Predictive model evaluation

Introduction to statistical learning and decision theory

Erik Štrumbelj

1 Introduction

The purpose of these notes is to introduce the reader to some of the fundamentals of an important part of the applied predictive modeling process – *model evaluation.* In fact, we begin this introduction by arguing that this is the most important part of modeling. We base our argument on two observations. First, there is no single best model or modeling paradigm. Therefore, in most applications, we will be faced with model selection decisions, choosing between different settings of a model or entirely different modeling paradigms. And second, a poorly designed model evaluation will lead to incorrect decisions, regardless of how good we are at writing models or at any other stage of the modeling process. On the other hand, a well designed model evaluation will illuminate any and all performance-related mistakes we might make in any of the models we use.

Model evaluation (comparison, selection) is strongly connected with learning. However, for now, we will keep the two separate and we will break down model evaluation into two components: (1) choosing what to measure[1] and (2) choosing how to measure it. And from these two components follow the two groups of model evaluation mistakes:

1. **Measuring the wrong thing.** That is, measuring a quantity that does not correspond to our practical objective.
2. **Measuring in a biased or inaccurate way.** That is, measuring potentially even the correct quantity, but doing so in a way that has a lot of error.

Before we move on to a more rigorous treatment of these concepts, let's illustrate them on a real-world example. And let's choose an example that most of us have been (and will be) in one way or another personally involved in – *evaluating student performance.* At university we assign grades to student performance in a course. These grades are then used for progression and graduation decisions and even for student comparison in processes such as awards for scholarly performance and stipends.

In the above example students play the role of learning algorithms[2] – their goal is to learn the course contents as well as possible or at least well enough to pass the exam. The role of model evaluation, on the other hand, is played by the teacher. And we'll assume that the goal of the teacher is to assign grades that correspond to the students' level of performance in the course.

As with any model evaluation process, the teacher must decide on what to measure and how to measure it. Until technology evolves to a stage where we can just measure the contents of a student's brain, we are somewhat limited in our options. In most cases teachers opt to query the student with tasks or questions related to the course contents, either via homework, projects, written or oral examinations. This sounds like a reasonable thing to do – good performance in related tasks and correct answers to substantive questions should be a good

[1]Unless otherwise noted, we will be using the term performance measure or just measure to refer to quantities we use in describing model performance and not in the measure-theoretic sense.

[2]Arguably, the teachers also play a part, as their objective is more often than not to help the students on their path to knowledge.

proxy for the student's skills and knowledge. Of course, we could also make a mistake in our choice of what to measure. For example, we could decide that a student's grade in a machine learning course will be based on how long they can hold their breath. We can all recognize that this is a silly choice and a choice we would never make, but we will later see how similarly silly (but somewhat less obviously so) choices are commonplace in applied predictive modeling.

Now that we have, at least conceptually, decided *what* to measure, we must decide *how* we will measure it. As is the case with most empirical work, our measurements will have measurement error. And this error will have two parts – *bias* (systematically deviating from what we are trying to measure) and *variance* (non-systematic deviation from what we are trying to measure or lack of accuracy)[3].

Let's for a moment assume that we will use a written exam with 5 problems chosen randomly from the course contents. And that the only remaining choice to make is who will grade the exam. The first option we will explore will be to use a trained circus monkey that will more-or-less randomly write grades on exams. While monkeys might seem like an obviously silly choice, they do have their advantages. In particular, their lack of comprehension of modern machine learning (not to mention reading skills) makes them more-or-less unbiased. Of course, grades assigned by a monkey would still be very poor – an empty exam can earn a top grade or an exceptional student can fail – but at least in expectation the monkey is fair and all the error will come from variance.

The second option we will explore will be a commonly used and arguably better one than using a monkey – using a teaching assistant. A teaching assistant is knowledgeable in machine learning and will adjust the grade to each individual students' solutions thus reducing the variance. However, with a reduction in variance typically comes a systematic bias. The teaching assistant might be on average more lenient or more strict. Of course, a relatively small bias is still a great trade-off for the reduced variance and an overall student evaluation performance that is without a doubt better than a monkey's. The third and final option is of course the course teacher himself. While some, students in particular, might argue that teachers can also be biased, inaccurate, or even inconsistent, that is not the case. By definition, *a student's grade is what the teacher says it is* and is therefore perfect, without bias or variance.[4]

The above only scratches the surface of the measurement problems we face in education. For example, having more questions on an exam will reduce variance, but after some point the student's tiredness might overwhelm the final grade. Oral examinations typically have lower variance but the teacher's bias might be more pronounced, while with written examinations it is typically the opposite. Luckily, we do not have to concern ourselves with such things in the evaluation of predictive models.

There is, however, a particular form of bias that is very common in predictive modeling – bias due to model *overfitting*. We should already be familiar with overfitting, at least on a conceptual level. Later we will discuss at length how to design a model evaluation procedure that will allow us to detect overfitting[5], but here we will just provide an illustrative example in the context of student performance evaluation. Students often prepare for exams by studying past exams, sometimes to the extreme of memorizing all past exam problems and solutions.

[3]This bias-variance decomposition should already be a familiar term for most of us.

[4]This is a joke... Or is it?

[5]It is not the purpose of model evaluation to prevent overfitting, merely to detect it. Dealing with overfitting is a modeling problem and different techniques can be used, such as bagging, feature selection, and regularization.

Unless the pool of problems is so large that it covers all there is to learn in a course, this will be a clear example of overfitting. This by itself is not a problem, but if it is not taken into account, for example, by always introducing new problems or non-trivial variants of existing problems, students might pass the exam without a proper level of knowledge of the subject matter. An extreme example of this problem would be to give the students an exam that was also used as a practice exam for preparation for the exam. Obviously, the students' grades would be biased – on average higher than their level of knowledge. Analogously, we should never evaluate our predictive models on training data. At least not without accounting for bias due to the learning algorithms capacity to learn.

1.1 Evaluating predictive models

Anyone with some practical experience in machine learning will be aware that there exists a plethora of different measures of predictive performance. Accuracy, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), absolute error, quadratic score, Brier score, spherical score, logarithmic score, AIC, BIC, DIC, AUC, F1, sensitivity, specificity. And the list goes on.

On the other hand, most of the models are fit with underlying optimization procedures that are based on maximizing the likelihood, minimizing the MSE, maximizing mutual information, etc. which, as we will show, are all related to maximizing the logarithmic score. So why do we have so many different measures? Why do people fit logistic regression using maximum likelihood and then choose the best model based on its accuracy or AUC? And how is this different from teaching students machine learning and then grading them based on how long they can hold their breath? The short answer is that this area of applied predictive modeling is a bit of a conundrum and full of bad practices. The long answer will take up the better part of these notes.

In the remainder of these notes we will cover *loss functions* (what to measure) and the *estimating generalization error* (how to measure). However, to fully appreciate the practical application of model evaluation, we will do so through the basics of statistical learning and statistical decision theory.

2 Empirical risk minimization (ERM)

In this section we will focus on one particular principled approach to statistical learning - *empirical risk minimization.* The key idea is not unlike the predictive modeling practice that we might already be familiar with: our goal is to select the model that has the lowest expected error and, because the true distribution of data is rarely known, we instead estimate it using available (empirical) data. However, a more principled approach will allow us to study the properties and limits of learning.

We will use the following notation throughout: Let $\mathcal{X}$ and $\mathcal{Y}$ be our dependent variable (input) and independent variable (output) space, respectively. Typically, we have $\mathcal{X} = \mathbb{R}^k$ and $\mathcal{Y} \in \mathbb{R}$ (regression) or some subset of $\mathbb{N}$ (classification, ordinal regression). Let $p_{X,Y}$ be the joint distribution of a pair of variables (X, Y) from $\mathcal{X} \times \mathcal{Y}$ and $D_n = \{(x_i, y_i)\}_{i=1}^n$ our dataset (assumed to be i.i.d. samples from $p_{X,Y}$; we will restrict ourselves to the i.i.d. setting). We will use $h : \mathcal{X} \to \mathcal{Y}$ to denote a model (function). A *learning algorithm* $\mathcal{A}$ given a dataset produces a model $\mathcal{A}(D_n) = h_n$. Let $\mathcal{H}$ to denote the set of all possible models for a particular learning algorithm (that is, the *hypothesis space*).

