CS5339 Lecture Notes #10: Model Selection

Jonathan Scarlett

March 31, 2021

Useful references:

- MIT lecture notes,[1] lectures 9 and 10
- Supplementary notes lec10a.pdf and lec11a.pdf
- Sections 1.3 and 3.4 of Bishop's "Pattern Recognition and Machine Learning" book
- Chapter 11 of "Understanding Machine Learning" book

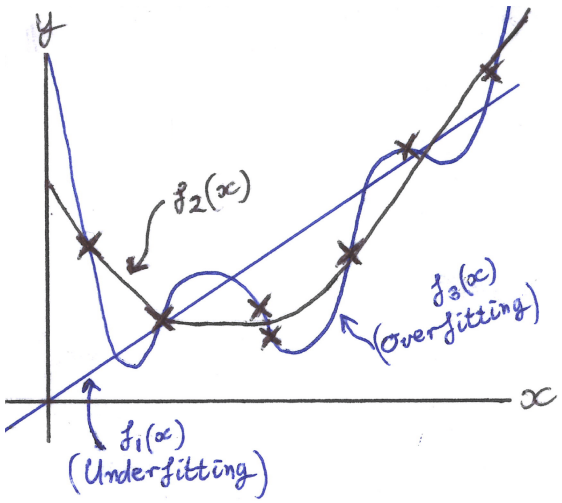
1 Model Selection

- The term *(mathematical) model* has broad meaning, but in this lecture will be taken to mean a class of prediction functions (like $\mathcal{F}$ from the previous lecture):
  - e.g., (linear classification model) Set of all linear classifiers on $\mathbb{R}^d$
  - e.g., (polynomial classification model) Set of all polynomial classifiers on $\mathbb{R}^d$ with degree $\leq p$
  - e.g., set of all classifiers formed by combining votes from 100 decision stumps
- We have looked at methods for constructing classifiers ($y_t \in \{-1, 1\}$) and prediction rules ($y_t \in \mathbb{R}$) for a given "model", e.g.,
  - A linear decision rule based on $\boldsymbol{\theta}^T \mathbf{x} + \theta_0$ (where we learn the parameters $\boldsymbol{\theta}$ and θ_0)
  - A decision rule based on $\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}) + \theta_0$ for a *given* choice of features $\boldsymbol{\phi}(\mathbf{x})$.
  - Kernel regression using the kernel $k(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^p$ for a *given* degree p.

  For concreteness, we will mostly use classification terminology (e.g., talk about classification rules produced by each model), but everything below is equally valid for both classification and regression unless stated otherwise.
- A fundamental question: How do we choose the model itself? (e.g., the set of features, the degree in the polynomial, or the kernel more generally)

[1] http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/lecture-notes/

- For a given data set $\mathcal{D}$, we can always make the training error smaller by making the model more complex (e.g., adding more features, increasing p). But is that a good idea?
- Intuition: Simpler classification (or regression) rules should generalize better to unseen data:
  - Related to the principle of *Ockam's razor*: The best answer is the simplest one consistent with what we can observe.
- But we need to avoid both *over-fitting* (overly complex model) and *under-fitting* (model is not rich enough). A related issue is that small training error does not imply small test error:



- Last lecture we explored these issues for a *single* model (function class). Here we look at the problem of *choosing between one of multiple models*.

2 Cross-Validation

- One of the most widespread practical methods for choosing a model is *cross-validation*:
  - Split the data set $\mathcal{D}$ into two sets $\mathcal{D}_{\text{train}}$ (training set) and $\mathcal{D}_{\text{val}}$ (validation set).
  - For each candidate model (e.g., choice of polynomial degree $p = 1, \ldots, p_{\max}$), form a classifier using the data $\mathcal{D}_{\text{train}}$ (e.g., using SVM).
  - Apply each classification rule to the inputs in $\mathcal{D}_{\text{val}}$, and let the final classifier be the one that classified the highest fraction of these correctly.
- In fact, this does not need to be viewed as only a "model selection" technique per se:
  - It could be used for *parameter selection* (e.g., choosing λ in ridge regression)
  - It could be used to pick one out of multiple classifiers we have learned within a single model (e.g., $f_{\text{Perceptron}}(\mathbf{x})$, $f_{\text{SVM}}(\mathbf{x})$, $f_{\text{Logistic}}(\mathbf{x})$, etc., all of which produce linear classifiers)
- Since $\mathcal{D}_{\text{val}}$ plays no role in the training, this split can be a bit wasteful. An alternative is *k-fold cross-validation*, in which all data is used for training:

