Fundamentals of ML

Mehdi Rostami

University of Toronto

Bank of Montreal

September 4, 2023

Regression function

- Find $a(X)$ that minimizes the risk $\mathbb{E}(Y - a(X)|X)^2$?
- $a(X) = \mathbb{E}(Y|X)$
- This is called "regression function" that in ML we want to estimate as it minimizes the risk
- When Y is binary, $\mathbb{E}(Y|X) = P(Y = 1|X)$
  - What is the expected value of Bernoulli distribution?
  - $Z \sim bernoulli(p)$ then $E(Z) = p$
  - $Z|W \sim bernoulli(P(Z = 1|W))$ then $E(Z|W) = P(Z = 1|W)$

What is Learning?

- PAC learning: Probably approximately correct
- Roughly speaking, a class of algorithms (like the class of all neural nets) is called PAC learnable, if, for any dataset, and any ϵ and δ (small positive numbers), there's an algorithm in that class so that the error (MSE) is smaller than ϵ with probability at least $1 - \delta$.

Model vs algorithm

- Different ML algorithms try to estimate this function using the input data using some universal function approximator like polynomial function (Taylor expansion), tree-based algorithms, splines, neural networks, etc.
- Note that I'm saying "algorithm", not models. People use the word "model" but that's technically wrong. These are algorithms or function approximators; we minimize the error between the observed and predicted values so the pick the best algorithm among the class of all such algorithms. We are just minimizing a loss (that's not coming from a distribution assumed for the outcome-input relationship); so we're not modeling the reality.
- In addition, in practice, we use many "algorithms" and pick the best one; this is not the right way of modeling things. To model the reality, you assume some form for the reality and use data to show that, and that's all.
- All ML algorithms are non parametric. Sure, we assume that the regression function belongs to $\{P_\theta | \theta\}$, but there's no restriction on θ.

No-Free-Lunch Theorem

- For every algorithm there exists a distribution (or data) on which it fails. In other words, there's no algorithm that's the best for all tasks.
- Pick an algorithm that you claim to be best for all datasets. I'll generate a dataset that some other algorithm will outperform your candidate.
- Summary: There is no universal learner: Every learner has to be specified to some task, and use some prior knowledge about that task (which class to choose), in order to succeed.

Linear Regression Model

- Goal: Calculate the first-order approximation of the Taylor expansion of $\mathbb{E}(Y|X)$
- Assumptions: Nothing! No linearity, normality or conditional IID is needed.
- For prediction, we don't need distributional assumptions; With the right interpretation of coefficients, linearity is also not needed
- Goal: Estimate the parameters

$$\mathbb{E}[Y|X] = X\beta = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p \tag{1}$$

- Once somehow estimated, prediction for a new observation $X = x$:

$$\hat{\mathbb{E}}[Y|X = x] = x\hat{\beta} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_p x_p \tag{2}$$

Estimation

- With either assuming that the data is normal or just using least square method, we have this objective function: $\min_\beta \mathbb{E}(Y - X\beta)^2$ which for a finite sample amounts to $\min_\beta \sum_{i=1}^{n}(y_i - X_i\beta)^2$
- A closed form solution is

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

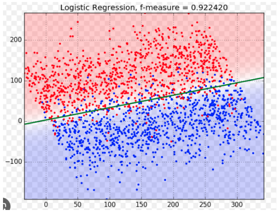
- The prediction for a new x is $x\hat{\beta} = x(x^T x)^{-1} x^T y$
- If $p < n$ and $X^T X$ is not singular, then $\hat{\beta}$ is a consistent and unbiased estimator of β, and if normality of y is assumed, it's asymptotically normal too (good for confidence interval calculation).

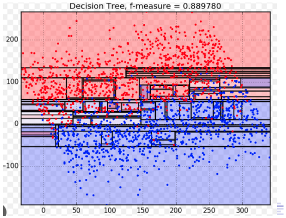
Bias-Variance Tradeoff

- Let $f(X)$ be the best approximator in a class of algorithms (e.g. best decision-tree/linear regression) of Y and $\hat{f}(X)$ be it's data-driven estimator,

$$\mathbb{E}\left[Y-\hat{f}(X)\right]^2=\mathbb{E}\left[f(X)-\hat{f}(X)\right]^2+\mathbb{E}\left[Y-f(X)\right]^2=$$
$$\left\{\mathbb{E}\left[\hat{f}(X)-\mathbb{E}[\hat{f}(X)]\right]^2+\left(\mathbb{E}\left[f(X)-\mathbb{E}[\hat{f}(X)]\right]\right)^2\right\}+\left\{\mathbb{E}\left[Y-\mathbb{E}[Y|X]\right]^2+\mathbb{E}\left[\mathbb{E}[Y|X]-f(X)\right]^2\right\}$$

