. again, once we've chosen the model we can go back and re-train on the whole

data. the advantage here is that if the data we have is precious little we are making better use of the little data we have than in the above method.
    - an extreme of this is $k = m$, the number of training examples. this is called leave-one-out cross validation.
- feature selection:
    - if there are $n$ features, there are $2^n$ possible set of features we can choose. this is too many to try all possible feature sets and choose 1.
    - a heuristic algorithm to pick a features set is as follows: start with a null set of features. go thru each possible feature set and see which single feature is best. now add that to the feature set. now go thru the remaining and see which single addition makes the best improvement (by cross validation), and so on. once you hit some threshold $k$ of max features, you can terminate OR once it stops getting too much better by the incremental feature. his is called forward selection algorithm.
    - Similarly, backward selection starts with all features and drops the worst ones iteratively.
    - If we have a very high number of features (typically in text problems), then the above algorithms aren't really feasible. in that case, to do feature selection, we can use some measure (that we come up with) of how information each feature is to do the prediction. one way to do this is to compute correlations between $x_i$ and $y$ for each $x, y$ and drop the ones that have low correlation. One such example measure is the KL divergence between $p(x, y)$ and $p(x)p(y)$. This is called the Mutual Information(MI). a high KL divergence indicates that $p(x, y)$ is and $p(x)p(y)$ are highly non-independent, which means they say things about each other, which means it is a feature worth trying.
    - Jensen-Shannon divergence is a symmetrized, smooth version of KL divergence defined as:
      $M = \frac{1}{2}(P + Q)$
      $JSD(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M)$
    - feature selection is also a way to prevent overfitting.
- while doing linear regression we chose $\theta$ so as to maximize likelihood of the data (maximize $\prod p(y_i|x_i, \theta)$). this formulation thinks of $\theta$ as some fixed, unknown value and then finds it. This is called the frequentist approach → where you believe $\theta$ has some fixed, unknown value.
- the alternative is the bayesian approach where we say lets treat $\theta$ as a random variable and we'll put a prior over it and we calculate $P(\theta|S)$ ($S$ is the training data) using bayes' rule: $\prod_i^m P(y_i|x_i, \theta)P(\theta)$. This $P(\theta|S)$ is the posterior. Now, given a new $x$, we make a prediction $P(y|x, \theta)$ by doing $\int_\theta P(y|x, \theta)P(\theta|S)d\theta$. This is called the bayesian approach. note the contrast with the frequentist approach.
- often, it is computationally intensive to do the full bayesian approach, so another way to do that is to estimate the most likely posterior value of $\theta$ by doing arg max $P(\theta|S)$ using bayes' formula $P(S|\theta)P(\theta) = \prod P(y_i|x_i, \theta)P(\theta)$. this estiamte of $\theta$ is called Maximum A Priori Estimate (MAP estimate). We do inference using $\theta_{MAP}^T x$.
- Just like with MLE, we worked out that for linear regression, the cost function is going to be least squares loss, if we do MAP estimate and work out the cost function, it turns out to be least squares loss $+\lambda||\theta||^2$, which penalizes the large $\theta$'s. we can use cross validation to choose $\lambda$.
- Online learning:
    - the stuff we've seen so far is called batch learning algorithms, because we are given a training set and then the model is set and then we use it for prediction.
    - algorithms where you do a mix of prediction and training as you go is called online learning. so its like you're given an example, you predict, you're told the true answer, you update the model, and shown example again and so on.

- stochastic gradient descent is a natural fit for online learning (think about why)