Machine Learning Study Guide

May 6, 2020

1 Supervised vs Unsupervised Learning

With supervised learning the data set contains the output. Regression is for trying to predict results with continuous output while classification is for discrete output. Unsupervised learning is when the data set has no feedback and want to derive structure by clustering based on relationships.

2 Multivariate Linear Regression

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n = \theta^T x$$

where we define the cost function by taking the mean square difference of the error:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_0(x_i) - y_i)^2$$

Assuming that the inputs are orthogonal then the estimators θ_j are equal to the univariate estimates. In order to orthogonalize the inputs:

1. regress x on 1 to produce the residual $z = x - \bar{x}1$
2. regress y on the residual z to give the coefficient $\hat{\theta_1}$

2.1 Gradient Descent

For each feature, repeat until convergence. $O(kn^2)$

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \text{ for } j = 0 : n$$

2.2 Feature Scaling

Since the range of values of raw data varies widely, objective functions will not work properly without normalization. For example, the majority of classifiers calculate the distance between two points by the Euclidean distance. If one of the features has a broad range of values, the distance will be governed by the this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. Also, gradient descent converges much faster with feature scaling than without it. Some methods include:

- Rescaling $x^{`} = \frac{x - min(x)}{max(x) - min(x)}$
- Standardization $x^{`} = \frac{x - \bar{x}}{\sigma}$
- Scaling to unit length $x^{`} = \frac{x}{\|x\|}$

2.3 Learning Rate

Ensure that the cost function is decreasing with every iteration. Declare convergence if $J(\theta)$ decreases by less than threshold ϵ.

2.4 Polynomial Regression

Combine multiple features into one ($x_1 x_2$, or x_1^2)

2.5 Normal Equation

Can calculate theta directly without iteration where $\theta = (X^T X)^{-1} X^T y$. This is slow for large n as $O(n^3)$ and need to calculate the inverse. Have to watch out for redundant (linearly dependent) features.

2.6 Homoscedasticity

An assumption of linear regression is that the error terms are the same across all values of the independent variables. While heteroscedastic error terms don't affect the quality of the estimators, it produces biased standard errors. This affects the quality of significance tests and confidence intervals. Heteroscedasticity can be resolved by transforming the dependent variable or using weighted least squares instead of OLS.

3 Classification

Binary classification is where y can take on only two values, 0 and 1. Since y values must be between 0 and 1, we can apply the sigmoid function (logistic function) to the new hypothesis.

$$h_\theta(x) = g(\theta^T X) \text{ , where } g(z) = \frac{1}{1+e^{-z}}$$

$h_\theta(x)$ will now give us the probability that our output is 1.

4 Logistic Regression

The new cost function becomes

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} -log(h_\theta(x))y - log(1-h_\theta(x))(1-y)$$

which guarantees that the cost function is convex.

$$h = g(X\theta) \text{ then } J(\theta) = \frac{1}{m}(-y^T log(h) - (1-y)^T log(1-h))$$

then gradient descent iteration becomes:

$$\theta = \theta - \frac{\alpha}{m} X^T (g(X\theta) - \overrightarrow{y})$$

5 Overfitting

Underfitting (high bias) is when the form of hypothesis function h maps poorly to the trend of the data, too simple, too few features. Overfitting (high variance) is when the hypothesis function fits the available data but does not generalize well to predict new data.