- Error = Estimation Error + Approximation Error = {Estimator Variance + Estimator Bias} + {Measurement error + Class-specific error}
- **Example**:
  1. $f(x)$ is a linear regression and $\mathbb{E}\left[Y-\hat{f}(X)\right]^2=4$ (1+0+1+2)
  2. $f(x)$ is a decision tree and $\mathbb{E}\left[Y-\hat{f}(X)\right]^2=4$ (1.5+.5+1+1)





Interpretation of Bis-Variance components

- **Estimation Error** is summation of the variance of the data-driven estimator and its bias from the best un-realistic estimator in the chosen class of algorithms.
  1. Depends on both data and the chosen class
  2. The bias component can be reduced by choosing a different class of algorithms. Unchanged by sample size!
  3. The variance component can be reduced if, for a chosen estimator, you force it to less focus on data.
- **Approximation Error** is the summation of measurement error in the data-generating process and the error of the chosen class of algorithms
  1. Unchanged by data or its size.
  2. The measurement error component is due to an error in Y measurements; can't be reduced.
  3. The class-specific error is induced by the class of algorithms we pick. Can't reduce it after you choose a class! Can only reduce it by choosing a more complex class of algorithms.

Predictions variance in linear regression

- For linear regression, the in-sample error and $\mathbb{E}(X^T\beta - X^T\hat{\beta})^2 = c\frac{p}{n}$ out-of-sample error is $\mathbb{E}(x^T\beta - x^T\hat{\beta})^2 = c\frac{p}{n-p-1}$
- When $p \ll n$ things are well.
- What happens if $p < n$ but $p \to n$? The model doesn't generalize well.
- How about $p \geq n$? Need hard-feature selection
- By increasing # of features in the model, complexity increases $\to$ bias goes down, variance goes up
- The above shows the dimensionality effect or curse of dimensionality in linear regression. It's valid in other algorithms too.
- Curse of dimensionality: Error increases by # of features.

Bias-variance of linear regression

- For linear regression predictions:

$$\mathbb{E}\left[Y - X\hat{\beta}\right]^2 = \mathbb{E}\left[X\hat{\beta} - \mathbb{E}[X\hat{\beta}]\right]^2 + \left(\mathbb{E}\left[X\beta - \mathbb{E}[X\hat{\beta}]\right]\right)^2 + \mathbb{E}\left[Y - X\beta\right]^2 =$$
$$\mathbb{E}\left[X\hat{\beta} - X\beta\right]^2 + \mathbb{E}\left[Y - X\beta\right]^2$$

- Since the linear regression parameters are unbiased, the second term is zero! So, in linear regression, we have no bias!
- Why linear regression is not the best:
  - If $p \ll n$ then the approximation error is very high
  - If p is large, not only the approximation error is high (although improved compared to low p), but the variance is high!

Bias-variance tradeoff - con't

- To minimize risk, there's a tradeoff, called the bias-variance tradeoff.
- Choosing the class of algorithms to be a very rich class decreases the approximation error but at the same time might increase the estimation error, as a rich class might lead to overfitting.
- Choosing the class of algorithms to be a very small set reduces the estimation error but might increase the approximation error or, in other words, might lead to underfitting.
- To have intuition about why there's a tradeoff, consider the extreme. The lowest bias can be achieved by a perfect fit of the data (zero error), but it doesn't generalize to unseen data at all, meaning high variance. On the other hand, the lowest variance is a constant, but it will have huge errors and is badly biased.
- Bias and variance cannot be directly observed, because $\hat{\theta}$ calculated on the data is a number that we cannot find its expectation accurately. Bootstrap can help estimate them
- The training MSE gives an intuition about the bias, and the validation MSE gives an intuition about the variance.

Regularization

- A solution to avoid an increase in variance is regularization
- Any modification to an algorithm that decreases the variance while increase the bias (class-specific error) can be regarded as a regularization method
- We sacrifice lower bias to get lower variance; In fact, the model complexity goes down by regularization; increasing one component of the error and decreasing another
- One way to impose regularization in loss minimization problems is to shrink the model parameters while estimating them during the optimization.
- Search $\hat{\beta}$'s that minimize the least squares in a constrained subspace

$$\min_{\beta} \mathbb{E}(Y - X\beta)^2, \quad \text{subject to}$$

$$h(\beta) < \frac{c}{\lambda}$$

- ATT conditions in convex optimization $\min_{\beta} \mathbb{E}(Y - X\beta)^2 + \lambda h(\beta)$
- This is not the only way of imposing regularization to ML algorithms. There are a few more that will be reviewed later. However, the above way of regularizing help study the properties.

