

Model Selection

January 9, 2012

A common problem in machine learning is to choose a model class for modeling data (e.g., in classification, regression or density estimation). As we have seen earlier, more complex model classes typically yield better empirical error but worst generalization error due to overfitting resulting from model-complexity.

How then should we choose a model? There is a famous principle, phrased by William of Occam (or Ockham). It says: "entia non sunt multiplicanda praeter necessitatem", or "entities must not be multiplied beyond necessity" (or as Einstein put it: "Make everything as simple as possible, but not simpler").

More formally, assume we have a classification problem and we can apply different hypothesis classes to it: $\mathcal{H}_1, \dots, \mathcal{H}_k$. Assume that larger i corresponds to a more complex \mathcal{H}_i as captured by its VC dimension (i.e., $VCdim(\mathcal{H}_i) \geq VCdim(\mathcal{H}_{i-1})$). Which class should we work with? Intuitively, more complex classes are more likely contain a hypothesis that is closer to the real rule. However, learning with these is likely to result in over-fitting. What we would really like to do is find the class that would result in a minimum generalization error. Although we can't estimate this error directly it turns out it can be approximate by a sum of the training error (often called the accuracy term) plus a term that increases with the complexity of the class (often called a complexity term). The first of these decreases with model complexity and the second increases. Their sum will thus have a minimum at some intermediate model complexity.

There are various ways of approximating generalization. We review a VC based one, and a Bayesian based one.

1 VC Approach - Structural Risk Minimization

Recall the main theorem saying that:

$$\mathbb{P}_{S_n \sim p} \left[\sup_{h \in \mathcal{H}} |e_S(h) - e_p(h)| \geq \epsilon \right] \leq 4\Pi_{\mathcal{H}}(2n) e^{-\frac{n\epsilon^2}{8}}$$

Recall that we can use it to obtain a confidence interval on generalization error given training error:

Proposition: For every $\delta > 0, n$ the following holds for all h with probability greater than $1 - \delta$:

$$e_S(h) - \sqrt{\frac{8}{n} \left[d \ln \left(\frac{2en}{d} \right) + \ln \frac{4}{\delta} \right]} \leq e_p(h) \leq e_S(h) + \sqrt{\frac{8}{n} \left[d \ln \left(\frac{2en}{d} \right) + \ln \frac{4}{\delta} \right]} \quad (1)$$

Lets say we have n samples and we want a bound on $e_p(h)$ that holds with probability at least $1 - \delta$ over the samples S_n drawn from p . Given n, δ we want the smallest ϵ such that the uniform convergence bound holds with confidence δ . To get confidence δ we need:

$$4\Pi_{\mathcal{H}}(2n)e^{-\frac{n\epsilon^2}{8}} \leq \delta \quad (2)$$

We know that $\Pi_{\mathcal{H}}(2n) \leq \left(\frac{2en}{d}\right)^d$ so if we have

$$4 \left(\frac{2en}{d}\right)^d e^{-\frac{n\epsilon^2}{8}} \leq \delta$$

This will also imply confidence δ . The minimum ϵ for which this holds is when we have equality.

$$\epsilon(n, \delta) \leq \sqrt{\frac{8}{n} \left[d \ln \left(\frac{2en}{d} \right) + \ln \frac{4}{\delta} \right]}$$

From which we get the desired confidence interval.

The above proposition suggests we can use the training error plus a term dependent only on n, d as an estimate for generalization error. Thus, a reasonable strategy for choosing between hypothesis classes $\mathcal{H}_1, \dots, \mathcal{H}_k$ is to apply the bound to each and choose the one that yields the minimal estimate of generalization (typically the upper limit of the confidence interval is used). This was suggested by Vapnik and is called structural risk minimization. It can be shown to be consistent in some cases.

2 Bayesian Information Criterion

We turn to a different approach to model selection. We focus on the parameter estimation setting, but equivalent results for classification are also available.

Assume we our in a Bayesian setting where parameters are generated in the following way:

- Choose a model class \mathcal{H}_i with probability α_i .

- Choose a parameters from the class via $p_i(\boldsymbol{\theta}|\mathcal{H}_i)$ where $\boldsymbol{\theta}$ is of dimension d_i (not to be confused with the VC dimension we discussed earlier).

