Point and interval estimates: the fundamental task of machine learning

When measuring the length of an object, we are essentially using a ruler to perform a statistical estimate. The measured result in the format of $x \pm \delta$ gives a point estimate x and an interval estimate of δ. The point and interval estimates are the most fundamental tasks in statistical inference. Another branch of statistic inference, hypothesis testing, usually also requires us do point and interval estimates first.

The fundamental tasks of most machine learning algorithms, which are either explicitly or implicitly connected to statistical models, are essentially doing point and interval estimates. The aforementioned x might be a function point in the functional space. Different algorithms give different point estimates of x (bias) and different interval estimates(variance). Note that many sampling distributions are not Gaussian, so the intervals might not be symmetric.

For unbiased estimator, the variance can be used to characterize the prediction error. For biased estimator, as in most of machine learning practice, the prediction error (e.g., squared error) for example, include both variance and bias.

Understanding bias and variance from closed-form solutions

Linear regression solved by singular value decomposition (SVD)

The following is a brief summary from the notes "linear regression theory_SVD_projection to column space_basis change_gradient descent.pdf", where many insights of machine learning can be obtained.

A linear regression problem can be described by finding x that satisfies $\min_x ||Ax - b||_2^2$. One way to find the best x is to project vector b to the column space of A, $C(A)$. This is because the error of two vectors is automatically minimized due to the 'vertical' projection. So the key is to find the p_A first.

For the most general case where A can be singular and any shape, the projection operator to $C(A)$ has the form $P = AA^\dagger$, where $A^\dagger$ is the pseudo-inverse of A. From SVD, $A = U\Sigma V^T$, we can write $A^\dagger = V\Sigma^{-1}U^T$, where in Σ^{-1} we only reverse the non-zero elements.

We thus obtain $p_A = AA^\dagger b = AV\Sigma^{-1}U^T b$. Therefore the SVD solution x can be obtained as,

$$Ax = p_A = AV\Sigma^{-1}U^T b \Rightarrow x = V\Sigma^{-1}U^T b = \sum_{i=1}^{r} \frac{v_i u_i^T b}{\sigma_i}$$

where r is the rank of A.

Bias and variance in the model

If we assume there is an error δb in b, then the solution becomes

$$x = \sum_{i=1}^{r} v_i \left(\frac{u_i^T b}{\sigma_i} + \frac{u_i^T \delta b}{\sigma_i} \right)$$

If the singular values σ_i is very tiny due to the deficiency of a matrix, then small error on the data will give big error as σ_i appears in denominator. This is one of reasons for variance in machine learning. For bias, it should in the first place from the linear model assumed. This should account for most of bias of the model. However, bias may also be arised from regularization as explained later.
The model above presents a variance 'amplification' mechanism, and in many cases can be a major mechanism of the observed variance. However, variance can be there even without 'amplification'. For example, when we measure the length with a ruler multiple times and take their mean. The systematic problem of the ruler such as the not accurate unit length may cause system error (or bias). The variance, however, can be attributed to the sum of many independent and random variables (e.g. blink of eyes, random estimates of the length from the ruler....). Thus the measured lengths usually have normal distribution with a specific variance.

An intuitive way of understanding bias-variance relation

Whenever we want to catch the details of the data such as high spatial frequency of images or high temporal frequency of time signals etc., we normally will have to catch the noise. Mathematically, details of a model such as higher order in polynomial fitting, will result in tiny singular values in the relevant matrix. These tiny values amplify the noise and thus increase the variance. Thus if we want to catch the details of data in order to reduce bias, then we have high probability to increase the variance. On the other hand, if we instead don't want catch the details of the data, then variance will be small but bias will be big due to over-simplified model.

Bias-variance trade off for a single model and a single dataset

https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff (https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff)

Decomposition of squared error

For regression, we can use the squared error and decompose squared error there into two parts. However, in classification problems, we don't have such type of squared error. There are many proposals to describe this, and there is yet no agreement on what is the "right" and/or the most useful formalism. Here we use the squared error decomposition to understand the bias-variance trade off.

$y = f(x) + \epsilon$, where the noise ϵ has zero mean and variance σ^2. An estimate function $\hat{f}(x)$ will be found to approximate the true function $f(x)$. The **squared error** is decomposed into:

$$E[(y - \hat{f})^2] = Var[y] + Var[\hat{f}] + (f - E[\hat{f}])^2 \equiv \sigma^2 + Var[\hat{f}] + Bias[\hat{f}]^2$$

It contains variance, bias and irreducible parts. Apart from the intrinsic noise that we have no way to eliminate, both the square of bias and variance contribute to the squared error, which describe how the model $\hat{f}$ deviates from the true function. As described later, within the same model, the variance and bias part cannot be improved simultaneously. However, it is possible that we can reduce both bias and variance by choosing different models and more data.