A *loss function* is a mapping $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$.[6] For example, two commonly used loss functions are $\ell(y, \hat{y}) = I_{y \neq \hat{y}}$ (0-1 loss) and $\ell(y, \hat{y}) = (y - \hat{y})^2$ (quadratic loss). Note that choosing the appropriate loss function(s) for our problem is not trivial. The theoretical and practical considerations will be discussed throughout these notes.

The quantity central to empirical risk minimization is the model's *risk* (or *generalization error*) - the expected loss over the distribution of the data

$$R(h) = \mathbb{E}_{X,Y}[\ell(Y, h(X))].$$

Let h_{opt} be the model that attains the minimum possible risk $R_{\text{opt}} = \inf_h R(h)$ over all possible models.[7] Similarly, let h^* be the model that attains the minimum possible risk $R^* = \arg\min_{h \in \mathcal{H}} R(h)$, that is, over all possible models in a learning algorithm's hypothesis space $\mathcal{H}$.

By definition, we have $R^* \geq R_{\text{min}}$. However, in order for learning to be feasible, the hypothesis space $\mathcal{H}$ of a learning algorithm is much less rich than all possible models, thus the theoretical minimum is rarely obtained.

Because the distribution of the data $p_{X,Y}$ is typically unknown, we have to utilize the empirical data D_n. In the most basic form of empirical risk minimization we do so by replacing the distribution of the data with the empirical distribution D_n, obtaining the *empirical risk* of a model h:

$$R_n(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(x_i)).$$

Learning a model then becomes a minimization problem:

$$h_n = \arg\min_{h \in \mathcal{H}} R_n(h).$$

We use h_n to distinguish it from the optimal choice for the learning algorithm h^*. Note that h_n depends on D_n (it is in essence a functional of the empirical CDF) and can be viewed as a random variable until observed. Also note that minimizing empirical risk typically results in overfitting. Later we will discuss other approaches to selecting h_n or estimating true risk - structural risk minimization, train/test evaluation, and cross-validation.

2.1 Examples of ERM estimators

Quadratic loss

Suppose $y \in \mathbb{R}$ and that our loss function is the quadratic loss $\ell(y, \hat{y}) = (y - \hat{y})^2$. What is the ERM estimator $\hat{y}$? That is, which estimator minimizes the empirical risk?

In this case the empirical risk is

[6] The notation ℓ is often used for the log-likelihood. The (negative)log-likelihood is also a loss function, but we use ℓ here in the more general sense of any loss function.

[7] For our purposes we do not have to precisely define what we mean by *all possible models*. Typically, it would be the set of all measurable functions.

$$R_n(\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2$$

and as we might already know from statistics, the sum of squared residuals is minimized at the sample average $\hat{\theta} = \overline{y}$. That is, the sample average is the ERM estimator under the quadratic loss.

Absolute loss

Now suppose $y \in \mathbb{R}$ and that our loss function is the absolute loss $\ell(y, \hat{y}) = |y - \hat{y}|$. What is the ERM estimator $\hat{y}$? That is, which estimator minimizes the empirical risk?

In this case the empirical risk is

$$R_n(\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}|.$$

Using $\frac{d}{dx}|x| = \text{sign}(x)$ we can observe that the derivative of the empirical risk will be a sum of $\text{sign}(y_i - \hat{y})$ terms. This will be 0 if the number of strictly positive and strictly negative terms will be exactly the same. That is, the same number of y_i must be greater than $\hat{y}$ as there are y_i less than $\hat{y}$. This implies that ERM estimator under the absolute loss is the sample median! It also shows that the ERM estimator does not have to be unique.[8].

Maximum likelihood estimation (MLE) as ERM

Given a parametric model (likelihood) $p(y_i|\theta)$, parametrized with θ, we define the maximum likelihood estimator (in the i.i.d. case) as:

$$\hat{\theta} = \arg\max_{\theta\in\Theta} \prod_{i=1}^{n} p(y_i|\theta).$$

or, equivalently,

$$= \arg\min_{\theta\in\Theta} \sum_{i=1}^{n} \left(-\log p(y_i|\theta)\right).$$

The negative logarithm of the predicted density at the observed value is also known as the log-loss. So, MLE is a special case of ERM, where $\mathcal{Y}$ is the space of densities (or distributions) and we use the log-loss under the assumed parametric model as our loss function.

Let's illustrate what MLE. Suppose our y is categorical and that we use the log loss as our loss function. The empirical risk is

$$R_n(\hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}\log\hat{y}(y_i),$$

[8]The sample median is not always unique

where $\hat{y}(i)$ is the probability that our estimator assigns to category i. Let n_i be the number of times the $i-$th category appears in our data. Now we rewrite the risk in terms of categories:

$$R_n(\hat{y}) = -\sum_{i=1}^{k} \frac{n_i}{n} \log \hat{y}(i) = -\frac{n_1}{n} \log \left(1 - \sum_{i=2}^{k} \hat{y}(i)\right) - \sum_{i=2}^{k} \frac{n_i}{n} \log \hat{y}(i).$$

We deliberately expressed the probability of one category with the probabilities of the other categories to explicitly include the constraint that the probabilities need to sum up to 1 (effectively, we have only $k-1$ parameters to optimize). Now we take the derivatives:

$$\frac{\partial}{\partial \hat{y}(i)} R_n(\hat{y}) = \frac{f_1}{1 - \sum_{i=2}^{k} \hat{y}(i)} - \frac{f_i}{\hat{y}(i)} = \frac{f_1}{\hat{y}(1)} - \frac{f_i}{\hat{y}(i)}.$$

These will be zero if the ratio between the relative frequency in the data $f_i = \frac{n_i}{n}$ and the assigned probability $\hat{y}(i)$ will be the same for all categories. And because both relative frequencies and probabilities need to sum up to 1, we can conclude that they must be the same. So, the estimator that minimizes log loss is the one that matches the empirical distribution of the data!

Suppose that the true probability distribution is q and f is the empirical distribution from before. The excess risk of the ERM estimator under the log loss, which we've shown is f, is then

$$R(f) - R(q) = -\sum_{i=1}^{k} q_i \log f_i + \sum_{i=1}^{k} q_i \log q_i = \sum_{i=1}^{k} q_i \log \frac{q_i}{f_i}.$$

The last term is also known as the Kullback–Leibler (or KL) divergence from f to q.

0-1 loss

Suppose the same categorical setting as above in the MLE example, but now our that we use the 0-1 loss (or accuracy). Note that 0-1 loss depends only on the true and predicted class, so we do not have to work with predicted probabilities. The risk is

$$R_n(\hat{y}) = \frac{1}{n} \sum_{i=1}^{n} I_{y_i \neq \hat{y}} = 1 - \frac{1}{n} \sum_{i=1}^{n} I_{y_i = \hat{y}} = 1 - \sum_{i=1}^{k} \frac{n_i}{n} I_{\hat{y}=i},$$

where n_i is again the number of times the $i-$th category appears in our data. It is immediately clear that we will minimize the risk by setting the estimator to the category with the largest relative frequency (the mode of the empirical distribution)! This also shows how multiple (in fact infinitely many) different sets of probabilities can obtain the same 0-1 loss as long as they have the same mode. Also, if two or more categories are tied for the mode, predicting any of them will be the ERM estimator.

2.2 Approximation-estimation decomposition

Now we are ready to introduce *excess risk* - the difference between the chosen model's risk and the optimal risk:

$$R(h_n) - R_{\text{opt}} = \overbrace{(R^* - R_{\text{opt}})}^{\text{approx. error}} + \overbrace{(R(h_n) - R^*)}^{\text{est. error}} .$$

The decomposition of excess risk on the right-hand side provides a useful view on model performance. The first term (approximation error) is the difference between the best attainable risk for this learning algorithm (by a model from the learning algorithm's hypothesis space) and the optimal risk. The second term (estimation error) is the difference between the chosen model's risk and the best attainable risk for this learning algorithm.

