

CSE 6740 Lecture 13

Which Model (Parameters) Should I Use? (Learning Theory and Generalization)

Alexander Gray
agray@cc.gatech.edu

Georgia Institute of Technology

Today

- ① More on generalization
- ② Model combination methods

More on generalization

VC theory and the bootstrap.

VC Dimension

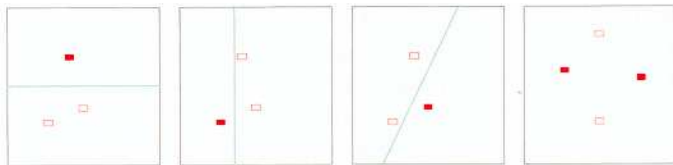
The previous estimates of optimism are mostly applicable to simple models and are likelihood-based. A more general measure of model complexity is given by VC theory.

The *Vapnik-Chervonenkis (VC) Dimension* of the function class $\mathcal{F}_\theta : \mathbb{R}^D \rightarrow \{0, 1\}$ indexed by parameter(s) θ is defined to be the largest number of points (in some configuration and under some labeling) that can be shattered by members of the class.

A set of points is said to be *shattered* by a class of functions if, no matter how we assign the location and label for each point, a member of the class can perfectly separate them.

VC Dimension

For example, the linear decision function can shatter any three points in the plane, but not any four. Hence the VC dimension of this function class is 3. In general the linear decision function in D dimensions has VC dimension $D + 1$, which is also the number of free parameters.



VC Dimension

A nonlinear family may have infinite VC dimension, because by appropriate choice of its parameters θ , any set of points can be shattered by this class.

The VC dimension can be defined for a real-valued function class $\mathcal{F}_\theta : \mathbb{R}^D \rightarrow \mathbb{R}$ as the VC dimension of the indicator class $\mathcal{F}_\theta = \{I(f_\theta(x) - \beta > 0)\}$, where β takes values over the range of f .

VC Bounds

For two-class classification, for *every* model in a function class with VC dimension h , with probability $1 - \eta$

$$R(M) \leq \hat{R}_{\text{tr}}(M) + \frac{\epsilon}{2} \left(1 + \sqrt{1 + \frac{4\hat{R}_{\text{tr}}(M)}{\epsilon}} \right) \quad \text{where} \quad (1)$$

$$\epsilon = c_1 \frac{h[\log(c_2 N/h) + 1] - \log(\eta/4)}{N}. \quad (2)$$

For regression,

$$R(M) \leq \frac{\hat{R}_{\text{tr}}(M)}{(1 - c_3 \sqrt{\epsilon})_+}. \quad (3)$$

'Recommended' values are $c_1 = c_2 = c_3 = 1$ for regression.

Structural Risk Minimization

The bounds suggest that the optimism increases with h and decreases with N in qualitative agreement with the AIC term $|M|/N$.

Note that these bounds hold for every member (parameter setting) of the function class; AIC described the behavior of a specific member of the class (the MLE).

Structural risk minimization chooses the function class with the smallest value of the upper bound.

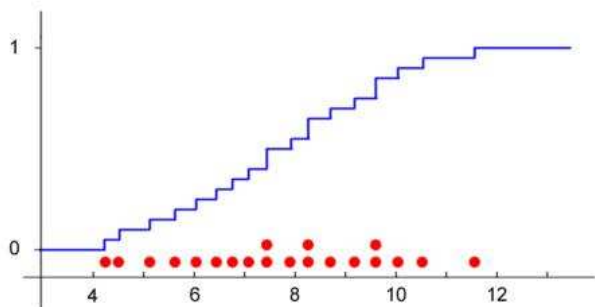
Note that VC theory does not give an actual estimate of the test error, only an upper bound on it. It is also not always easy to derive the VC dimension of a model class.

Empirical Distribution Function

Let $X_1, \dots, X_N \sim F$ be IID. The *empirical distribution function* \hat{F}_N is the CDF that puts mass $1/N$ at each data point X_i :

$$\hat{F}_N(x) = \frac{\sum_{i=1}^N I(X_i \leq x)}{N} \quad (4)$$

where $I(X_i \leq x) = 1$ if $X_i \leq x$, otherwise 0.