5.1 Addressing Overfitting

- Reduce the number of features
- Regularization: reduce magnitude of parameters θ_J

$$min_\theta \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

where λ is the regularization parameter. The gradient descent step becomes

$$\theta_j = \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum(h_\theta(x^i) - y^{(i)})x_j^{(i)}$$

where $1 - \theta\frac{\lambda}{m}$ will always be less than 1, reducing θ_j on every update. The normal equation becomes

$$\theta = (X^TX + \lambda L)^{-1}X^Ty$$

where L is an $(n+1) \times (n+1)$ identity matrix with $x_{0,0} = 0$. The cost function for regularization becomes

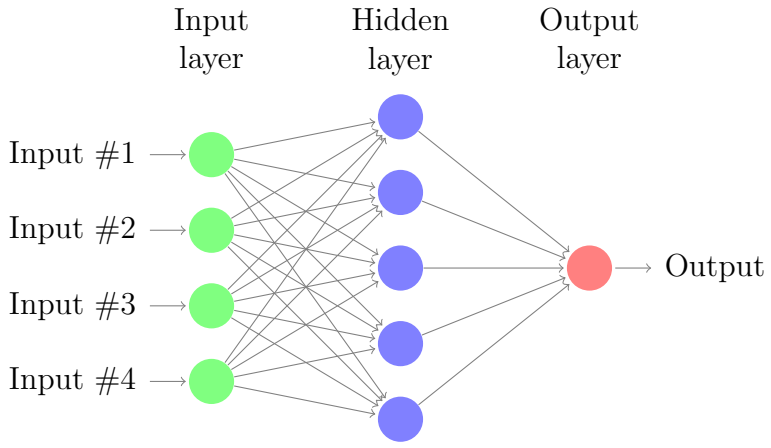
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}log(h_\theta(x^{(i)})) + (1 - y^{(i)})log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

6 Neural Networks

Neurons are computational units that take inputs that are channeled to outputs. x_0 of the input node is known as the bias unit. We use the logistic function as the activation function and θ's are weights. The hidden layers are defined as:

- $a_i^{(j)}$ = activation unit of layer j
- $\theta^{(j)}$ = matrix of weights controlling function mapping from layer j to $j+1$

If the network has s_j units in layer j and s_{j+1} units in layer $j+1$, then $\theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$, the $+1$ coming from the bias nodes.



$$
\begin{aligned}
a_1^{(2)} &= g\big(\theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3\big) \\
a_2^{(2)} &= g\big(\theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \theta_{23}^{(1)}x_3\big) \\
a_3^{(2)} &= g\big(\theta_{30}^{(1)}x_0 + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3\big) \\
h_\theta(x) = a_1^{(3)} &= g\big(\theta_{10}^{(2)}x_0 + \theta_{11}^{(2)}x_1 + \theta_{12}^{(2)}x_2 + \theta_{13}^{(2)}x_3\big)
\end{aligned}
$$

To vectorize we let $z_k^{(2)} = \theta_{k0}^{(1)}x_0 + \theta_{k1}^{(1)}x_1 + \theta_{k2}^{(1)}x_2 + ... + \theta_{kn}^{(1)}x_n$ such that $a_k^{(2)} = g(z_k^{(2)})$ and $x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$, $z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \vdots \\ z_n^{(j)} \end{bmatrix}$

When dealing with multiclass classification rewrite the classes as binary vectors of length equal to the number of classes.

6.1 Backpropagation

Given training set $(x^{(i)}, y^{(i)}), (x^{(m)}, y^{(m)}))$, set $\Delta_{ij}^{(l)} = 0 \, \forall \, l, i, j$ then

$$\begin{aligned}
&\text{for } i = 1 \, to \, m \\
&\text{set } a^{(i)} = x^{(i)} \\
&\text{perform forward propagation to compute } a^{(l)} \\
&\text{using} y^{(i)}, \text{compute} \delta^{(L)} = a^{(L)} - y^{(i)} \\
&\text{compute} \delta^{(L-1)}, \delta^{(L-2)}, ..., \delta^{(L)} \\
&\Delta_{ij}^{(2)} = \Delta_{ij}^{(2)} + a_j^{(2)} \delta_i^{(l+1)} \\
&D_{ij}^{(2)} = \frac{1}{m} \Delta_{ij}^{(2)} + \lambda \theta_{ij}^{(2)} if j \neq 0
\end{aligned}$$

where L is the total number of layers and $a^{(L)}$ is the vector of outputs of activation units for last layer.