The above can be viewed as a more general bias-variance decomposition. A very expressive algorithm such as a large neural network will have low approximation error (h^* closely fit h_{opt}) on the other hand, due to this expressiveness, we might be prone to overfitting (h_n is expected to vary a lot from h^*), which would result in a higher estimation error. For ordinary least-squares linear regression the opposite is typically true - it has high approximation error because $\mathcal{H}$ is restricted to linear models, on the other hand, it is very robust - even for small n (but assuming still relatively large relative to the number of input variables) the estimated linear model h_n will be close to the optimal attainable h^*.

Analogously we could say that our circus monkey from the introduction has optimal approximation error (because it assigns grades at random, it's hypothesis space includes all possible models), on the other hand, its estimation error is very large - the grades it assigns randomly vary around the bets possible grades he can generate. The teaching assistant is a much better (and robust) learning algorithm so his grades no not vary much around the optimal grades. On the other hand, certain biases, such as being too strict or too lenient might reduce the teaching assistant's hypothesis space and result in a higher approximation error.

Note that the learning algorithms that are most often used in practice are so used because they have a good *inductive bias*. That is, the assumptions made by the algorithms restrict the hypothesis space in a way that benefits estimation error without having too much of an adverse effect on approximation error.

2.3 Consistency of empirical risk minimization

At first glance empirical risk minimization (ERM) is an appealing and intuitive approach, however, we are yet to discuss what (if any) guarantees we have when we use it. In particular, how good is the model h_n compared to h^* or, in terms of risk, what can we say about R_n as an estimator of R?

First, ERM is in general not unbiased. In fact, for learning algorithms it is typically positively biased - the empirical risk of the selected model will underestimate the true risk or $R_n(h_n) < R(h_n)$, hence the selected model will typically not be $h*$ and $R(h_n) < R(h^*)$. This is the principle that underlies overfitting.

We do know that $R_n(h) \xrightarrow{P} R(h)$ for any $h \in \mathcal{H}$ as it follows from the Law of large numbers. However, is this enough to conclude that $R(h_n) \xrightarrow{P} R(h^*)$ (that is, that the selected model's risk will converge in probability to the risk of the optimal model or that ERM is consistent)? It turns out that a stronger condition is necessary (and sufficient) - that the convergence in probability is uniform across h: $\sup_{h\in\mathcal{H}} |R_n(h) - R(h)| \xrightarrow{P} 0$ (convergence in the Glivenko-Cantelli sense).

We now show that the uniform convergence is sufficient (the necessary part is beyond the scope of these notes). We begin with two inequalities:

- The empirical risk of the model that maximizes empirical risk will be less or equal to the empirical risk of the optimal model: $R_n(h_n) \leq R_n(h^*)$.
- The risk of the model that maximizes empirical risk will be greater or equal to the risk of the optimal model: $R(h_n) \geq R(h^*)$.

Note that neither h_n nor h^* have to be unique for this argument to hold. The above inequalities lead to

$$0 \leq R_n(h^*) - R_n(h_n) + R(h_n) - R(h^*),$$

rearranging the terms gives

$$= R(h_n) - R_n(h_n) + R_n(h^*) - R(h^*),$$

and

$$\leq \sup_{h \in \mathcal{H}} \left(R(h) - R_n(h)\right) + R_n(h^*) - R(h^*).$$

The model h^* is unknown but a constant, so the term $R_n(h^*) - R(h^*)$ goes to 0 (in probability) for reasons discussed above. Under the assumption of uniform convergence the term $\sup_{h \in \mathcal{H}} \left(R(h) - R_n(h)\right)$ also converges to 0. Therefore, uniform convergence is a sufficient condition for both $R(h_n) \xrightarrow{P} R(h^*)$ and $R_n(h_n) \xrightarrow{P} R_n(h^*)$!

2.4 Generalization error bounds

Informally, uniform convergence will be met if the learning algorithm's hypothesis space is not too rich. In the other extreme case the model could have infinite capacity. In such a case, we also say that the problem is unlearnable - no matter how many data points we have, there will always be a model that fits all the data points but does not attain $R(h^*)$.

The Vapnik-Chervonenkis (VC) learning theory formalizes this notion of richness (VC dimension, capacity). We will not discuss this theory, only state one of the most important concepts - a bound on true risk. For more on VC theory and structural risk minimization refer to Ch. 5 of *Scholkopf, B., & Smola, A. J. (2001). Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press.*

As we already mentioned before, we will typically have $R(h) \geq R_n(h)$. That is, the empirical risk will underestimate true risk. But can we know by how much? VC theory provides an answer in the form of an upper bound on the true risk:

With probability $1 - \delta$ we have

$$R(h_n) \leq R_n(h_n) + O\left(\sqrt{\frac{d_{VC}}{n} \log \frac{n}{d_{VC}} - \frac{1}{n} \log \delta}\right),$$

where n is the number of observations and d_{VC} the VC dimension of the learning algorithm.

Intuitively, this makes sense - the more data we have, the closer the empirical risk will be to the actual risk, and the more capacity to learn the algorithm has, the less we can say about the true risk from empirical risk alone.

This gives rise to another principled approach to learning - *structural risk minimization* (SRM). The idea of SRM is to divide the hypothesis space $\mathcal{H}$ into nested subsets $\mathcal{H}_1 \supset \mathcal{H}_2 \supset ... \supset \mathcal{H}_k \supset \mathcal{H}$ the capacity of which we are able to compute. Then we perform empirical risk minimization for each subset select the model that minimizes the upper bound on the true risk. That is, the model, where the sum of the empirical risk and the capacity penalty is the lowest. As opposed to ERM, where we only minimize the empirical risk. Note that Support Vector Machines are a representative of SRM - it can be shown that maximizing the margin corresponds to minimizing the bound on empirical risk.

Regularization (penalized likelihood)

$$h_n = \arg\min_{h \in \mathcal{H}} (R_n(h) + \lambda C(h)),$$

where $C(h)$ is some measure of model complexity and λ is a regularization constant, is related to SRM and can be thought of as an approach to limiting the capacity of a model. Similarly, Bayesian statisticians limit the capacity of a model by putting priors on the parameters.

Note that cross-validation and related procedures that estimate a model's true risk in fact indirectly estimate the learning algorithm's capacity to learn.

2.5 Choosing a loss function

Choosing a loss function is first and foremost a practical decision that depends on the problem at hand. And, at least in theory, there exists for every possible loss function a scenario where that loss function is the most appropriate choice - trivially, when the goal is to minimize that loss function. However, that does not imply that any loss function can be justified for any scenario. In this section we'll offer some arguments when and why certain loss functions should be chosen and why some commonly used loss functions have typically undesirable properties.

But first note that in the ERM framework the loss function is part of the learning/estimation process. That is, the decision making part is part of the learning process.[9] As such, it does not make sense to learn the models using one loss function and then select a model using a different loss function. For example, to train logistic regression using MLE and a decision tree using mutual information and then select the model that has the best accuracy.

There are exceptions to this rule - situations where it is practically unfeasible to train the model using the desired loss function. For example, 0-1 loss (accuracy) is discontinuous and difficult to optimize for. We might instead use a substitute or *surrogate loss*, such as hinge loss or log loss, to simplify optimization, even though we will in the end select the model with the best accuracy.

In some cases we might not know what the learning algorithm will be used for, for example, when developing a new learning algorithm. In such cases we might prefer a more general loss functions, such as log loss, which results in optimizing for the (information-theoretic) distance between the predicted and true distribution.

[9]As we will see in the Bayesian decision theory section, there are alternative frameworks where that is not the case.