The probability of data given the model is:

$$p(x^n|\mathcal{H}_i) = \int p(x^n|\boldsymbol{\theta})p_i(\boldsymbol{\theta}|\mathcal{H}_i)d\boldsymbol{\theta} \quad (3)$$

And we can also calculate:

$$p(\mathcal{H}_i|x^n) = \frac{p(x^n|\mathcal{H}_i)p(\mathcal{H}_i)}{p(x^n)} \quad (4)$$

To compare two different models we need to calculate:

$$\frac{p(\mathcal{H}_i|x^n)}{p(\mathcal{H}_j|x^n)} = \frac{p(x^n|\mathcal{H}_i)p(\mathcal{H}_i)}{p(x^n|\mathcal{H}_j)p(\mathcal{H}_j)} \quad (5)$$

Assuming the model priors are the same we are interested in $p(\mathcal{H}_i|x^n)$. This is generally a difficult task but we can approximate it for large enough n . This will lead to the so called Bayesian Information Criterion (BIC).

$$p(x^n|\boldsymbol{\theta})p_i(\boldsymbol{\theta}|\mathcal{H}_i) = e^{n\left(\frac{1}{n}\sum_i \log p(x_i|\boldsymbol{\theta}) + \frac{1}{n}\log p_i(\boldsymbol{\theta}|\mathcal{H}_i)\right)} = e^{nf_n(\boldsymbol{\theta})} \quad (6)$$

The expression in the exponent is dominated by its maximum. Lets write a Taylor expansion around the maximum:

$$f_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_i \log p(x_i|\boldsymbol{\theta}) + \frac{1}{n}\log p_i(\boldsymbol{\theta}|\mathcal{H}_i) \quad (7)$$

Now, for n large enough, the maximum of the above is attained at $\hat{\boldsymbol{\theta}}_{ML}^i$, the maximum likelihood parameter.

$$f_n(\boldsymbol{\theta}) \approx \frac{1}{n}\log p(x^n|\hat{\boldsymbol{\theta}}_{ML}^i) - \frac{1}{2}(\hat{\boldsymbol{\theta}}_{ML}^i - \boldsymbol{\theta})^T A(\hat{\boldsymbol{\theta}}_{ML}^i)(\hat{\boldsymbol{\theta}}_{ML}^i - \boldsymbol{\theta}) \quad (8)$$

where $A(\hat{\boldsymbol{\theta}}_{ML}^i)$ is the negative Hessian of $f_n(\boldsymbol{\theta})$ (and is PSD since this is a maximum).

$$\begin{aligned} \int e^{nf(\boldsymbol{\theta})}d\boldsymbol{\theta} &= e^{\log p(x^n|\hat{\boldsymbol{\theta}}_{ML}^i)} \int e^{-\frac{n}{2}(\hat{\boldsymbol{\theta}}_{ML}^i - \boldsymbol{\theta})^T A(\hat{\boldsymbol{\theta}}_{ML}^i)(\hat{\boldsymbol{\theta}}_{ML}^i - \boldsymbol{\theta})}d\boldsymbol{\theta} \\ &= e^{\log p(x^n|\hat{\boldsymbol{\theta}}_{ML}^i)} (2\pi)^{d_i/2} |nA(\hat{\boldsymbol{\theta}}_{ML}^i)|^{-0.5} \\ &= e^{\log p(x^n|\hat{\boldsymbol{\theta}}_{ML}^i)} (2\pi)^{d_i/2} n^{-d_i/2} |A(\hat{\boldsymbol{\theta}}_{ML}^i)|^{-0.5} = e^{\log p(x^n|\hat{\boldsymbol{\theta}}_{ML}^i)} \left(\frac{2\pi}{n}\right)^{-d/2} |A(\hat{\boldsymbol{\theta}}_{ML}^i)|^{-0.5} \end{aligned}$$

Taking the log we obtain:

$$\log p(x^n | \mathcal{H}_i) = \log p(x^n | \hat{\theta}_{ML}^i) - \frac{d_i}{2} \ln n \quad (9)$$

The above is commonly referred to as the Bayesian Information Criterion (BIC). This suggests the following scheme for choosing a good model: For each \mathcal{H}_i calculate its BIC score and choose the one with maximal BIC. As $n \rightarrow \infty$ the complexity term becomes negligible so that more complex model will be chosen.

Note that again we have a tradeoff between two conflicting terms. The likelihood increases as the model becomes more complex and $-\frac{d_i}{2} \ln n$ (the complexity term) decreases. Thus the above will be minimized for a model that trades off accuracy and complexity.