Examples

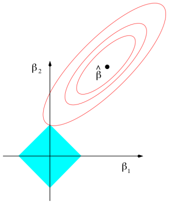
- ATT conditions in convex optimization $\min_\beta \mathbb{E}(Y - X\beta)^2 + \lambda h(\beta)$
- **Ridge** (L_2): $h(\beta) = ||\beta||^2 \rightarrow$ Shrinkage and unique solution (the latter was the trigger for discovering the idea of regularization)
- **LASSO** (L_1): $h(\beta) = ||\beta|| \rightarrow$ Sparsity and shrinkage. Could generate multiple solutions.
- **Elastic net**: $h(\beta) = ||\beta|| + ||\beta||^2 \rightarrow$ Inherits both properties

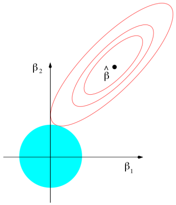
Visual interpretation

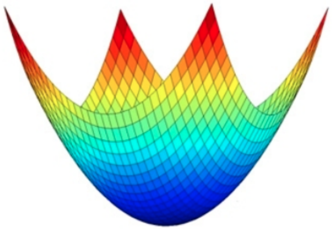
- For example: $\min_\beta \mathbb{E}(Y - X\beta)^2 + \lambda||\beta||$ or

$$\min_\beta \mathbb{E}(Y - X\beta)^2, \quad \text{subject to}$$

$$||\beta|| < \frac{c}{\lambda}$$







LASSO

- Very effective method in practice as compared to other linear models
- The best first method to try for feature selection
- Selective Inference can give uncertainty measures (p-values) for the estimators of the parameters.
- λ is a measure of complexity; very large values of λ zero out all coefficients, while smaller values could cause overfitting. Requires hyperparameter tuning to select λ.
- Larger the regularization parameter λ, tighter the constraint optimization and thus more shrinkage.
- Extreme cases $\lambda = 0$ we impose no regularization and $\lambda = \infty$ the algorithm learns nothing

Bayesian interpretation

- Assuming **conditional normality** for y, the log posterior becomes the regulation objective - because posterior is the multiplication of likelihood and prior.
- The normal prior for $\beta \rightarrow$ Ridge
- The Laplace prior for $\beta \rightarrow$ LASSO
- How? Because posterior $\propto$ likelihood $\times$ prior:

$$P(\beta|Y,X) \propto e^{-\sum_{i=1}^{n}(y_i - x\beta)^2} \times e^{-\alpha||\beta||^2},$$
$$P(\beta|Y,X) \propto e^{-\sum_{i=1}^{n}(y_i - x\beta)^2} \times e^{-\alpha||\beta||}.$$

- inimizing log posterior is the same as minimizing the regularized loss seen before.
- Intuition: In regularization we have less focus on the data $\rightarrow$ We have less variance (remember the statement from a few pages back)

Properties of regularized estimators

- Helps bias-variance tradeoff for predicted values; overall better fit
- First consequence $\rightarrow$ the regularized estimator is biased.
  - That is $\mathbb{E}[\hat{\beta}_{regularized}] \neq \beta$. Easy to see in the case of Ridge.
- Second consequence $\rightarrow$ the regularized estimator is less variable.
  - That is $Var(\hat{\beta}_{regularized}) \leq \hat{\beta}_{OLS}$
  - Easy to see for Ridge when orthogonal covariates, $\hat{\beta}_{ridge} = \frac{n}{n+\lambda}\hat{\beta}_{OLS}$
  - Similar result in non-orthogonal case
  - Or with the change in the data, the regularized estimator varies less.
  - Intuition for lower var: It's as if we focus less on the data by introducing an informative prior.
  - Variance of predictions is low: $X\hat{\beta}_{regularized}$; the lower the values of β, the less sensitive to the variations (or errors/deviations) from X.
  - Stability of algorithm when L_2 regularization is used is $\propto \frac{1}{\lambda n}$
  - Stable algorithms don't overfit
- How can we see in practice/simulations the variance of other regularized estimators?

Other types of regularization

A regularization technique doesn't necessarily need to add a penalty to the loss function! Anything that decreases the model complexity (and the variance) can be regarded as regularization.

- Early stopping
- Batch learning (randomly) imposes some regularization (Like in NNs)
- Inputting a random subset of features into the model (like a random forest)
- Ensembling
- Dropout
- ?

Drawback of informal regularization: Hard to study the theoretical properties.
Think about how these regularizations decrease focus on the data that help reduce variance of the estimator.