The examples of ERM estimators in Section 2.1 already offer some insight. If we are interested in predicting the median, for example, we should use the absolute loss, not the quadratic. The use of MLE (ERM with log loss) also comes with some guarantees.[10]

We provide furher insights into choosing a loss function in Section 2.6 and in the empirical examples in Section 4.

2.6 Proper scoring rules

The framework of *scoring rules* and the concept of *propriety* offers some insights into the selection of loss functions. This framework developed in the area of probabilistic forecasting, in particular, in meteorology. That is, in problems where the goal is to predict the entire distribution/density, not just point predictions. For more information on scoring rules, the following paper is a great starting point *Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. Journal of the American statistical Association, 102(477), 359-378.*

Formally, a scoring rule is a function $S : \mathcal{P} \times \mathcal{Y} \rightarrow \mathbb{R}$, where $\mathcal{P}$ is typically the set of all measurable functions on $\mathcal{Y}$. In other words, a scoring rule $S(p, y)$ takes a probability distribution/density $p \in \mathcal{P}$ and an outcome $y \in \mathcal{Y}$ and assigns a score.

Scoring rules and loss functions are closely related. For example, log loss is a scoring rule: $S_{\log}(p, y) = \log p(y)$. Other loss functions can also be formulated as scoring rules, although this is more complicated for loss functions that are based on point predictions and one could argue that these are not actually scoring rules. For example, 0-1 loss is $S_{\text{0-1}}(p, y) = 1$ if the mode of p is y and 0 otherwise. If the model only predicts the mode (and not the distribution), we can still compute this score by assuming that the predicted category was the mode. And a scoring rule that corresponds to squared loss could be defined as $S_{\text{sq}}(p, y) = -(E_p[Y] - y)^2$. Similarly, we can compute this score for a model that only gives point predictions by assuming that the model's prediction is the expectation.

Now we are ready to define what we mean by propriety. A scoring rule S is said to be *proper* if the reward is maximized at $S(p_{\text{true}}, y)$, where p_{true} is the true distribution. That is, if reporting the true distribution maximizes the score. A scoring rule is said to be *strictly proper* if p_{true} not only maximizes the score but is in fact the only p that maximizes the score. That is, reporting anything other than the true distribution will reduce your score. If a scoring rule is not proper, we'll refer to it as *improper.*

In words, a proper scoring rule encourages the forecaster (model) to report the true probability. If a scoring rule is not strictly proper, the model might get a maximum score even though it is not reporting the true probabilities, but the model that does will still get the maximum score. For improper scoring rules, however, it would be possible for a model to get a better score by reporting something other than the true distribution.

Strictly proper scoring rules

So which of the commonly used loss functions are strictly proper? For categorical/discrete y the log loss and quadratic score $S(p, y) = -\sum_{i=1}^{k}(p_i - I_{y==i})^2$ (also known as the Brier score) are strictly proper. Log loss is also strictly proper for continuous y. Another commonly used

[10] As we discussed in the probability course, MLE is under certain regularity conditions consistent, asymptotically normal, and asymptotically optimal/efficient.

(or severely underused) proper scoring rule is the rank probability score (RPS), which can be applied in the discrete

$$S_{\text{RPS}}(p, y) = -\sum_{i=1}^{k}(F_p(i) - I_{y \geq i})^2$$

and continuous setting

$$S_{\text{RPS}}(p, y) = -\int (F_p(x) - I_{y \geq x})^2 dx,$$

where F_p is the CDF of p. In words, the RPS is the squared distance between the predicted and empirical CDF, where the latter is based on the single actual outcome. Unlike the log and quadratic loss, the RPS takes into account the ordering of the categories.

Note that we are deliberately consistently writing all scoring rules in a way such that maximizing S maximizes the reward. In different sources you will encounter a different sign and maximizing the reward will correspond to minimizing the score. For example, quadratic score (Brier score) is more often written as $\sum_{i=1}^{k}(p_i - \hat{\theta})^2$, although this is inconsistent with the name *score* as the reward is maximized by minimizing the function.

Rules that are not strictly proper

Many commonly used loss functions are not strictly proper. For example, 0-1 loss (or accuracy) is not strictly proper. In order to maximize our score, we do not have to report the true probabilities (or what we believe to be the true probabilities). Instead, it suffices to report probabilities where the mode will correspond to what we believe to be the mode. For example, if in the binary case we believe the true distribution to be $p_1 = 0.6, p_2 = 0.4$, we will maximize our score by reporting and distribution that satisfies $p_1 > 0.5$. This highlights the danger of using probabilities from models that were selected on the basis of their accuracy.

This argument can be extended to any loss function or scoring rule that partitions the models into non-singleton equivalence classes. That is, any two models that always have the same mode will have the same accuracy and for any probabilistic prediction there are always infinitely many different probabilistic predictions that have the same mode. Therefore, the score can not be strictly proper, because the true distribution will have other distributions in its equivalence class. It follows that sensitivity, specificity, F1, and AUC are not strictly proper, squared loss (or mean squared error) is proper but not strictly proper in the continuous case (as we stated before, it is strictly proper in the discrete case), and absolute loss is not strictly proper in neither the discrete nor the continuous case.

Furthermore, some of the commonly used loss functions are in fact improper. Absolute loss is improper in the discrete case - it encourages the model to assign the maximum probability to the mode of the true distribution. Accuracy is proper in the iid setting but can be improper in the some more complex settings.[11] AUC is also improper, but in cases that are unlikely to arise in practice.[12]

[11]See `https://www.fharrell.com/post/class-damage/`.

[12]See *Byrne, S. (2016). A note on the use of empirical AUC for evaluating probabilistic forecasts. Electronic Journal of Statistics, 10(1), 380-393.*

Summary

When interested in any sort of inference from the probability or density reported by the model, we should avoid the use of scoring rules that are not strictly proper. In cases where we are interested only in forecasting a particular aspect of the probability or density, such as the actual outcome, the use of rules that are not strictly proper is acceptable. The use of improper scoring rules is difficult to justify.

Choosing between strictly proper scoring rules, of which there are infinitely many, unfortunately again comes without clear rules or recommendations, other than if you are unsure about the decision making requirements in your particular problem (or do not care to think about them), stick with the log loss. This paper is a good starting point for further investigation of the topic: *Merkle, E. C., & Steyvers, M. (2013). Choosing a strictly proper scoring rule. Decision Analysis, 10(4), 292-304.* We also provide some empirical evidence in Section 4.

3 Bayesian decision theory

Bayesian decision theory offers a principled alternative to ERM. First, let's quickly review the Bayesian inference framework. Given a parametric model (*likelihood*) $p(y|\theta)$ and a *prior* distribution on the parameters $p(\theta)$, we infer the *posterior* distribution of the parameters using Bayes' theorem:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}.$$

When the goal is not only to infer the parameters but also to make predictions for new data, we do so through the *posterior predictive* distribution:

$$p(y_{\text{new}}|y) = \int p(y_{\text{new}}, \theta|y)d\theta = \int p(y_{\text{new}}|\theta, y)p(\theta|y)d\theta = \int p(y_{\text{new}}|\theta)p(\theta|y)d\theta.$$

The posterior predictive distribution is a mixture of of likelihoods over the posterior distribution of the parameters. Note that the final step is due to conditional independence of y_{new} and y given the parameters θ (all the information from y required to predict a new observation is contained in the likelihood and parameters).

We can see how the Bayesian framework is based on updating our belief about the parameter values (and subsequently new data).

The *Bayes risk* is defined as:

$$R(\hat{\theta}|y) = \mathbb{E}_{p(\theta|y)}[\ell(\theta, \hat{\theta})] = \int \ell(\theta, \hat{\theta})p(\theta|y)d\theta$$

and the estimator $\hat{\theta}$ that maximizes the Bayes risk is called a Bayes estimator. Observe that the only difference between Bayes risk and ERM risk is that the latter is an expectation over the unknown distribution of the data, while the former is an expectation over the posterior distribution of the model.