Empirical Distribution Function

For any fixed value x ,

$$\mathbb{E} \left(\widehat{F}_N(x) \right) = F(x). \quad (5)$$

$$\mathbb{V} \left(\widehat{F}_N(x) \right) = \frac{F(x)(1 - F(x))}{N}. \quad (6)$$

$$\widehat{F}_N(x) \xrightarrow{P} F(x). \quad (7)$$

This is called the *Glivenko-Cantelli Theorem*: If $X_1, \dots, X_N \sim F$,

$$\sup_x \left| \widehat{F}_N(x) - F(x) \right| \xrightarrow{P} 0. \quad (8)$$

Bootstrap

The bootstrap is a way we can pretend we have a way to get samples from the underlying distribution.

We can use this to estimate test errors and confidence intervals, which are effectively about unseen data from the underlying distribution.

Let $T = g(X_1, \dots, X_N)$ be a statistic, or some function of the data. Suppose we want to know $\mathbb{V}_F(T)$, the variance of T .

For example if $T = \bar{X}$ then $\mathbb{V}_F(T) = \sigma^2/N$ where $\sigma^2 = \int (x - \mu)^2 dF(x)$ and $\mu = \int x dF(x)$. Thus the variance of T is a function of F .

Bootstrap Variance Estimation

There are two steps:

- ① Estimate $\mathbb{V}_F(T)$ with $\mathbb{V}_{\hat{F}}(T)$.
- ② Approximate $\mathbb{V}_{\hat{F}}(T)$ by drawing samples from \hat{F} .

For $T = \bar{X}$ we have for Step 1 that $\mathbb{V}_{\hat{F}}(T) = \hat{\sigma}^2/N$ where $\hat{\sigma}^2 = \frac{1}{N} \sum_{i=1}^N N(X_i - \bar{X})^2$. In this case we are done. But when we don't know the form of $\mathbb{V}_{\hat{F}}(T)$ we have to approximate it using samples.

Bootstrap Variance Estimation

Suppose we draw an IID sample T_1, \dots, T_B from the distribution of T , which we'll call G . By the law of large numbers, as $B \rightarrow \infty$,

$$\bar{T} = \frac{1}{B} \sum_{b=1}^B T_b \xrightarrow{P} \int t dG(t) = \mathbb{E}(T) \quad (9)$$

i.e. if we draw a large sample from G , we can use the sample mean to approximate $\mathbb{E}(T)$. Similarly, we can use the sample variance to approximate $\mathbb{V}(T)$:

$$\frac{1}{B} \sum_{b=1}^B (T_b - \bar{T})^2 \xrightarrow{P} \mathbb{V}(T). \quad (10)$$

Bootstrap Variance Estimation

How do we get at the distribution of T ? All we have are X values. We can talk about their distribution, F . If we could sample values X_1^*, \dots, X_N^* from F , we could compute $T(X_1^*, \dots, X_N^*)$.

How do we get at F ? We'll use \hat{F} in its place.

How can we draw samples from \hat{F} ? Note that \hat{F} puts mass $1/N$ at each data point X_1, \dots, X_N . Thus drawing an observation from \hat{F} is equivalent to drawing one point at random from the original dataset. Thus: To simulate X_1^*, \dots, X_N^* from \hat{F} it suffices to draw N observations *with replacement* from X_1, \dots, X_N .

Bootstrap Variance Estimation

Here is the method:

- ① Draw $X_1^*, \dots, X_N^* \sim \hat{F}$.
- ② Compute $T^* = g(X_1^*, \dots, X_N^*)$.
- ③ Repeat the above two steps B times to obtain T_1^*, \dots, T_B^* .
- ④ $v_{\text{boot}} = \frac{1}{B} \sum_{b=1}^B \left(T_b^* - \bar{T}^* \right)^2$ where $\bar{T}^* = \frac{1}{B} \sum_{b=1}^B T_b^*$.

Keep in mind that two approximations are taking place:

$$\mathbb{V}_F(T) \approx \mathbb{V}_{\hat{F}}(T) \approx v_{\text{boot}}. \quad (11)$$

The first approximation is the more significant of the two.

Bootstrap Confidence Intervals

How can we obtain a confidence interval for T (in other words, (a, b) such that $\mathbb{P}_F(T(F) \in (a, b)) = 1 - \alpha$)? This would tell us the probable range of our predictions, say.

We could use the Normal interval: $T \pm z_{\alpha/2} \hat{se}_{\text{boot}}$ where $\hat{se}_{\text{boot}} = \sqrt{v_{\text{boot}}}$. This is not accurate unless the distribution of T happens to be close to Normal.

