Machine Learning Algorithms Overview

Hanchung Lee

February 2020

1 Linear Regression

1. **What are the basic concepts? What problem does it solve?**

   Linear Regression, in essence, is trying to fit a hyperplane through the data points that minimizes the sum of the distances of all the points to that hyperplane. That hyperplane is then used to predict the location of new data points. In other words, linear regression is a model $f: R^{d+1} \rightarrow R$ for a given data pair (x, y) where $x \in R^{d+1}$ and $y \in R$. It is if if the function f is linear, with weights w of the model is the parameters to be learned. Since regression is a mapping to R, it can be used to estimate a value given a set of inputs x.

2. **What are the assumptions?**

   Regression assumes that the output y is linearly dependent on x. However, it will be linear dependent as long as w is linear. It also assumes error term is a normally distributed random variable.

3. **What are the steps of the algorithm?**

   We model linear regression problem $\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{b}$ using both least squares (LS) or maximum likelihood (ML) solutions:

   $$\textbf{LS}: \ arg\min_{w} \|y - \mathbf{X}w\|_2^2$$

   $$\textbf{ML}: \ arg\max_{w} -\frac{1}{2\sigma} \|y - \mathbf{X}w\|_2^2$$

   Or, alternatively, solving it using vector calculus approach, we solve $\frac{\partial}{\partial X} \|\mathbf{X}\mathbf{W} + \mathbf{b}\|_2^2 = 0$, and we get

   $$\mathbf{X} = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{b}$$

4. **What is the cost function?**

   Regression can be done using loss functions that spans the R space. Mean square error and mean absolute error are the common cost function of choice. We used mean squared error for our formulation.

   Mean squared error (L2 error) $\|y - y'\|_2^2 = (y - \mathbf{X}w)^2$

5. **Whare are the advantages/disadvantages?**

   Linear regression is one of the most fundamental machine learning model. It is easy to implement and train. The trained model is also highly explainable. The modeling framework can be extended by different tricks to make it more robust, such as feature engineering/extension/selection, kernel tricks, and adding regularization.

   On the flip side, linear regression only model linear relationships between dependent and independent variables (which can be polynomials), which is not enough for more complex relationships. It is also sensitive to anomalies and requires data normalization to prevent that. Also, it requires more samples than the number of parameters as a rule of thumb, which is common across most machine learning models.

2 Ridge Regression

1. **What are the basic concepts? What problem does it solve?**

   Ridge regression is an extension of base regression with an `L2 norm` regularization term $\lambda \|\mathbf{W}\|_2^2$ where λ is the learning rate that adjusts the regularization of parameters $\mathbf{W_i}$. [1] The λ term is also called the *shrinkage*. And when $\lambda = 0$, it is just regular linear regression.

   In other words, we use ridge regression to fix if a linear regressor has high variance to improve the out of the sample fit of the model. We can find the learning rate parameter λ using cross validation.

2. **What are the assumptions?**

   As with linear regression, Ridge regression assumes that the output y is linearly dependent on x. However, it will be linear dependent as long as w is linear. It also assumes error term is a normally distributed random variable.

3. **What are the steps of the algorithm?**

   From the descriptions above, we can model ridge regression as linear regression problem with a regularization term.

   $$\mathbf{y} = \mathbf{XW} + \mathbf{b} + \lambda \|\mathbf{W}\|_2^2$$

   Thus, the loss function using least squared solution would be:

   $$L_{W_{ridge}} = \|\mathbf{y} - (mathbfX\mathbf{W} + \mathbf{b})\|_2^2 + \lambda \|mathbfW\|_2^2$$

   Pulling in the bias term into W and solving it for W, we get:

   $$\mathbf{W} = (\lambda I + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{Xy},$$

   where the λ term gets cancelled out as λ goes to ∞.

4. **What is the cost function?**

   Regression can be done using loss functions that spans the R space. Mean square error and mean absolute error are the common cost function of choice. We used mean squared error for our formulation.