6.2 Constructing a network

- set number of input and outputs (number of classes)
- pick number of hidden units per layer
- default to 1 hidden layer and if more than 1 recommend to have same number of units

6.3 Training a network

- randomly initialize weights
- perform forward propagation to get $h_\theta(x^{(i)})$ for any $x^{(i)}$
- implement backpropagation to compute partial derivatives of cost function
- use gradient checking to confirm backpropagation
- use gradient descent or optimization function to minimize cost function

A neural network with fewer parameters is prone to underfitting and is also computationally cheaper. A large neural network with more parameters is prone to overfitting and computationally expensive. In that case you can use regularization to address the overfitting.

7 Decision Trees

7.1 Regression Trees

Grow tree until some minimum node size, then prune tree using cost complexity pruning. Let $|T|$ denote the number of terminal nodes in T. Let

$$\hat{c}_m = \frac{1}{N_m} \sum y_i$$
$$Q_m(T) = \frac{1}{N_m} \sum (y_i - \hat{c}_m)^2$$

then the cost complexity criterion is

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|$$

Want to find for each α the subtree to minimize $C_\alpha(T)$. The tuning parameter $\alpha \geq 0$ governs the tradeoff between tree size and its goodness of fit to the data. Large values of α result in smaller trees.

7.2 Classification Trees

Let

$$\hat{p}_{mk} = \frac{1}{N_m} \sum I(y_i = k)$$

be the proportion of class k observations in node m. Classify the observations in node m to class $k(m) = argmax_k \hat{p}_{km}$, the majority class in node m. Different measures $Q_m(T)$ of node impurity include the following:

Figure 1: Learning curve with respect to polynomial degree

- Misclassification error: $1 - \hat{p}_{mk(m)}$
- Gini index: $\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
- Cross entropy: $-\sum_{k=1}^{K} \hat{p}_{mk} log \hat{p}_{mk}$

Cross entropy and the Gini index are differentiable and more amenable to numerical optimization. They are also more sensitive to changes in the node probabilities than the misclassification rate. For this reason, they should be used when growing the tree. To guide cost-complexity pruning, misclassification rate is typically used.

8 Evaluating a Hypothesis

To start, split the dataset into training and test set. Further split into training, cross validation, and test sets using cross validation. High bias is underfitting and high variance is overfitting. Training error will tend to decrease as the degree d of the polynomial increases. Cross validation error will tend to decrease as we increase d to a point and then will increase. This is visualized in Figure 1.

8.1 Picking λ for regularization

- Pick a range of λ's and a set of models and learn θ
- Compute training error $J_{training}(\theta)$ with $\lambda = 0$
- Compute cross validation error $J_{CV}(\theta)$ with $\lambda = 0$
- Choose model with lowest J_{CV}

8.2 Learning Curves

If experiencing high bias:

- Low training set size: causes $J_{train}(\theta)$ to be low and $J_{CV}(\theta)$ to be high

Figure 2: Learning curve with respect to dataset size

- large training set size: both J_{train} and J_{CV} to be high
- more data won't help high bias

If experiencing high variance:

- Low training set size: J_{train} low, J_{CV} high
- Large training set size: J_{train} increase with n, J_{CV}
- More data will help high variance

9 Support Vector Machines

With support vector machines we want to find the line that best splits two classifications such that we choose the line with the largest margin between the two classes. The cost function:

$$\min_{\theta} C \sum_{i=1}^{m} \Big[y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \Big] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

Hypothesis:

$$h_\theta \begin{cases} 1 & \theta^T x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

9.1 Kernels

Compute features from similarity function, kernels, the data and some arbitrary landmark point.

Gaussian Kernel:

$$f = k(x, l^{(i)}) = exp\Big(- \frac{||x - l^{(1)}||^2}{2\sigma^2} \Big)$$

where the numerator is the Euclidean distance between x and l. If x is far from l then $f \approx 0$. If $x \approx l$ then $f \approx 1$.