- Split $\mathcal{D}$ into k equal parts $\mathcal{D}_1, \ldots, \mathcal{D}_k$;
- Perform training of the combined data set containing all parts except one, and for each classifier compute the average (validation) error on the the remaining one
- Repeat this for all k possible choices of "the remaining one"
- Choose the classifier that gave the smallest validation error on average

- In the case $k = n$ (n being the total data size), this is called *leave-one-out cross validation.*
- Sometimes a rule-of-thumb of $k = 5$ or $k = 10$ is suggested, but in general there is no "best choice".
- Beyond training and validation sets, often an additional *test set* $\mathcal{D}_{\text{test}}$ is used to estimate the performance of the final classifier. $\mathcal{D}_{\text{test}}$ is not used during training or validation.
  - Basic principle: If we used some data set, *even in a subtle way*, to produce our final classifier, then the empirical error on that data set might be biased.
  - An example of such a "subtle way" is as follows: What if we don't like the test error we obtained, so we go back and try something different? What if we repeat such a process 1000 times?[2]
  - Clearly, the classifier produced by cross-validation depends on both $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$, so both were used to produce the final (cross-validated) classifier.

3 Statistical Learning Viewpoint and Structural Risk Minimization

- Recall that statistical learning theory models learning in terms of an unknown data distribution $P_{\mathbf{X}Y}$, a function class $\mathcal{F}$ (e.g., linear classifiers), and a loss function $\ell(y, f(\mathbf{x}))$ (e.g., 0-1 loss).
- In this context, the model selection problem is to *choose a good function class* from a number of candidates $\mathcal{F}_1, \ldots, \mathcal{F}_M$.
  - Let $\mathcal{F}_{\text{all}} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_M$ contain all hypotheses under consideration.
  - Let f_{opt} be the best function (i.e., smallest average loss under $P_{\mathbf{X}Y}$) in $\mathcal{F}_{\text{all}}$, and let $f_*^{(m)}$ be the best in $\mathcal{F}_m$ alone (i.e., f^* from the previous lecture).
  - Let $f_{\text{erm}}^{(m)}$ be the empirical risk minimization rule (from the previous lecture) restricted to $\mathcal{F}_m$, i.e., the function from $\mathcal{F}_m$ with the smallest training error.
- **A fundamental trade-off.**
  - "Complex" classes $\mathcal{F}_m$ tend to have lower values of $R(f_*^{(m)})$ (the best classifier in the class is better)
  - However, as we saw in the previous lecture, they also have a higher chance of overfitting and generalizing poorly. Hence, actually approaching the performance $f_*^{(m)}$ might not be possible if our data size is limited.
- Useful facts from the previous lecture:
  - Recall that $R_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i))$ denotes the empirical risk.

[2] Having said this, in practice it seems that this idea of re-using test data typically doesn't cause too many problems.

– The analysis of the previous lecture shows that

$$R(f_{\text{erm}}^{(m)}) - R_n(f_{\text{erm}}^{(m)}) \leq \sqrt{\frac{1}{2n} \log \frac{2|\mathcal{F}_m|}{\delta_m}} \tag{1}$$

with probability at least $1 - \delta_m$ (or similarly for infinite classes via the VC dimension).

– When this high-probability event holds, $R_n(f_{\text{erm}}^{(m)}) + \sqrt{\frac{1}{2n} \log \frac{2|\mathcal{F}_m|}{\delta_m}}$ is an upper bound on $R(f_{\text{erm}}^{(m)})$ that we can compute (without knowing $P_{\mathbf{X}Y}$). We can use this idea to perform model selection – choose the function class indexed by m with the smallest such upper bound.

- **Structural risk minimization.**

  – Choose $\{\delta_m\}_{m=1}^M$ such that $\sum_{m=1}^M \delta_m \leq \delta$, and use the union bound to deduce that (1) holds *for all* m with probability at least $1 - \delta$.

    * The simplest such choice of $\{\delta_m\}_{m=1}^M$ is $\delta_m = \frac{1}{M}$.