Similarly, we can define the Bayes risk when the loss function is defined on observations not on parameters:

$$R(\hat{y}|y) = \mathbb{E}_{p(y_{\text{new}}|y)}[\ell(y_{\text{new}}, \hat{y})] = \int \ell(y_{\text{new}}, \hat{y}) p(y_{\text{new}}|y) dy_{\text{new}}.$$

Note, however, that there is no conceptual difference between the two - it is just that in one the quantity of interest is the parameter, while in the other it is the observation. Bayes risk can be computed for any quantity by using the posterior distribution of that quantity. For example, if we had a discriminative model.

3.1 Examples of Bayes estimators

We will see from the below examples that in the Bayesian framework the inference and decision theory are completely separate (unlike in ERM, where they are done jointly). This is extremely convenient, because parameter inference can be done without having to know how the model will be used, which substantially simplifies decision making.

Quadratic loss

Suppose we have a likelihood $p(y|\theta)$, a prior $p(\theta)$ and we compute the posterior distribution $p(\theta|y)$. We will not make any further assumptions - the results of this section will apply to any choice of likelihood and prior and any data.

For the first example suppose our loss function is the quadratic loss $\ell(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2$. Let's compute the Bayes estimator - the estimator that maximizes Bayes risk

$$R(\hat{\theta}|y) = \int (\theta - \hat{\theta})^2 p(\theta|y) d\theta.$$

Taking the derivative wrt $\hat{\theta}$ and setting to 0 we get

$$0 = \int (\hat{\theta} - \theta) p(\theta|y) d\theta = \int \hat{\theta} p(\theta|y) d\theta - \mathbb{E}[\theta],$$

which will hold when $\hat{\theta} = E[\theta]$. That is, the Bayes risk under the quadratic loss is maximized by reporting the posterior mean!

Again, note that in this context there is conceptually no difference between making a decision based on the parameter or if we based our loss on the predictive distribution. That is, the posterior mean is a Bayes estimator under the quadratic loss regardless.

Absolute loss

For the second example suppose our loss function is the absolute error $\ell(\theta, \hat{\theta}) = |\theta - \hat{\theta}|$. Now we have

$$R(\hat{\theta}|y) = \int |\theta - \hat{\theta}| p(\theta|y) d\theta = \int_{-\infty}^{\hat{\theta}} (\hat{\theta} - \theta) p(\theta|y) d\theta + \int_{\hat{\theta}}^{\infty} (\theta - \hat{\theta}) p(\theta|y) d\theta.$$

Taking the derivative wrt $\hat{\theta}$ and setting to 0 we get

$$\int_{-\infty}^{\hat{\theta}} p(\theta|y) d\theta - \int_{\hat{\theta}}^{\infty} p(\theta|y) d\theta = 0$$

and then

$$\int_{-\infty}^{\hat{\theta}} p(\theta|y)d\theta + \int_{\hat{\theta}}^{\infty} p(\theta|y)d\theta = 2\int_{\hat{\theta}}^{\infty} p(\theta|y)d\theta,$$

which leads to

$$\frac{1}{2} = \int_{\hat{\theta}}^{\infty} p(\theta|y)d\theta = 1 - F_{\theta|y}(\hat{\theta}).$$

Therefore, the Bayes risk is minimized by $\hat{\theta}$, such that $F_{\theta|y}(\hat{\theta}) = \frac{1}{2}$, which is the median of the posterior distribution $\theta|y$. We leave it to the reader to check that this is indeed a minimum and not an inflection point.

0-1 loss

For the third example suppose our $\theta = \{1, ..., k\}$ is categorical and our loss function is 0-1 loss $\ell(\theta, \hat{\theta}) = I_{\theta \neq \hat{\theta}}$. Now the risk is

$$R(\hat{\theta}|y) = \sum_{i=1}^{k} I_{\theta \neq \hat{\theta}} P(\theta = k|y) = 1 - \sum_{i=1}^{k} I_{\theta = \hat{\theta}} P(\theta = k|y) = 1 - P(\theta = \hat{\theta}|y).$$

So, to minimize risk, we have to pick the $\hat{\theta}$ that maximizes the posterior $P(\theta|y)$. That is, the Bayes estimator for a categorical variable under 0-1 loss is the posterior mode!

4 Toy examples

The purpose of these toy examples is twofold. First, to illustrate some of the pathologies of ERM and certain loss functions that we discussed earlier in these notes. And second, to provide some empirical evidence on how different loss functions encourage different predictions, which might be helpful in choosing which loss function to use in a particular case.

4.1 ERM discards information about the distribution of the data

To illustrate this, suppose our data are from a normal distribution and we choose absolute loss. We have already derived that the ERM estimator in this case is the sample median and that the Bayes estimator is the median of the posterior or predictive posterior. However, in this particular case the predictive posterior is normally distributed and the mean and median coincide. The mean of the posterior will typically be different from the sample mean, so in this scenario the ERM and Bayes estimator disagree!

Now we empirically verify which is better. We do this by generating 15 samples from a standard normal distribution and evaluating the average absolute loss of the estimators on 10000 independent samples from the same distribution. We repeat this process 1000 times. Results are shown in Figure 1.

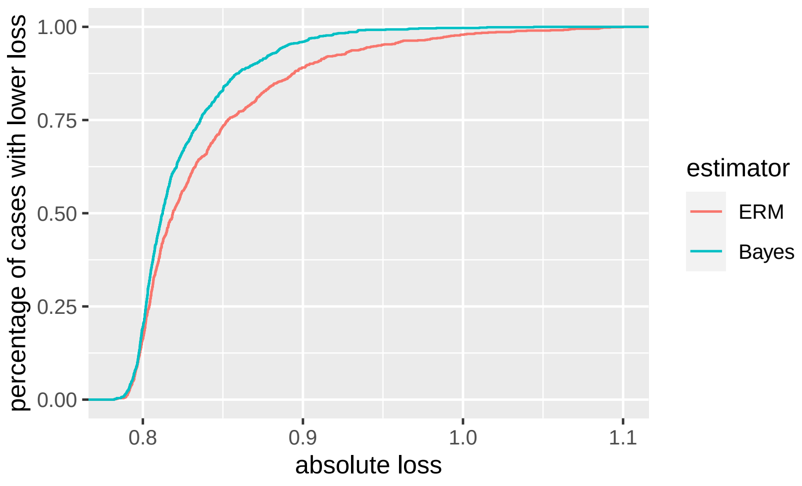


Figure 1: ECDF of average absolute loss over 1000 repetitions of the experiment. The Bayes estimator achieves lower average absolute loss.

4.2 Discrete probability forecasts

In this example we evaluate 5 different probabilistic forecasts with 5 different loss functions. The purpose is to illustrate the effect of using a particular loss function. Note that because we know the true probabilities, we are able to compute the expected loss. The results are shown and discussed in Figure 2.

This example was inspired by Minka's paper `https://tminka.github.io/papers/erm.html` where the reader can also find more discussion on the topic of ERM being an incomplete inductive principle.

4.3 Continuous point and density forecasts

In this example we evaluate 6 different density forecasts with 4 different loss functions. The purpose is to illustrate the effect of using a particular loss function. Note that because we know the true density, we are able to compute the expected loss. The results are shown and discussed in Figure 3.

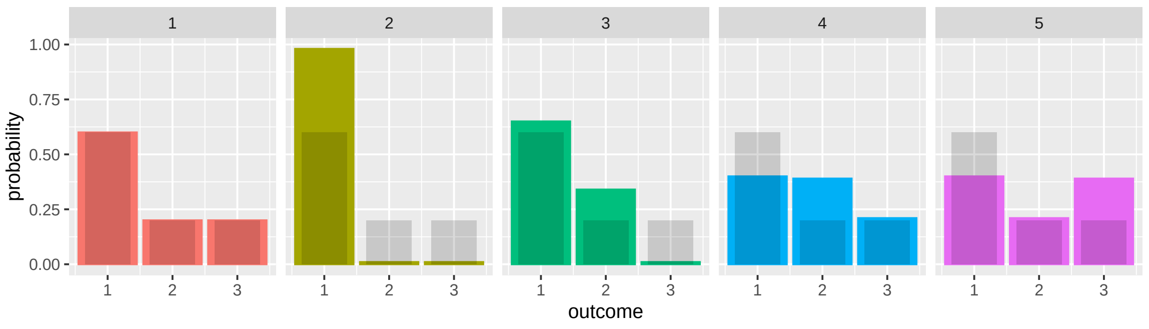
5 Estimating generalization error

Estimating the generalization error (or risk) of a model is, as the name suggests, an estimation problem. As such, our discussion of the topic will revolve around different estimators, their bias and variance.