   Mean squared error (L2 error) $\|y - y'\|_2^2 = (y - \mathbf{X}w)^2$

5. **Whare are the advantages/disadvantages?**

   Ridge regression extends from Linear regression with a regularization term. Thus it shares similar properties as linear regression: easy to implement, easy to train. The trained model is also highly explainable. The modeling framework can be extended by different tricks to make it more robust, such as feature engineering/extension/selection.

   On the flip side, it also shares some of the problems with linear regression only model, where it expects linear relationships between dependent and independent variables (which can be polynomials), which is not enough for more complex relationships. It is also sensitive to anomalies and requires data normalization to prevent that. Also, it requires more samples than the number of parameters as a rule of thumb, which is common across most machine learning models.

3 Lasso Regression

1. **What are the basic concepts? What problem does it solve?**

   Lasso regression is an extension of basic regression with an `L1 norm` regularization term $\lambda \|\mathbf{W}\|_1$ where λ is the learning rate vector that adjusts the regularization of parameters $\mathbf{W_i}$. [1] The λ term is also called the *shrinkage*

   In other words, Lasso is similar to Ridge where both helps adding biases to the model to improve the out of the sample fit of the model. Lasso is especially useful for dealing with sparse data.

2. **What are the assumptions?**

   The same as regression, assuming `i.i.d.` of sampled data.

3. **What are the steps of the algorithm?**

   Same as regression except adding an L1 norm term to penalize Weight coefficients of some variables.

   Cross validation is used to select the regularization hyperparameter λ.

4. **What is the cost function?**

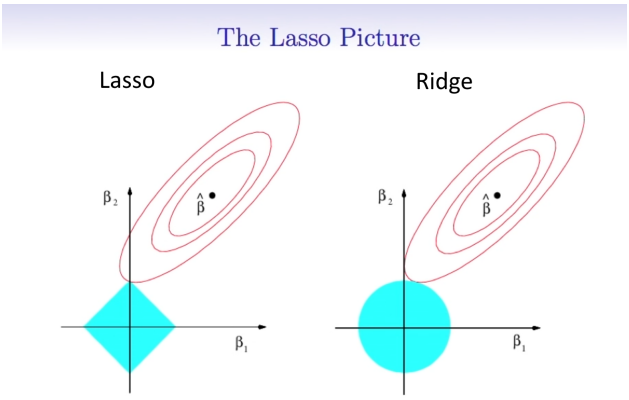
$$\mathbf{W}^{lasso} = argmin_{\mathbf{W}} \left\{ \frac{1}{2} \sum_{i=1}^{N} (y_i - \mathbf{W}_0 - \sum_{j=1}^{P} x_{ij} \mathbf{W}_j)^2 + \lambda \sum_{j=1}^{P} \|\mathbf{W}_i\|_1 \right\}$$

   [1], where the first term within the *argmin* is the mean squared error and the second term is the Lasso regularization. This can be solved with convex optimization.

5. **Whare are the advantages/disadvantages?**

   The prime difference between Lasso (L1) vs Ridge (L2) regularization of regression is that Lasso has the capability of forcing coefficients to 0, and in effect, zeros out an feature variable. In essence, Lasso regression is doing feature selection.

   In a geometric sense, in high dimeisions, L1 norm will have many sharped edges and corners. During convex optimization, if the optimization hits the edges or corners, the coefficient become zero. ON the other hand, L2 norm is a smooth surface thus it won't hit exactly model



   Thus, we can say that Lasso Regression yields a *sparse* model where only a subset of the variables is in play.

   Ridge is more suited for models with dense features while Lasso is more suited for models with sparse features. Use cross validation to determine the most suitable regularization.

4 Gradient Boosting

1. **What are the basic concepts? What problem does it solve?**

   Gradient boost decision tree models works by training many trees in sequence and ensemble the trees together at the end. When training sequentially, each of the newer trees are trained on the residual error observed by the previous tree. GBDT models tends to fit the training data well, with low bias and high variances. Because of that, it uses a learning rate λ to scale contribution from a new tree. The idea is to take many small steps towards the right direction and ensemble them together result in a better prediction.

2. **What are the assumptions?**

   Decision tree based models makes no assumption about the distribution of the underlying data. In other words, decision trees are non-parametric.