A better method is to use the interval $(T_{1-\alpha/2}^*, T_{\alpha/2}^*)$, corresponding to quantiles of T_1^*, \dots, T_B^* . This is a true confidence interval.

Bootstrap Test Error Estimation

How can we use the bootstrap to estimate test error? This is a bit trickier since we can draw samples for both the training and validation sets, but when they overlap we introduce bias. One way to approach this is to use the *leave-one-out bootstrap*:

$$\hat{R}_{\text{looboot}} = \frac{1}{N} \sum_{i=1}^N \frac{1}{|C^{-i}|} \sum_{b \in C^{-i}} L(y_i, \hat{f}_b^*(x_i)) \quad (12)$$

where C^{-i} is the set of indices of the bootstrap samples b that don't contain observation i .

Bootstrap Test Error Estimation

This can be improved to handle the fact that we are using fewer training data than possible, to obtain something called the *.632+ bootstrap*:

$$\hat{R}_{.632+} = (1 - \hat{w})\hat{R}_{\text{tr}} + \hat{w}\hat{R}_{\text{looboot}} \quad (13)$$

where

$$\hat{w} = .632 / \left(1 - .368 \left(\frac{\hat{R}_{\text{looboot}} - \hat{R}_{\text{tr}}}{\hat{\gamma} - \hat{R}_{\text{tr}}} \right) \right) \quad (14)$$

with $\hat{\gamma} = \frac{1}{N^2} \sum_{i=1}^N \sum_{i'=1}^N L(y_i, \hat{f}(x_{i'}))$, which considers all possible combinations of targets and predictors to estimate the error rate if the inputs and outputs were independent.

Model combination

As long as we're building a bunch of models, why not use them all?

Bagging

This is called *bagging* (short for “bootstrap aggregation”):

- ① Draw $X_1^*, \dots, X_N^* \sim \hat{F}$.
- ② Compute $\hat{f}^* = \hat{f}(X_1^*, \dots, X_N^*)$.
- ③ Repeat the above two steps B times to obtain $\hat{f}_1^*, \dots, \hat{f}_B^*$.
- ④ $\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b^*(x)$.

This approaches the estimator $\hat{f}_{\text{ag}}(x) = \mathbb{E}_{\hat{F}} \hat{f}_b^*(x)$ as $B \rightarrow \infty$.

Bagging

Under squared-error loss, with IID data $\sim F$,

$$\mathbb{E}_F \left(Y - \hat{f}^*(x) \right)^2 = \mathbb{E}_F \left(Y - \hat{f}_{\text{ag}}(x) + \hat{f}_{\text{ag}}(x) - \hat{f}^*(x) \right)^2 \quad (15)$$

$$= \mathbb{E}_F \left(Y - \hat{f}_{\text{ag}}(x) \right)^2 + \mathbb{E}_F \left(\hat{f}^*(x) - \hat{f}_{\text{ag}}(x) \right)^2$$

$$\geq \mathbb{E}_F \left(Y - \hat{f}_{\text{ag}}(x) \right)^2. \quad (17)$$

So the true population average never increases MSE. This suggests that bagging will almost always reduce the MSE.

Bagging

The extra error due to using only one bootstrap sample is due to the variance of $\hat{f}^*(x)$ around its mean $\hat{f}_{\text{ag}}(x)$.

This argument can't be made for 0-1 loss. In that case, bagging a good classifier can make it better but bagging a bad classifier can make it worse.

Bayesian Model Averaging

We can think of the probability of making any prediction \hat{y} in terms of its posterior distribution

$$f(\hat{y}|\{x\}_N) = \sum_j f(\hat{y}|M_j, \{x\}_N) f(M_j|\{x\}_N) \quad (18)$$

and its posterior mean

$$\mathbb{E}f(\hat{y}|\{x\}_N) = \sum_j \mathbb{E}(f(\hat{y}|M_j, \{x\}_N)) f(M_j|\{x\}_N). \quad (19)$$

This prediction is a weighted average of the individual predictions, with weights proportional to the posterior probability of each model.

Bayesian Model Averaging

Recall the posterior for a model

$$f(M|\{x\}_N) \propto f(M)L(\{x\}_N|M) \quad (20)$$

$$\propto f(M) \int f(\{x\}_N|\theta, M)f(\theta|M)d\theta. \quad (21)$$

We could try to compute this directly, or use the BIC approximation.