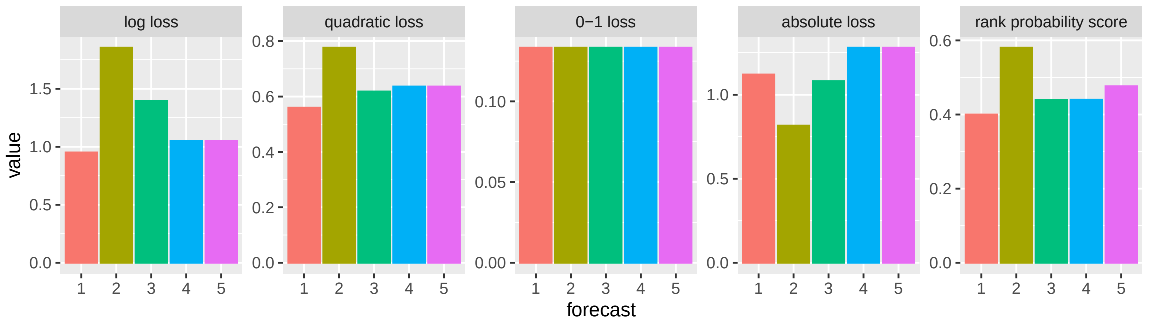
In predictive modeling practice, the most typical case is that we have a data set D_n and we want to select from a set of learning algorithms the best learning algorithm for the problem at hand. The best in the sense that the model learned by that learning algorithm on D_n will have the lowest generalization error.

5.1 Holdout estimation

The simplest but very effective estimator, when enough data are available, is the so-called *holdout estimator* of risk - estimating the risk of a model using a test set. We will discuss

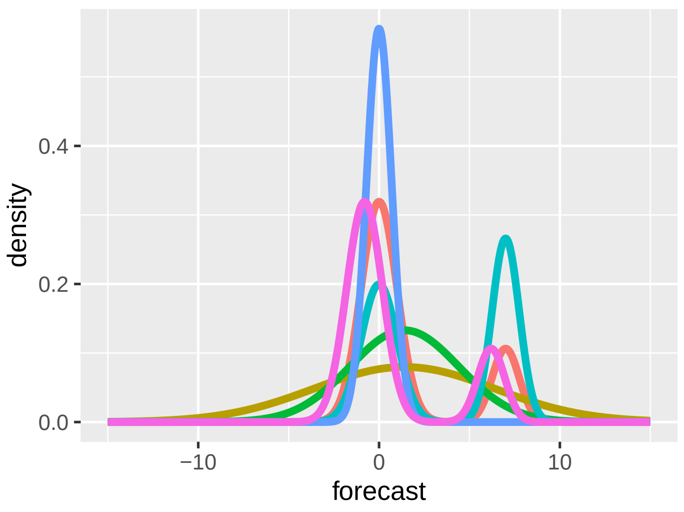


(a) five different probabilistic forecasts (in gray are the true probabilities from 1)

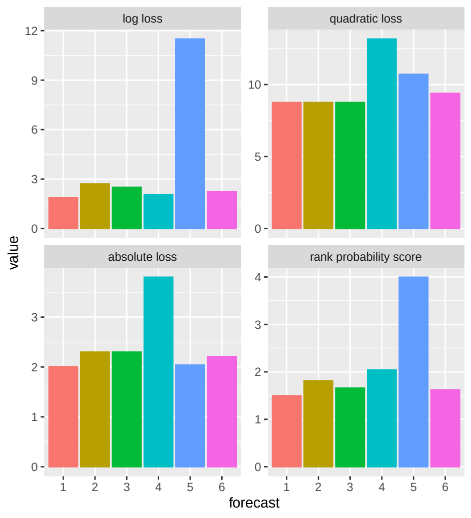


(b) expected loss for five different loss functions

Figure 2: Forecast 1 are the true probabilities. As such, Forecast 1 has the lowest log loss, quadratic loss, and rank probability score (these are all strictly proper scoring rules). It also has the lowest 0-1 loss (it is a proper scoring rule) but all the other 4 forecasts, despite being substantially different, attain the same 0-1 loss, because all five forecasts assign the most probability to outcome 1. Forecast 2 gets the lowest absolute loss but is the worst (or tied for worst) for all other 4 loss functions. Forecasts 4 and 5 are symmetric (if there is no order) and their losses are identical for all loss functions, except the rank probability score, where Forecast 4 is better. Finally, Forecast 4 has better log loss than Forecast 3 but for the quadratic score it is the other way around - the log loss is more sensitive to small probabilities.



(a) six different density forecasts (true density is in red)



(b) expected loss for four different loss functions

Figure 3: Forecast 1 is the true density. As such, Forecast 1 has the lowest (or tied for lowest) log loss, quadratic loss, absolute loss, and rank probability score (all four are proper scoring rules). However, quadratic and absolute loss are not strictly proper - they evaluate the forecaster only on a particular aspect of the predicted density (or a point prediction). So, any point forecaster whose prediction matches the mean or median, respectively, will attain minimum loss. Similarly, in our case, all density forecasts with the same mean or median, respectively. Log loss and RPS are strictly proper, so only the true density attains the minimum loss. This again illustrates that proper scoring rules should be preferred. As was the case with discrete forecasts, the RPS takes into account the ordering. Forecasts 4 and 6 illustrate the preferences of log loss and RPS.

three different variants that differ only in what is used as the test set.

Idealized case

First, we imagine a scenario when additional to the set D_n we have a set of D_m, which are also IID samples from the data generating process that we can use for estimating risk. This is an idealized case because in practice it is unlikely that we would have so much data available that we could afford a large enough separate training and test sets - in most cases, we would eventually train the model on all available data, before using it for the task at hand.

However, if we can afford a completely separate test set and our D_n is large enough so that we will not train the model on all data (that is, we are still interested in estimating the generalization error of the model learned on D_n), then we can employ a very simple but effective estimator of true risk:

$$\hat{R} = \frac{1}{m} \sum_{(x_i, y_i) \in D_m} \ell(y_i, h_n(x_i)).$$

In words, we train the model on D_n and then estimate its generalization error with its average error on the test data. This estimator is unbiased - we are using the actual model that we are interested in and we are evaluating it on a completely independent set of IID observations from our data generating process. Due to the Law of large numbers, we know that the average will converge to the expected value.

Furthermore, due to the Central limit theorem, we know how the variance of the estimator will behave - it will be approximately normally distributed with its variance $\sqrt{m}$ smaller than the variance of the loss function. This leads to a simple estimator of the variance of the above estimator:

$$\sigma^2_{\hat{R}} = \frac{\sigma^2_\ell}{m},$$

where σ^2_ℓ is the variance of the loss function of h_n for the data generating process. We typically don't know σ^2_ℓ, but we can estimate it from the sample using the standard unbiased estimator of variance

$$\hat{\sigma}^2_\ell = \frac{1}{m-1} \sum_{(x_i, y_i) \in D_m} (\ell(y_i, h_n(x_i) - \hat{R})^2.$$

In many cases an estimate of risk and its variance are sufficient to make a decision. However, we might also proceed by constructing confidence intervals, performing a hypothesis test, or employing the bootstrap or a Bayesian analysis as an alternative.