3. **What are the steps of the algorithm?**

   Gradient boost initializes by building `depth=1` node. It builds a tree depending on the depth and splits parameters. Then for the following trials it kept on building new trees to minimize the residual errors $(y - \gamma)$from the previous tree, and scale the contribution with a learning rate λ. The whole process is repeated M times and the maximum depth of the individual trees is J.

   A formal definition is as follows [1]

Algorithm Gradient Tree Boosting Algorithm

(a) Initialize $f_o(x) = argmin_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$

(b) For $m = 1$ to M:

i. For $i = 1, 2, \ldots, N$ compute the pseudo residual term $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$

ii. For a regresion tree to the target r_{im} given terminal regions $R_{jm}, j = 1, 2, ..., J_m$

iii. For $j = 1, 2, ..., J_m$, find the new predicted value γ
$\gamma_{jm} = argmin_\gamma \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$

iv. Update $f_m(x) = f_{m-1}(x) + \lambda \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

(c) Output $\hat{f}(x) = f_M(x)$

---

4. **What is the cost function?**

   Given the standard loss function setup, $L(f) = \sum_{i=1}^{N} L(y_i, f(x_i))$, proper loss function $L(f)$ can be used for different settings. A summary table is as follows [1]:

Setting	Loss Function	$\frac{-\partial L(y_i, f(x_i))}{\partial f(x_i)}$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regression	$\|\|\|y_i - f(x_i)\|\|\|$	sign[$y_i - f(x_i)$]
Regression	Huber	$y_i - f(x_i) for \|y_i - f(x_i)\| \leq \delta_m$ $\delta_m sign[y_i - f(x_i)] for \|y_i - f(x_i)\| > \delta_m$ where $\delta_m \alpha th - quantile\{\|y_i - f(x_i)\|\}$
Classification	Multinominal Deviance	kth component: $I(y_i G_k) - p_k(x_i)$

5. **What are the advantages/disadvantages?**

   Gradient boost is very similar to AdaBoost. The key difference is that for Gradient Boost, it can have a tree size that is larger than a stomp (depth of 1). Because gradient boosting is a boosting model that ensembles sequentially trained models, it is a naturally a low bias model. However, the downside is to a low bias model is the variance can be high and has to be reduced by adding regularization. Also, because the model is trained sequentially, it is compute intensive to train.

5 Naive Bayes

1. **What are the basic concepts? What problem does it solve?**
2. **What are the assumptions?**
3. **What are the steps of the algorithm?**
4. **What is the cost function?**
5. **Whare are the advantages/disadvantages?**

6 Support Vector Machines

1. **What are the basic concepts? What problem does it solve?**

   Support Vector Machines finds the hyperplane that separates the data with the greatest margin. Two key concepts of Support Vector Machine is it uses soft margins instead of max margins to reduce variances and it uses the kernel trick to implicitly increase the feature space to reduce bias.

   It can be used for both classification and regression problems. It uses a kernel function to compute the inner product of two variables in a higher dimensional space. The data that resides in the soft margin are called support vectors, with distance to the soft margin line ϵ. Hyperparameter **C** is learning rate and is found using cross validation.

2. **What are the assumptions?**

   Support Vector Machine treats all the data points as equals so it is important to standardize the data. Otherwise it does not make assumptions about the underlying data distribution.

3. **What are the steps of the algorithm?**

   Support Vector Machine is a convex optimization problem where it **Solve:** $\min_{w,b,\epsilon} \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^{n} \epsilon_i$ such that $y_i(w^T x_i - w_0) \geq (1 - \epsilon_i)$ and $\epsilon_i \geq 0, \quad \forall i$. After derivation, Support Vector Machine function is:

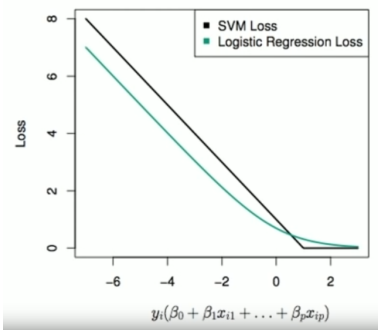
   $$f(x) = w_o + \sum \hat{\alpha_i} K(x, x_i)$$

   where $\hat{\alpha_i}$ are zero for data not in the support set and K is the Kernel function. The common kernels are polynomial, radio basis kernel (behaves similar to nearest neighbor), and neural network kernel. [1]

nth-Degree polynomial	$K(x, x') = (1 + (x, x'))^d$
Radial basis function (RBF)	$K(x, x') = exp(-\gamma \|x - x'\|_2^2$
Neural network	$K(x, x') = tanh(\kappa_1(x, x') + \kappa_2)$

4. **What is the cost function?**

   Support Vector Machine uses hinge loss, which has the form of loss plus penalty. $L[y, f(x)] = [1 - yf(x)]_+$The following compares the negative log likelihood loss to hinge loss. Note that sharp hinge corner, that is when are set to zero.



5. **Whare are the advantages/disadvantages?**

   When classes are nearly separable, SVM does better than logistic regression and LDA. When classes are not, then SVM does similarly well as logistic regression. We can use kernel tricks with LDA and logistic regression as well, but more compute intensive.

   The downside with SVM is it does not provide a probability estimate, low interpretability, and does not do any feature selection.

7 Neural Networks

1. **What are the basic concepts? What problem does it solve?**

   According to the Universal Approximator Theorum, neural networks is an universal function approximator that can be used approximate any functions that map $X \to Y$. It can be used abstracts away some of the feature engineering steps in classical machine learning tasks. For example, instead of designing image filters, CNN learns the filters itself, or instead of tagging word semantic labels, RNNs learns the relationships between words.

   The most fundamental neural network layer is a linear layer where $f(\mathbf{X}) = \sum_{m=1}^{M} g_m(\mathbf{W_m X})$, where the input values go through an affine transformation and scaled by an activation function g_m. Activation functions usually tend to scale the outputs to pseudo range bound, such as Sigmoid and Tanh.

2. **What are the assumptions?**

   The assumptions are highly dependent on the structure of neural networks. For example, some neural network architectures such as convolution neural networks or recurrent neural nets assumes that the underlying data are *i.i.d* while some other architectures such as Bayesian neural networks or Graph neural nets could have different assumptions about the relationships between feature spaces.

3. **What are the steps of the algorithm?**

   Neural networks is predominant solved using stochastic gradient descent with an auto differentiation library. First, inputs are feed forward through the network for an output $\hat{y}$, and apply the loss function. The algorithm then take the gradients of the loss function and back propagate the network to obtain the differences in parameters between target and output. The parameters are then updated with the differences. One small batch at a time.

Algorithm Stochastic gradient descent (SGD) update at training iteration k [2]

Require: Learning rate ϵ_k

Require: Initial parameter θ

while stopping criterion not met **do**

Sample a minibatch of m examples from the training set $\{x^{(1)}, ..., x^{(m)}\}$ with corresponding targets $y^{(i)}$

Compute gradent estimate: $\hat{g} \leftarrow +\frac{1}{m}\nabla_\theta \sum_i L(f(x^i;\theta), y_{(i)})$

Apply update: $\theta \leftarrow \theta - \hat{g}$

end while

4. **What is the cost function?**

   Neural networks is an flexible architecture thus different loss functions can be used for different tasks. Typical loss functions includes cross entropy and negative log likelihood for classification and mean absolute error (L1) and mean squared error (L2) for regressions.

5. **Whare are the advantages/disadvantages?**

   Rather than considering neural networks as a kind of model, it is closer to a highly customizable building blocks for machine learning models. Depends on the complexity of the network constructed, it can be capable of learning a large amount of information from data. At the same time it excels in learning implicit relationships.

   However, it is an black box model that is quite difficult to interpret and usually has to rely on large data sets and large amount of compute power to train. It is also difficult to debug and fine tune.

References

[1] Hastie, Tibshirani, and Friedman, *The Elements of Statistical Learning*. 2nd Edition, 2009.

[2] Goodfellow, Bengio, Courville, *Deep Learning*, 2016

[3] Nasirany, Thomas, Wei, and Yang, *A Comprehensive Guide to Machine Learning*, 2018