  – Choose the m that gives the smallest upper bound on $R(f_{\text{erm}}^{(m)})$:

$$\hat{m} = \operatorname*{arg\,min}_{m=1,\ldots,M} \left\{ R_n(f_{\text{erm}}^{(m)}) + \sqrt{\frac{1}{2n} \log \frac{2|\mathcal{F}_m|}{\delta_m}} \right\}.$$

  – **Intuition.** The first term represents our estimate of how good the ERM classifier of the m-th class is; the second term penalizes more complex classes for which we have higher chance of overfitting.

  – We can use the same idea for infinite classes via the VC dimension.

4 Bayesian Viewpoint and the BIC Criterion

For the sake of consistency with typical literature, from now on the models will be written as $\mathcal{M}_1, \ldots, \mathcal{M}_M$ instead of $\mathcal{F}_1, \ldots, \mathcal{F}_M$. A concrete example is that the model $\mathcal{M}_m$ is a polynomial regression model with degree m, so model selection amounts to choosing the degree (up to a maximum considered value M).

Bayesian modeling.

- Bayesian models place probability distributions on *anything* that is unknown (e.g., parameters $\boldsymbol{\theta}$ of the model, or even the model itself).

- There are different levels of Bayesian modeling:

  – For a fixed model $\mathcal{M}$, its parameters $\boldsymbol{\theta}$ may be treated as random, with a prior distribution (e.g., Gaussian $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_0, \sigma_0^2 \mathbf{I})$). This is known as *Bayesian modeling*.

  – The model itself may be also treated as random with some prior (e.g., $\Pr[\mathcal{M} = \mathcal{M}_m] \propto (1 - \epsilon)^m$ to favor smaller values of m). This is sometimes called *fully Bayesian modeling*.

- Let's first look at the case of random $\boldsymbol{\theta}$ for a given model $\mathcal{M}$:

– Since $\boldsymbol{\theta}$ is now treated as random, we can seek the *posterior distribution* via Bayes' rule:

$$p(\boldsymbol{\theta}|\mathcal{D},\mathcal{M}) = \frac{p(\mathcal{D}|\boldsymbol{\theta},\mathcal{M})p(\boldsymbol{\theta}|\mathcal{M})}{p(\mathcal{D}|\mathcal{M})}, \quad \text{or} \quad \text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}.$$

Note that the evidence can be expanded as $p(\mathcal{D}|\mathcal{M}) = \int p(\mathcal{D}|\boldsymbol{\theta},\mathcal{M})p(\boldsymbol{\theta}|\mathcal{M})d\boldsymbol{\theta}$.

– Choosing $\boldsymbol{\theta}$ to maximize $p(\boldsymbol{\theta}|\mathcal{D},\mathcal{M})$ gives the *maximum a posteriori (MAP)* estimate. But there is no need to commit to a single estimate! Instead, we can predict by weighting all possible $\boldsymbol{\theta}$ according to the posterior:

$$p(y|\mathbf{x},\mathcal{D},\mathcal{M}) = \int p(y|\mathbf{x},\boldsymbol{\theta},\mathcal{M})p(\boldsymbol{\theta}|\mathcal{D},\mathcal{M})d\boldsymbol{\theta}.$$

- Now, how do we choose whether $\mathcal{M}_i$ or $\mathcal{M}_j$ is better?

– If we don't have any prior knowledge of the model (e.g., the prior $p(\mathcal{M})$ is uniform), then it is natural to consider

$$\mathcal{M}_i \text{ favored over } \mathcal{M}_j \iff p(\mathcal{D}|\mathcal{M}_i) > p(\mathcal{D}|\mathcal{M}_j). \tag{2}$$

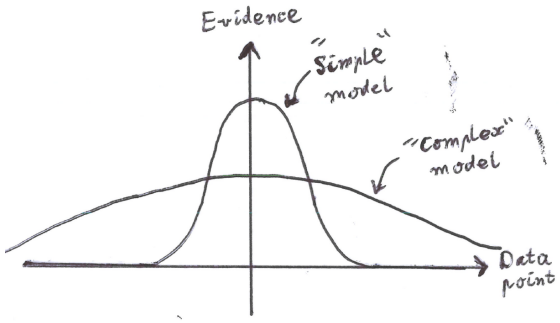
In other words, choose the model with the *highest evidence*.