Note that the above estimator is valid regardless of the underlying distribution of the losses - the average is a good estimator of expectation. The estimator of variance, however, depends on the Central limit theorem, which is an asymptotic normality argument that might not hold for small data in combination with extreme deviations of the underlying distribution from the normal distribution. To illustrate, suppose we are interested in estimating 0-1 loss (accuracy) and our model achieves $\sim 94\%$ accuracy on an independent test set of 16 observations (15 out of 16 correctly classified). The estimated variance and standard deviation of the losses would therefore be 0.0625 and 0.25, respectively. This would lead us to a standard error estimate of $SE = \frac{0.25}{\sqrt{16}} = 0.0625$. If we naively construct a $\pm$ 2 standard errors CI around the estimate, we would get $[\frac{15}{16} - 0.125, \frac{15}{16} + 0.125] = [0.8125, 1.0625]$, which of course does not make sense, because accuracy must be between 0 and 1. Such a CI will underestimate the uncertainty in the estimator.

In such cases we should either take the underlying distribution into account and construct a more appropriate estimator or, as a simpler but typically no less effective alternative, estimate the variance using non-parametric bootstrap or some other robust estimator of variance.

Train-test split

A more common case would be to reserve a randomly selected subset of the available data D_n for testing. We will refer to the held-out data as D_m and to the remaining training data as D_{n-m}.

The discussion of the idealized cases transfers to this more common case, with one exception. We are no longer estimating the performance of the model trained on D_n. Instead, we are estimating the performance of the model trained on D_{n-k}:

$$\hat{R} = \frac{1}{m} \sum_{(x_i, y_i) \in D_m} \ell(y_i, h_{n-k}(x_i)).$$

As such, our estimator will be a *biased* estimator of the performance of the model h_n (trained on all data). In particular, it will typically be positively biased (overestimate risk) or, in other words, pessimistic (underestimate performance). The larger the m, the more biased it will be. Note that this is based on a fairly reasonable assumption that the learning algorithm is *smart* - that its true risk is a decreasing function in n (that is, that the performance of the models it learns does not decrease if we have more training data).

Additionally, the estimator now depends on the assignment of observations to the training and test data sets. This adds to the variability of the estimator, but we can account for this variability by repeating the process multiple times, each time for a different training-test split.

Testing on training data

As an alternative, we could just train the model on all data and test it on all data:

$$\hat{R} = \frac{1}{n} \sum_{(x_i, y_i) \in D_n} \ell(y_i, h_n(x_i)).$$

In other words, we could use empirical risk to estimate true risk. Again, we can proceed as in the idealized case, with one exception. As we already stated in the section on ERM, this will work in the limit (as n grows very large). This estimator is, however, biased. In particular, it is negatively biased - it will underestimate risk and therefore overestimate the performance of the model. The more the learning algorithm is robust to overfitting and the larger the n, the smaller the bias.

5.2 Cross-validation

When it can be applied, the holdout estimator is simple and effective. In many cases, however, our n will be relatively small. So small that we can not apply holdout estimation. If we choose a large enough m (test set), we will bias our estimates, because our training data will be too small. On the other hand, if we choose a small enough m to get a larger $n - k$ (training set) and reduce the bias, we will have too large a variance of the estimator. That is, if n is relatively small, the error of the estimator will be too large, regardless of how we choose to split the data.

In such cases we need an alternative to holdout estimation that will better utilize the available data. The most common alternative is cross-validation (CV).

We refer the interested reader to the following survey paper for further reading:

Arlot, S., & Celisse, A. (2010). A survey of cross-validation procedures for model selection. Statistics surveys, 4, 40-79.

Definition and variants

We start with a very general definition of cross-validation: cross-validation is an estimator that is the average of two or more holdout estimators. That is, we split D_n into training and test sets in two or more different ways and then average the holdout estimators we compute for each individual split.

Different variants of CV arise from the different splitting schemes we can use. We can subdivide the splitting schemes into two groups: *exhaustive splitting* and *partial splitting.*

In **exhaustive splitting**, we first pick a test set size m and then compute the holdout estimator for each of the $\binom{n}{m}$ possible splits that have m test observations. Exhaustive splitting is also known as delete-d cross-validation, where d is the number of test observations ($d = m$). For $d = 1$ we get the special case of delete-1 or, more commonly, leave-one-out cross-validation (LOOCV).

In **partial splitting** we again pick a test set size m, but only compute the holdout estimator for some of the $\binom{n}{m}$ possible splits. The most popular variant is k-fold cross-validation. In $k-$fold cross-validation we partition the training data into k partitions (folds) and then compute k holdout estimations, where in each estimation one of the folds is the test set and all the other folds are the training set. For $k = n$ we again have the special case of LOOCV.

Another variant of partial splitting is Monte Carlo cross-validation. It is an approximation to delete-d cross validation - we choose an m but then instead of exhaustively computing all $\binom{n}{m}$ holdout estimations, we select, uniformly at random, only r of all possible splits with m test observations. The same split can in theory be selected more than once. That is, Monte Carlo cross-validation is, as the name already suggests, a Monte Carlo approximation to delete-d cross-validation. The number of samples r is a trade-off between approximation error and computation time - it should be large enough to make the approximation error small enough.

k-fold cross-validation

This is the most popular variant of cross-validation. The properties of the $k-$fold cross-validation estimator of risk are revealed by studying how the estimator depends on choice of k. Analogous to the discussion of the holdout estimator, the $k-$fold CV estimator is typically positively biased - it underestimates the performance, because it is based on models that are trained on fewer than n data points. As long as the learning algorithm is smart, the bias of the estimator will be decreasing in k - the larger the k, the smaller the bias.

This, unfortunately, does not imply that LOOCV is the best choice. First, LOOCV is computationally very intensive, which is an important practical consideration. Second, if the model is not stable, that is, if it is very sensitive to small changes in the training data (see Section 5.3), the variance of the $k-$fold CV estimator might increase with k and the lowest error is achieved by some k between 2 and n. This has led to the sometimes misleading recommendation that $10-$fold cross-validation should be preferred to LOOCV for model selection. For a more detailed discussion of this, refer to the following seminal paper:

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In IJCAI (Vol. 14, No. 2, pp. 1137-1145).

For stable models, however, LOOCV will be the best estimator of risk. In fact, it has been shown for several learning algorithms that that is the case (see the survey paper we reference at the beginning of the cross-validation section).

And third, although an estimator of risk has lower bias and variance than another estimator of risk, it is still possible that it is worse at model selection. That is, being better at estimating the risk of models A and B separately does not imply that one is better at estimating the difference. This is related to an interesting phenomena where we can be worse at selecting a model although we have a larger training set and a larger test set (also known as the cross-validation paradox). See Section 5.4 for an illustrative example.

Similarly, k-fold cross-validation for $k \ll n$) might in some cases be better than LOOCV for model selection, because the additional bias is actually beneficial. See the following paper for further discussion:

Zhang, Y., & Yang, Y. (2015). Cross-validation for selecting a model selection procedure. Journal of Econometrics, 187(1), 95-112.

Note that $k-$fold cross-validation is computationally much less intensive than exhaustive splitting (we have re-train the model only k times, regardless of the number of data points), which is the main reason for its popularity. However, this comes at a cots. As was the case with holdout estimation, if n is relatively small, the estimator might strongly depend on the partitioning. That is, choice of partition introduces additional variance to the estimation. However, as with holdout estimation, we can quantify and mitigate this additional variance by repeating the process for different partitions and then aggregating the results. This is also known as repeated $k-$fold cross-validation. Note that in the special case of LOOCV there is only one possible way of partitioning the data, so repetitions do not make sense.

5.3 Additional topics and references

In this section we list some additional topics. With the exception of model stability, all of these topics are very relevant to everyday practice, so every practitioner should be at least aware of the basic principles and techniques.

Nested CV

Many learning algorithms also have additional hyper-parameters (or tunable parameters). For example, the regularization parameter λ, learning rates in neural network models, and number of trees in random forests. The values of these parameters either have to be set manually by the user or fit to the data. In the latter case, we would like to do so optimally, which typically involves another layer of model evaluation.