3 Cross-Validation

Here's what we would really like to do in order to measure generalization: Given training data S_n do:

- For each \mathcal{H}_i find the best hypothesis in the class h_i .
- Draw a large sample of size m : S_m .
- Measure the error of h_i on S_m . Denote it by e_i .
- Return \mathcal{H}_i that minimizes e_i .

The error e^i is a good approximation to the generalization error $e_p(h^i)$, and as $n \rightarrow \infty$ it will converge to this error. However, it is usually not practical to draw such sets. One thing we could do is to leave aside part of our training data (say γn) and evaluate the error on it. This is the so called *hold-out* set approach. The problem is that we are giving up on some of our training data.

Another, more effective approach is to divide S_n into K equal size parts (denote R_k the k^{th} part) and then for $k = 1 \dots K$ do:

- Train on $S_n \setminus B_k$. Measure the error of the result on B_k . Denote it by r_k .

Approximate the generalization error by $\frac{1}{K} \sum_{k=1}^K r_k$.

Do this for all models H_i and then choose the model that minimizes this error. You can then train this model on all the data (which is likely to give you an even better generalization error, which you will not have an approximation of). This process is called K fold cross-validation. When $K = n$ the procedure is called leave-one-out.

What does this process measure? It is not the generalization error of any one hypothesis but is rather closer to the generalization error that the learning algorithm will have when averaging over different datasets. Lets show this formally for leave-one-out.

Claim: Given a distribution $p(x, y)$ and a learning algorithm L that takes input S_n and output a hypothesis $h_{S_n}^L$ the following holds:

$$E_{S_n \sim p} [e^{LOO}(S_n)] = E_{S_{n-1} \sim p} [e_p(h_{S_{n-1}}^L)] \quad (10)$$

Proof:

$$\begin{aligned} E_{S_n \sim p} [e^{LOO}(S_n)] &= \frac{1}{n} E_{S_n \sim p} \left[\sum_{k=1}^n I(h_{S_n^{-k}}^L(x_k) \neq y_k) \right] \\ &= \frac{1}{n} \sum_{k=1}^n E_{S_n \sim p} [I(h_{S_n^{-k}}^L(x_k) \neq y_k)] \\ &= \frac{1}{n} \sum_{k=1}^n E_{S_n^{-k} \sim p} E_{(x_k, y_k) \sim p} [I(h_{S_n^{-k}}^L(x_k) \neq y_k)] \\ &= \frac{1}{n} \sum_{k=1}^n E_{S_n^{-k} \sim p} E_{(x_k, y_k) \sim p} [I(h_{S_n^{-k}}^L(x_k) \neq y_k)] \\ &= \frac{1}{n} \sum_{k=1}^n E_{S_n^{-k} \sim p} e_p(h_{S_n^{-k}}^L) = E_{S_{n-1} \sim p} [e_p(h_{S_{n-1}}^L)] \end{aligned}$$

So we obtain a nearly unbiased estimator of the expected error of the classifier. Similarly if we consider K fold, we will estimate the error when training on $n - n/K$ examples.

One problem with leave-one-out is that it is computationally expensive. There are however cases (e.g., in regression and density estimation) where it can be computed in roughly the same time it takes to train over S_n .

4 Minimum Description Length

This is a rather different approach to the problem of model selection. We will talk about it after we've discussed compression.

5 Classification in practice

Here are several approaches to constructing classifiers:

- Generative models: Assume a model $p(x, y|\theta)$ and maximize the likelihood of $p(x^n, y^n|\theta)$. Example: $p(x, y|\theta) = c_y \mathcal{N}(\mu_y, \sigma^2)$

- Discriminative models: Assume a model $p(y|x, \theta)$ and maximize $\sum_i \log p(y_i|x_i; \theta)$. Example. Logistic regression:

$$p(y|x, \theta) = \frac{e^{y\theta \cdot x}}{e^{\theta \cdot x} + e^{-\theta \cdot x}} \quad (11)$$

Note that this results in a linear classification rule.

- Nearest neighbors. We would like to know $p(y|x)$. If this distribution doesn't change much in a neighborhood of x we can try:

$$p(y|x) \approx \frac{1}{|N(x)|} \sum_{i: \mathbf{x}_i \in N(x)} I(y = y_i) \quad (12)$$

- Support vector machines. Find a hyperplane with maximum margin.
- Boosting, decision trees, and others.