– This is a powerful rule if it can be implemented, but unfortunately computing the evidence $p(\mathcal{D}|\mathcal{M})$ is usually not feasible.

– There are some exceptions (e.g., Gaussian settings), and for general scenarios there are powerful approximation methods (e.g., Monte Carlo, variational methods), but we will not cover these.

- Remark. The rule (2) (implicitly) favors both simpler models and models that fit the data better – we don't need to design a prior $p(\mathcal{M})$ to explicitly favor simpler models!

– Intuition: The more parameters/complexity, the less probability density in simple regions:



– See, e.g., Ch. 28 of MacKay's book "Information Theory, Inference, and Learning Algorithms"

Model selection via information criteria.

- The *Bayesian information criterion (BIC)* selection rule can be motivated as approximating (2) in the large-sample limit (under some technical assumptions). It is given by

$$\hat{m} = \arg\max_m \text{BIC}_m,$$

where

$$\mathrm{BIC}_m = 2 \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}_m, \mathcal{M}_m) - d_m \log n,$$

and where $\hat{\boldsymbol{\theta}}_m$ is the maximum-likelihood estimate of the parameter vector $\boldsymbol{\theta}_m$ for model $\mathcal{M}_m$, and d_m is the number of parameters in $\boldsymbol{\theta}_m$.

- Again, we have an "achieve good data fit" term (negative log-likelihood), and a "penalize complex models" term ($d_n \log n$).
- See lec10a.pdf for optional extra reading with a derivation. The derivation is also stepped through in one of the advanced tutorial questions.
- This is much easier to compute, as there are no integrals or posteriors involved.

- There is also a similar-looking selection rule that looks very similar, but comes from a different motivation (not covered here): The *Akaine Information Criterion* (AIC)

$$\mathrm{AIC}_m = 2 \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}_m, \mathcal{M}_m) - 2d_m,$$

where the factors of two are included to match the form of BIC.

- General guidelines:
  - BIC usually works better if the true model is among $\mathcal{M}_1, \ldots, \mathcal{M}_M$.
  - AIC is usually preferable otherwise ("all models are wrong").
  - Both are mainly suited to n much larger than d (e.g., $d \ll \sqrt{n}$ for typical proofs to work).

5 Feature Selection

The feature selection problem.

- Suppose we have a huge number d of features; to avoid carrying $\boldsymbol{\phi}(\mathbf{x})$ notation around, let's just work with $\mathbf{x} = [x_1, \ldots, x_d]^T$, each entry of which is a feature.
- To avoid overfitting and dealing with irrelevant or redundant features, we would like to select a small number $s \ll d$ of them and use those for classification. How do we select these features?
  - This is the topic of *feature selection* (also known as *subset selection*).
  - We will only give a flavor of 1–2 techniques (out of many more) for this task.
- If we had a reasonably small number of candidate sets $S_1, S_2, \ldots$ of features, we could pick the best one using validation (i.e., train a model for each candidate set, and then choose the one that does best on the validation data), or one of the above model selection methods.
- However, a different approach is needed if we are considering *all possible* subsets – $\binom{d}{s}$ is a huge number even for moderate values of d and s (And typically we don't even know *a priori* which s to pick)

Feature selection via greedy methods.

- Greedy methods work by adding one feature at a time:
  - Initialize the set $\widehat{S}$ of features to be empty
  - Search over the d features, pick the one that maximizes some score, and add it to $\widehat{S}$
  - Search over the remaining $d-1$ features, pick the one that maximizes the score *when used alongside the first one picked*, and add it to $\widehat{S}$
  - etc.

  One may stop after a pre-specified number s of features, or take note of the entire order and then choose $s \in \{1, \ldots, n\}$ by checking which of the resulting n classifiers works best on validation data.
- The choice of score function is important – examples include (i) a quantity called mutual information that measures "how much information" one random variable reveals about another; (ii) the reduction in residual error after performing least squares in regression.
- It is easy to imagine that greed is not always good – e.g., what if two (or more) features are very useful *when used together*, but not very useful on their own?

Feature selection via the Lasso.