CV can also be employed to select the values of these hyper-parameters. When performing CV for model evaluation, we can, for every holdout split, employ CV on the training data for that split, to perform an additional layer of CV. That is, for each holdout split, we perform CV (using only the training data) to select the optimal value from a set of values. We then use that hyper-parameter value to train a model on the training data and evaluate it on the test data for that fold. This is known as *nested cross-validation.*

The following paper is a good starting point for further investigation of nested cross-validation. It also provides empirical evidence that in cases where we have few hyper-parameters, we can replace the computationally intensive nested CV with a simpler approach of selecting hyper-parameter values that maximize CV estimates of performance:

- Wainer, J., & Cawley, G. (2018). Nested cross-validation when selecting classifiers is overzealous for most practical applications. arXiv preprint arXiv:1809.09446.

Stratified cross-validation

Stratified cross-validation is a combination of stratified sampling and cross-validation. Stratified sampling is in essence an estimator variance reduction technique. If we in our population have distinct groups (strata), such that withing each group the quantity of interest varies little, but between groups it varies a lot, we can reduce the overall variance of the estimator by treating each group as an independent population and then aggregate the estimates. Typically, we draw from each group proportionately to the size of the group. However, we might also draw more samples from the group with higher variance, to further reduce the overall error of the estimate.

In model evaluation stratification is applied to cross-validation predominately as a technique for avoiding problems with zero observations from a class in the training set in classification problems. If we partition the data (or perform a training-test split) completely at random, it might happen that a certain class value is not represented in the training error, which will prevent most learning algorithms from learning it. Instead, we partition the data so that each class value is proportionately represented in every fold. This, however, introduces a bias - if our estimate of the relative frequency of each class is poor, which it might be, because it is based on a finite number of observations, we might under or overestimate model performance, depending on what the model learns and what the true relative frequencies are.

Predictive model evaluation for large data sets

Sometimes our data set will be so large that CV approaches will be impractical. In such cases we typically have enough data to employ holdout evaluation. If, however, we would still like to perform CV, we can resort to approximate methods and predictive criteria.

In practice, it we commonly apply simple-to-compute predictive criteria, such as Mallow's C_p, AIC, BIC, DIC, and WAIC. They are typically composed of the empirical risk (error on the training set) and a correction that is based on model complexity, such as the number of parameters of the model for linear models. For most of these it turns out that they are an approximation to LOOCV. For some models, such as linear regression, we can, somewhat surprisingly, compute LOOCV estimates from the original model fit without re-training the model on n subsets. A detailed introduction to the topics of this paragraph can be found in the following lecture notes:

`https://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/21/lecture-21.pdf`

This paper is a good starting point for understanding AIC, LOOCV and other predictive criteria from a Bayesian perspective:

- Gelman, A., Hwang, J., & Vehtari, A. (2014). Understanding predictive information criteria for Bayesian models. Statistics and computing, 24(6), 997-1016.

approximate LOOCV from samples of the posterior distribution:

- Vehtari, A., Gelman, A., & Gabry, J. (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. Statistics and computing, 27(5), 1413-1432.

Non-IID data

In this text we focused on the IID case. In practice, however, there are many relevant cases where observations are dependent and the methods discussed do not apply, such as temporal and spatial data.

In such cases we typically resort to holdout evaluation. For temporal data, we use a rolling (train on all observations up to this point) or sliding (train on last k observations) window approach that respects the time component - to avoid an optimistic bias, we should not make predictions using data that were not available at the time of the observation that we are predicting for. For spatial data, this is more complicated, because these data lack the natural ordering of temporal data.

However, it is possible to employ CV procedures even when the data are strongly correlated. These two papers are a good starting point:

- Roberts, D. R., et al. (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography, 40(8), 913-929.
- Bergmeir, C., Hyndman, R. J., & Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. Computational Statistics & Data Analysis, 120, 70-83.

Model averaging instead of model selection

The goal of model selection is to select the best model and then we typically proceed with inference or prediction as if that model is the true model. In practice, however, multiple models may provide an adequate explanation of the world. Discarding all but one, just because its performance was better by what is possibly a relatively small amount, may lead to underestimating the uncertainty and poorer predictions. An alternative to model selection is to keep all the models and weigh them according to their performance. This idea is very natural in the Bayesian framework and is known as Bayesian model averaging. Stacking, bagging (for example, random forests), and other *ensemble learning* techniques can also be viewed as a form of model averaging.

The following classic and more recent review papers are an excellent starting point for further investigation of Bayesian model averaging:

- Hoeting, J. A., Madigan, D., Raftery, A. E., & Volinsky, C. T. (1999). Bayesian model averaging: a tutorial. Statistical science, 382-401.
- Fragoso, T. M., Bertoli, W., & Louzada, F. (2018). Bayesian model averaging: A systematic review and conceptual classification. International Statistical Review, 86(1), 1-28.

We refer the interested reader to the following survey paper and book on ensemble learning:

- Dietterich, T. G. (2000). Ensemble methods in machine learning. In International workshop on multiple classifier systems (pp. 1-15). Springer, Berlin, Heidelberg.
- Zhou, Z. H. (2012). Ensemble methods: foundations and algorithms. CRC press.

Model stability

In our discussions of the variance of CV estimators, we mentioned that it depends strongly on the properties of the learning algorithm. In particular, on the stability of the learning algorithm - how sensitive it is to changes in the training data set.

The following paper is a good starting point and review of different formalizations of algorithmic stability:

Kutin, S., & Niyogi, P. (2002). Almost-everywhere algorithmic stability and generalization error. In Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence (pp. 275-282).

5.4 Toy examples

The cross-validation paradox

With this toy example we illustrate a counter-intuitive phenomena where increasing both the training and test set results in worse performance in model selection.

Our example is similar to the one performed in

Zhang, Y., & Yang, Y. (2015). Cross-validation for selecting a model selection procedure. Journal of Econometrics, 187(1), 95-112.

We generated a simple binary classification data set with three input variables X_1, X_2, and X_3. The class proportions are 40% ones and 60% zeros. Conditional on $Y = 1$, the three input variables are standard normal. Conditional on $Y = 0$, X_2 is standard normal, but $X_1 \sim N(0, 0.4)$ and $X_2 \sim N(0, 0.3)$. That is, X_1 and X_2 allow us to discriminate at least to some extent between $Y = 1$ and $Y = 0$.

We will compare (A) Linear Discriminant Analysis (LDA) that uses only X_1 and X_2 and (B) LDA that uses all three input variables. We are interested in classification accuracy. Because input variable X_3 is just noise, we already know that model (A) is better than model (B).

Our experiment goes as follows: We randomly generate 30 training cases and 70 test cases (the class proportions are enforced for both). We then train models (A) and (B) on the training set and estimate their performance on the test set. We this process 50000 times and count the proportion of cases where the average test accuracy of (A) is better than (B) (ties are ignored). That is, we estimate the probability of the procedure selecting the best model.

We repeat this process for larger training and test datasets (130-170, 230-270, 330-370, and 430-470). For comparison, we also conduct the experiment where only the test set increases (30-170, 30-270, 30-370, and 30-470). We summarize the results in the following table:

Experiment	1	2	3	4	5	6	7	8	9
n_{train}	30	130	230	330	430	30	30	30	30
n_{test}	70	170	270	370	470	170	270	370	470
$\hat{p}_{\text{correct}}$	0.606	0.566	0.559	0.559	0.554	0.640	0.660	0.680	0.690
SE	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002

Counter-intuitively, the probability of selecting the better model decreases, although both the training and test data set increase! For comparison, if we increase only the test set, the probability of selecting the better model increases!

As the training data set increases, we get better models, and as the test data set increases, we get a better estimate of the models' performance. However, as the this example demonstrates, this does not imply that we will be better at selecting the better model. The underlying mechanism is that the difference between models decreases faster (as a function of training data size) than our ability do discern differences increases (as a function of test data size). Note that due to the mechanism illustrated by this example, k-fold cross-validation for $k \ll n$ might in some cases be better than LOOCV for model selection.