Choose initial landmarks at the same points as the training examples. Then map the landmarks (training examples) to features using the kernel function. Group the features into a vector:

$$x \in R^{n+1} => f \in R^{m+1}$$
$$\text{Predict } y = 1 \; if \; \theta^T f \geq 0$$

$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T f^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T f^{(i)}) \right] + \frac{1}{2} \theta^T M \theta$$

A large C will yield lower bias, higher variance, a small C will yield higher bias, lower variance. Large θ^2 then the features f_i vary more smoothly giving higher bias, lower variance.

If you have a large number of features relative to the number of training examples use logistic regression. If n is small and m is intermediate of size, then go with the SVM with Gaussian kernel.

10 K-mean clustering

Pick K cluster centroids for the K number of desired clusters. Assign points to the closest centroids. Move the centroid to the mean of the assigned points. Repeat assignment and move until convergence.

10.1 Objective

Use the norm to get the distance between centroid and data point:

$$\|x^{(i)} - \mu_k\|$$

Where we find the index of the cluster centroid by finding the minimized L2 norm:

$$\min_k \|x^{(i)} - \mu_k\|^2$$

The optimization objective is:

$$J(c^{(1)}, ..., c^{(m)}, \mu_1, ..., \mu_k) = \frac{1}{m} \sum_{i=1}^{m} \|x^{(i)} - \mu_{c(i)}\|^2$$

10.2 Random Initialization

Should have $K < m$ training examples. Randomly pick K training examples and set $\mu_1, ..., \mu_k$ equal to these examples. Different initializations can lead to different solutions so run X amount of times and choose the one with smallest cost.

10.3 Choosing the number of clusters

Can choose based on visual inspection. Elbow method where you run with a range of values for k, plot the cost, and choose the K where it kinks (elbow). Might end up with a plot that is smooth and has no clear kink. Choose based on how this clustering will be used downstream.

11 Dimensionality Reduction

11.1 Principal Component Analysis

PCA attempts to reduce the dimensionality by fitting the data to a lower dimensionality surface that minimizes the sum of square distances from the original plane to the new plane. For a 2-dimensional dataset, the goal is to reduce from the 2-dimension to a 1-dimension, find a direction (vector) onto which to project the data so as to minimize the projection error. For an n-dimensional dataset, reduce from n-dimension to k-dimension, find k vectors onto which to project the data, so as to minimize the projection error. Projecting onto the linear subspace defined by these vectors.

When we project the data x to z, via the linear transformation W, the resulting representation has a diagonal covariance matrix which immediately implies that the individual columns of z are mutually uncorrelated. This ability of PCA to transform data into a representation where the elements are mutually uncorrelated is a very important property of PCA. It is a simple example of a representation that attempt to disentangle the unknown factors of variation underlying the data. In the case of PCA, this disentangling takes

the form of finding a rotation of the input space that align the principal axes of variance with the basis of the new representation space associated with z. While correlation is an important category of dependency between elements of the data, we are also interested in learning representation that disentangle more complicated forms of feature dependencies. For this, we will need more than what can be done with a simple linear transformation.

$$\begin{aligned}
Var[x] &= \frac{1}{n-1}X^TX \\
&= \frac{1}{n-1}(U\Sigma W^T)^TU\Sigma W^T \\
&= \frac{1}{n-1}W\Sigma U^TU\Sigma W^T \\
&= \frac{1}{n-1}W\Sigma^2W^T \\
z &= Wx \therefore \\
Var[z] &= \frac{1}{n-1}Z^TZ \\
&= \frac{1}{n-1}W^TX^TXW \\
&= \frac{1}{n-1}WW^T\Sigma^2WW^T \\
&= \frac{1}{n-1}\Sigma^2
\end{aligned}$$

where W and U are composed of orthonormal vectors $U^TU = I$.