- For convenience, we focus here on feature selection in the basic linear regression model (without offset) in which the data point $(\mathbf{x}_t, y_t)$ for $t = 1, \ldots, n$ are modeled as follows:

$$y_t = \boldsymbol{\theta}^T \mathbf{x}_t + z_t$$

  for some unknown parameter $\boldsymbol{\theta} \in \mathbb{R}^d$, where z_t is noise.
  - In the case that only $s \ll d$ features are relevant, this amounts to $\boldsymbol{\theta}$ having only s significant entries (e.g., the rest are zero, or close to zero). Such a $\boldsymbol{\theta}$ is called (approximately) *s-sparse*.
  - The ideas below can be applied to binary classification (e.g., in the logistic regression setting), non-linear regression, and more.
  - An offset θ_0 can also be included; we omit it for simplicity.
- The *Lasso estimator* performs ℓ_1-regularized least squares:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{n} (y_t - \boldsymbol{\theta}^T \mathbf{x}_t)^2 + \lambda \Big(\sum_{j=1}^{d} |\theta_j| \Big)$$

  for some $\lambda \geq 0$, or in vector notation (see the Linear Regression lecture):

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2 + \lambda \|\boldsymbol{\theta}\|_1,$$

  where the *ℓ_1-norm* $\|\cdot\|_1$ is defined to be the sum of the absolute values of a vector. (In contrast, the ℓ_2 norm is $\|\boldsymbol{\theta}\|_2 = \sqrt{\sum_{j=1}^{d} \theta_j^2}$)
  - Looks almost the same as ridge regression, but with $\|\boldsymbol{\theta}\|_1 = \sum_{j=1}^{d} |\theta_j|$ in place of $\|\boldsymbol{\theta}\|_2^2 = \sum_{j=1}^{d} \theta_j^2$. (But looks can be deceiving!)

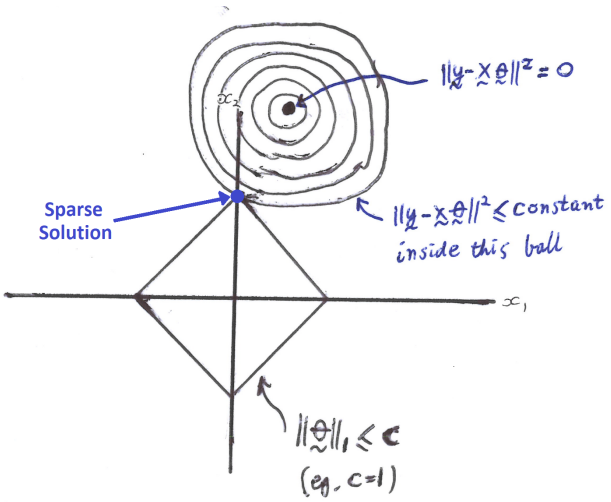
- Once the Lasso solution $\hat{\boldsymbol{\theta}}$ is found, the estimated *set of relevant variables* is chosen as

$$\hat{S} = \{j : \hat{\theta}_j \neq 0\}. \tag{3}$$

- Two examples of how Lasso might be used in practice:

  1. Find a good choice of the regularization parameter λ via cross-validation, and apply (3) directly.
  2. Alternatively, sweep from a large λ (no variables selected) down to a small λ (all variables selected) and take note of the order in which the variables are added. Choose the s variables that were added first, where s is selected separately similarly to the above discussion on greedy methods.

- **Intuition 1.** The Lasso favors sparse solutions because the ℓ_1-ball is "pointy":



- **Intuition 2.**

  – "Ideally" we might minimize $\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2 + \lambda\|\boldsymbol{\theta}\|_0$, where $\|\boldsymbol{\theta}\|_0$ is the number of non-zeros in $\boldsymbol{\theta}$. That is, we seek to balance between goodness of fit $\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2$ and a complexity term $\|\boldsymbol{\theta}\|_0$.

  – However, this is a hard combinatorial optimization problem. Replacing $\|\boldsymbol{\theta}\|_0$ by $\|\boldsymbol{\theta}\|_1$ is a *convex relaxation* that makes the computation much easier.

- Various types of theoretical guarantees are known for Lasso (e.g., conditions on $\mathbf{X}$ under which the correct support of $\boldsymbol{\theta}$ is identified) but these are beyond the scope of this course.