Stacking

Here the weight on each model was given by the model's posterior probability. What if we just say we want the *optimal* weighting of the predictions of our models, $\hat{f}_j(x)$? Under squared-error loss, if F is the distribution of the data, this would be

$$\hat{w} = \{\hat{w}_j\} = \arg \min_w \mathbb{E}_F \left(Y - \sum_j w_j \hat{f}_j(x) \right)^2. \quad (22)$$

This cannot be worse than any single model:

$$\mathbb{E}_F \left(Y - \sum_j \hat{w}_j \hat{f}_j(x) \right)^2 \leq \mathbb{E}_F \left(Y - \hat{f}_j(x) \right)^2. \quad (23)$$

Stacking

If we just did this replacing the expectation by an average over the training set, the most complex model would get weight 1 and all others would get weight 0, since it would be one that best fits the training set.

So we must account for generalization somehow. *Stacking*, short for stacked generalization, uses these weights:

$$\hat{w}_{\text{stack}} = \arg \min_w \sum_{i=1}^N \left(y_i - \sum_j w_j \hat{f}_j^{-i}(x_i) \right)^2, \quad (24)$$

where \hat{f}_j^{-i} denotes the prediction of the model trained without the i^{th} point. Then $\hat{f}_{\text{stack}}(x) = \sum_j \hat{w}_{\text{stack}} \hat{f}_j(x)$.

Stacking

Note that LOOCV is a special case, if we restrict the minimization to weight vectors which have one weight equal to 1 and all the others 0.

This general idea extends to other models to combine the predictors. The formulation with constant weights corresponds to linear regression.

Boosting

Now consider a sequential method which selects the next model based on all the ones before: $\hat{f}_1(x), \hat{f}_2(x), \dots$

Suppose we're doing classification, thus considering a function class $\mathcal{F}_\theta : \mathbb{R}^D \rightarrow \{-1, 1\}$. The final classifier is

$$\hat{f}_{\text{boost}}(x) = \text{sgn} \left(\sum_j \alpha_j \hat{f}_j(x) \right) \quad (25)$$

The weights $\{\alpha_j\}$ will give higher influence to the more accurate classifiers in the sequence.

Boosting

Classically, each successive model consists of the same function class and different parameters. These are obtained using different weightings of the training points: $\{w\}_N^1, \{w\}_N^2, \dots$

Each successive classifier will increase the weights on points that were misclassified by the last classifier and decrease the weights on points that were correctly classified.

There are many variants of this scheme. The most popular is called AdaBoost, shown next.

Boosting

- 1 Initialize each weight to be $w_i = 1/N$.
- 2 For $j = 1$ to J :
 - 1 Fit a classifier $\hat{f}_j(x)$ to the training data using weights $\{w_i\}$.
 - 2 Compute its weighted 0-1 error:
$$e_j = (1/\sum_{i=1}^N w_i) \sum_{i=1}^N w_i I(y_i \neq \hat{f}_j(x_i)).$$
 - 3 Set $\alpha_j = \log((1 - e_j)/e_j)$.
 - 4 Set $w_i = w_i \exp(\alpha_j I(y_i \neq \hat{f}_j(x_i)))$.
- 3 Output $\hat{f}_{\text{boost}}(x) = \text{sgn}(\sum_j \alpha_j \hat{f}_j(x))$.

Boosting

In practice it can use weak learners (those with just over 50% accuracy) to create an overall model with high accuracy.

This can be shown to minimize the loss $L(y, \hat{f}(x)) = \exp(-y\hat{f}(x))$.

The exponential weighting of misclassified points causes a robustness issue. In problems with a significant number of severely misclassified points, boosting does poorly.

Boosting methods can be formulated for any kind of loss.

Which Model Selection Method?

The simpler the model, the less you have to do. For more accurate model selection, which is needed by more complex models, use resampling methods.

Gaussian? Just use the training data. Linear regression? Use Mallows' C_p . Nonparametric? Use cross-validation or the bootstrap.

For ultimate prediction performance, use model combination methods.

In general: More accuracy requires more computation.

Main Things You Should Know

- What VC-based bounds are about
- What the bootstrap is and what it's for
- What bagging is and its properties
- What stacking is and its properties
- What boosting is and its properties