Start by applying feature scaling/mean normalization. Then compute the covariance matrix and it's eigenvectors. The eigenvectors will be a matrix of size $n \times n$ and we chose the first k vectors, U_{reduce}. We compute the lower dimensional dataset:

$$z = U_{reduce}^T x$$

12 Ensemble Methods

Ensemble methods use multiple learning algorithms to obtain better predictive performance than any of the models alone.

12.1 Bagging

Bagging, formerly known as bootstrap aggregating, involves having each model in the ensemble have equal weight. It trains each model on a random sample from the training set (usually uniform with replacement). Reduces variance.

12.2 Random Forest

Sample N cases at random with replacement to create a subset of the data. At each node, select a random subset of features and choose the predictor that provides the best split. The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the repsonse variable, these features will be selected in many of the B trees, causing them to become correlated.

12.3 Boosting

Boosting takes weak models and combines them into one strong model. Reduces bias.

12.3.1 Adaboost

Adaptive boosting is boosting giving a higher weight to stronger models. Each weak classifier is trained on a random subset of the total training set. Each sample gets a weight determining if it should be included in the training set. After training each classifier, increase the weight on the misclassified samples so that they make a larger part of the next classifier. Each classifier is given a weight depending on its accuracy. A classifier with 50% accuracy is given zero weight, and less than 50% is given negative weight.

12.3.2 Gradient Boosting Machine

GBM adds weak classifiers together where subsequent classifiers are regressed on the residuals of the previous classifier.

13 Variable/Model Selection

13.1 z-score

To test the hypothesis that a particular coefficient $\hat{\beta}_j = 0$ we get the standardized coefficient or Z-score:

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{v_j}}$$

where v_j is the jth diagonal element of $(X^T X)^{-1}$. The z_j is distributed under the t-distribution and a large value of z_j will lead to reject the null hypothesis.

13.2 F score

To compare two models with a different number of dependent variables, we can use the F score to determine if the coefficients of the additional variables can be set to zero.

$$F = \frac{(RSS_0 - RSS_1)/(p_1 - p_0)}{RSS_1/(N - p_1 - 1)}$$

13.3 R^2

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. The definition of R-squared is fairly straight-forward; it is the percentage of the response variable variation that is explained by a linear model.

$$R^2 = 1 - \frac{RSS(f)}{RSS(y)}$$
$$R^2 = 1 - \frac{\sum(f_i - \bar{y})}{\sum(y_i - \bar{y})x`x`}$$

13.4 Akaike Information Criterion (AIC)

The AIC offers a relative estimate of the information lost when a given model is used to represent the process that generates the data. In doing so, it deals with the trade-off between the goodness of fit of the model and the complexity of the model.

Let $\hat{L}$ be the maximized values of the likelihood function for the model; $\hat{L} = P(x|\hat{\theta}, M)$, then the AIC value of the model is:

$$AIC = 2k - 2ln(\hat{L})$$

where k is the number of estimated parameteres in the model.

13.5 Bayesian Information Criterion (BIC)

Similar to the AIC but also includes n, the number of data points in x:

$$BIC = ln(n)k - 2ln(\hat{L})$$

13.6 Subset Selection

13.6.1 Best subset regression

Finds for each $k \in 0, 1, 2, ..., p$ the subset of size k that gives the smallest residual sum of squares.

13.6.2 Forward stepwise selection

Starts with the intercept and sequentially adds into the model the predictor that most improves the fit.

13.6.3 Backward stepwise selection

Starts with the full model and sequentially deletes predictors, typically using the F-ratio to choose the predictor to delete.

13.7 Shrinkage Methods

13.7.1 Ridge Regression

See regularization. Shrinks all directions, but shrinks low-variance directions more.

13.7.2 Lasso

Lasso is similar to ridge regression except penalizing by the L1 norm of the estimators.