Side note about the difference between squared error and variance:

- For unbiased estimator, bias part will be zero and thus squared error is equivalent to variance.
- For biased estimator, the variance is a subset of squared error.
- Be careful not to confuse the squared error and variance, even though they can be same in special cases.

Closed form bias-variance trade-off relation for K nearest neighbor (KNN) model

From a closed-form for k-nearest neighbor (KNN) regression model, we see that there is actually a trade-off between bias and variance for a single model. If we try to decrease the bias, we might increase the variance, or verse versa.

In the case of KNN regression, a closed-form expression exists that relates the bias-variance decomposition to the parameter k (Wikipedia):

$$E[(y - \hat{f}(x))^2 \mid X = x] = \left(f(x) - \frac{1}{k} \sum_{i=1}^{k} f(N_i(x)) \right)^2 + \frac{\sigma^2}{k} + \sigma^2$$

where the bias (first term) is a monotone rising function of k, while the variance (second term) drops off as k is increased.

Reducing variance with a bias-variance trade off -- Regularization

Understanding regularization with closed-form linear regression

Coming back to the linear regression problem introduced earlier, if A is with full column rank matrix, then $A^T A$ is a full-rank $n \times n$ matrix and thus invertible. In this case, the projection operator to $C(A)$ is defined as $P = AA_{left}^{-1} = A(A^T A)^{-1} A^T$, where the pseudo-inverse $A^\dagger$ is replaced by A_{left}^{-1} (details in linear algebra notes).

Thus we have

$$Ax = p_A = A(A^TA)^{-1}A^Tb \tag{1}$$
$$\Rightarrow x = (A^TA)^{-1}A^Tb$$

If A^TA is not full rank, then it is singular and not invertible. A way to handle this is to use regularization by introducing the regularizing term to the cost function.

$$C(x) = \frac{1}{2}\sum_i \left(b_i - a_i^Tx\right)^2 + \frac{1}{2}\lambda\|x\|^2$$

where a_i^T is the row vector of matrix A, and thus a_i is a column vector obtained from $(a_i^T)^T = a_i$. This indicates different data samples are given in terms of different rows. Taking derivative w.r.t. x and setting it to zero gives,

$$\sum_i \left(b_i - a_i^Tx\right)a_i = \lambda x \Rightarrow x = \left(\lambda I + \sum_i a_ia_i^T\right)^{-1}\left(\sum_j b_ja_j\right)$$

In the matrix form, we have

$$x = (A^TA + \lambda I)^{-1}A^Tb$$

where I is identity matrix and λ is the regularization parameter. A^TA is square symmetric matrix and is assumed to be singular here. After adding λI to A^TA, the singular and symmetric matrix A^TA might become full rank (for $\lambda > 0$) and invertible. Therefore, the variance problem will be reduced.

The effects of regularization

- We now know that the introduction of regularization can significantly improve the rank of the matrix. From the SVD solution earlier, the singular values denominator σ_i will be far away from zero. Singular values are not just with clear-cut zero or not-zero values in numerical calculations. In other words, a matrix is not just non-invertible with zero singular values or infinity condition number. In practice, we may encounter many very tiny singular values which correspond to ill-conditioned problems (very big but not infinity condition number). In this case, regularization will suppress the big variance caused by tiny singular values.
- While the tiny singular values σ_i is no longer tiny due to the addition of λ and thus variance problems are mitigated, we also add λ to those not-so-tiny singular values. These later additions are not only unnecessary as those σ_i are not tiny to cause variance, but also introduce bias to the final solutions. For large λ, the bias introduced might even be dominant over the bias for a particular model chosen.

Other ways of understanding regularization

Regularization favors a simpler model with small weights

- l_2-regularization relies on the assumption that a model with small weights is simpler than a model with large weights. Thus, by penalizing the square values of the weights in the cost function you drive all the weights to smaller values. It becomes too costly for the cost to have large weights. This leads to a smoother model in which the output changes more slowly as the input changes.
- An intuitive example: if you have two ways of fitting your data, such as $y = 2x_1 + 0x_2$ or $y = x_1 + x_2$, you prefer the latter because the penalty is $2^2 + 0^2 = 4$ in the former and $1^2 + 1^2 = 2$ in the latter. In general the effect of each weight on the prediction will be linear but its penalty will be quadratic. Thus it will pay off to put lots of small values instead of just a few big ones.
- From Eq.(1), we also know that the weights, i.e. the solution x, will generally becomes smaller because the addition of λ to A^TA. In other words, regularization generally reduce the numerical value of weights.
  https://stats.stackexchange.com/questions/247940/how-does-the-l2-regularization-penalize-the-high-value-weights
  (https://stats.stackexchange.com/questions/247940/how-does-the-l2-regularization-penalize-the-high-value-weights)

l_2 regularization is equivalent to Gaussian Prior

https://stats.stackexchange.com/questions/163388/l2-regularization-is-equivalent-to-gaussian-prior/163450#163450
(https://stats.stackexchange.com/questions/163388/l2-regularization-is-equivalent-to-gaussian-prior/163450#163450)

- l_2 regularization imposes a Gaussian prior to the weights. When λ is very small, the variance of the Gaussian prior, which is $\frac{1}{\lambda}$ is very big. This enforce many weights are similar, and hence penalize the big weights.
- l_1 regularization, however, is equivalent to imposing a Laplace prior to the weights and thus can favor a few even only one weights by adjusting regularization parameter.

Regularization achieves the trade-off of minimizing the cost and keeping the weights small

- In the regularization for linear regression, the effect of regularization comes in the term $\theta_j = \theta_j \left(1 - \alpha\frac{\lambda}{m}\right)$, where α is learning rate, λ is regularization parameter, and m is the number of samples. Thus the regularization parameter always makes the weights smaller.
- If λ is too big, it can make all the weights (except constant θ_0) very small or even zero, but cost function will not be minimized. Too big λ usually gives big bias. If λ is very tiny, then it cannot make the weights smaller and hence we cannot obtain a simpler model to reduce variance. Therefore, we need choose an optimal λ to achieve both goals of minimizing cost and making weights small.

Technical details for regularizing matrix-form linear regression

- Details in the regularization part of Machine learning course in coursera.
- Normally we don't penalize θ_0. So in the sum of regulation term, we have $\sum_{i=1}^{n}$. If we do, it makes no big difference. When regularize normal equations, remember it is not adding λI. The first element should be zero for the constant term θ_0.

Reduce variance/bias without bias-variance trade-off : ensembling

How ensembling suppresses variance - from a statistical perspective?

We are estimating the sample mean of a single sample. It is easy to calculate the mean and the sample variance from this single sample. However, this calculation from a single sample is not so accurate and it is usually with big variance. To have a better estimate, we often obtain many independently samples to achieve a better estimate. See details in the notes of statistics. Now consider the distribution of a statistic, sample mean $\bar{X}$, with a normal distribution (center limit theorem), then it is easy to obtain its variance as:

$$Var(\bar{X}) = Var(\frac{\sum_{i=1}^{n} X_i}{n}) = \frac{1}{n^2} Var(\sum_{i=1}^{n} X_i) = \frac{n\sigma^2}{n^2} = \frac{\sigma^2}{n}$$

So as the number of samples increases, the standard error will decrease. The above results about sample mean assume that samples are drawn from an identically, independent distribution (iid). If we drop the independent assumption but consider an averaged correlation p among samples, then we have https://en.wikipedia.org/wiki/Variance (https://en.wikipedia.org/wiki/Variance), https://en.wikipedia.org/wiki/Covariance (https://en.wikipedia.org/wiki/Covariance)

$$Var(\bar{X}) = p\sigma^2 + \frac{1-p}{n}\sigma^2$$

Although the variance is now related to correlation, the idea of using more samples to reduce variance is still valid. First, we can make sure the samples from a bootstraping process are as independent as possible. Second the variance is decreasing as the number of samples increases. However, iid is best. Therefore, to reduce variance with more samples, it is better to obtain independent as possible.

How ensembling suppresses variance - from noise canceling perspective ?

Let us resort to the SVD solution to linear regression

$$x = \sum_{i=1}^{r} v_i \left(\frac{u_i^T b}{\sigma_i} + \frac{u_i^T \delta b}{\sigma_i} \right)$$

Tiny singular values σ_i may significantly amplify the noise to the date and hence cause big variance in predictions. However, if we have a huge number of samples, then the noise arising from each sample may partially cancel each other due to their different signs. This is true, particularly if the noise follows a normal distribution. This is actually the well used technique in signal processing.

Bagging and boosting

Bootstrap aggregation (bagging) and boosting are two major examples for reducing variance/bias without a trade-off. Bagging essentially employs the aggregation of data, while boosting employs the aggregation of different algorithms.

As shown earlier, the samples obtained should be as much iid as possible to efficiently suppress noise. For example, because decision trees can suffer from high variance, bagging is employed to reduce the variance. However, because the trees obtained by bootstrapping are often highly correlated, people often use random forest to replace the normal bagging. Random forest forces the trees to be more independent and thus provides better variance reduction.

More details on bagging and boosting are in the notes related to decision trees.

Reduce variance without bias-variance trade-off : improving matrix rank

As mentioned several times, the tiny singular values of a matrix is one of key reasons causing prediction variance. Therefore, we may handle the matrix in the first place to increase the prediction performance.

The key reason leading to tiny singular values is the correlated columns or rows in the relevant matrix. Therefore it is highly desirable to eliminate correlated features before modeling. Here we refer to the correlation among features but not among feature and targets.

Using principal component analysis to reduce high variance is not good due to, e.g. we don't use target labels, and thus throw away useful information. The elimination of correlated columns is a process of step-by-step eliminating one column if it is strongly correlated to at least one another column.

When calculating correlation, be careful of the so-called 'spurious' correlation due to the general shift as in many time-series signals.