CS 229 Stanford

Ch1 Supervised Learning

1. **Gradient descent**: a search algorithm that starts with some “initial guess” of parameters, and that repeatedly changes parameters to make cost function smaller.
2. **Batch gradient descent** : In a single step, take entire training set to update parameters.

**Stochastic (Incremental) gradient descent** : BGD has to scan through the entire training set, which makes a costly operation when data size is large.

1. **To preform supervised Learning, we have to decide functions/hypothesis that map X to Y.**
2. **Linear Regression**
   1. 1st way : Set Cost function as MSE using Gradient Descent
   2. 2nd Way : Given MSE, we can apply Normal Equations to represent parameters mathematically.
   3. Why MSE for regression problem?
      1. Probabilistic Interpretation:
         1. Assumption: Y = Theata^T \* X + error , Error ~IID N(0,. Sigma^2)
         2. P(Y|X ; theta) = normal distribution and it can be regarded as likelihood function.
         3. Maximize log likelihood 🡨🡪 minimize MSE
   4. Locally Weighted Linear Regression (Non-parametric algorithm)
      1. If there is sufficient data, this makes the choice of features less critical.
      2. Weight is exp( - | x\_i - x | ) and x is the point to evaluate, so if difference is small, w is close to 1, and if the difference term is large, w will be small.
      3. Therefore, parameters are chosen giving a much higher “weight” to the errors on training examples close to the query point.
      4. Parametric vs. Non-Parametric
         1. While parametric learning algorithm has fixed, finite number of parameters, and after we’ve fit the parameters, training data is no more needed. Non-Pararmetric way means the amount of stuff we need to keep in order to represent the hypothesis h grows exponentially with the size of the training set.
3. **Classification and Logistic Regression**
   1. While linear regression takes h(x) as Theta\*X , Logistic regression takes h(x) as 1/(1+e^(-theta\*X)), non-linear function of Theta\*X , called logistic function or sigmoid function.
   2. Probabilistic way: Log-likelihood maximization
      1. P(y=1 | x; theta) = h(x), P(y=0 | x; theta) = 1 – h(x)
      2. P(y | x, theta) = h(x)^y \* (i-h(x))^(1-y ) using Bernoulli Distribution.
      3. By taking cost function as Likelihood function and set :



Same update rule with linear regression problem.

**Newton’s Method**

**Although it requires fewer iterations to get value close to minimum, it’s more expensive than gradient descent ( because it requires us to calculate n by n Hessian )**

**When Newton’s method is applied to maximize the logistic regression log-likelihood, resulting method is called Fisher Scoring**

When the goal is to find parameter theta 🡪 F(theta) = 0



What if the goal is to maximize function l? 🡪 points where its first derivative l’(theta) = 0. F(theta) = l’(theta)

🡪 C:\Users\09732\AppData\Local\Microsoft\Windows\INetCache\Content.Word\캡처2.png



**Generalized Linear Models**

Classification using logistic regression or perceptron algorithm 🡪? Find a decision boundary

**Discriminative** : Learn P(Y|X) directly or learn mapping directly from the space of input X to the labels

**Generative** : model P(X|Y) 🡪 model the distribution of a label’s feature.

After modeling p(y) – class priors, and p(x|y), we can apply Bayes rule to derive the posterior distribution on y given x.

1. **Gaussian Discriminant Analysis**
   1. **Assumption : P(X|Y) ~ multivariate normal distribution**
   2. **Compressed vs. Spread-out**
   3. **P(X|Y) is multivariate Gaussian 🡪 P(Y|X) follows a logistic function. But, not the other way around. : GDA makes stronger Assumption.** 
      1. **When such modeling assumption(P(X|Y) is Gaussian) is correct, GDA performs better**
      2. **With weaker assumptions, logistic regression is more robust and less sensitive to incorrect modeling assumptions. (Possion data (Non-Guassian data )🡪 Logistic Ok, but not for GDA )**
2. **Naïve Bayes**
   1. **X : discrete**
   2. **Example : X: Text, Y: Spam or Not (Text Classification)**
      1. **Represent an email via a feature vector whose length is equal to the number of words in dictionary**
      2. **Build a generative model : P(X|Y) 🡪 2^50000 possible outcomes.**
      3. **Therefore, make an assumption that observations of X are conditionally independent given y. (Naïve Bayes Assumption), which results Naïve Bayes Classification 🡪 e.g. knowledge of word “buy” doesn’t affect your beliefs about the word “price” occurrence.**
   3. **When the original, continuous-valued attributes are not modeled by a multivariate normal distribution, discretizing the features and using Naïve Bayes will often result in a better classifier.**
   4. **Laplace Smoothing** 
      * 1. **In a binary classification(spam/non-spam), when a new word used as a testing data, but not used in training set, the prediction of the new word results in 0/0. To avoid such unusual cases, the maximum likelihood of estimater, which is fraction of each class over the number of observations, is changed by adding 1 to numerator and number of class to denominator.**

**Part 5: SVM**

**H(x) = g(w\*x + b), g(z) = 1 if z >=0 , -1 otherwise.**

**Goal : Separate the positive and negative training examples with a gap (geometric margin)**

1. **Functional margin vs. Geometric margin ( Optimal margin classifier )**
   1. **Optimization Problem**
      1. **Objective function : Max gamma (functional margin)**
      2. **Constraint: y(w\*x+b) >= gamma and ||w|| = 1(functional margin == geometric margin)**

**🡪 Convert functional margin to geometric margin, which allows us to add an arbitrary scaling constraint on w and b w/o changing anything.**

**Objective function : Max gamma\_hat / ||w||**

**Constraint : y(w\*x+b)>= gamma\_hat (geometric margin)**

**🡪 Lagrange duality**

**Objective function : min 0.5||w||^2**

**Constraint : y(w\*x+b) >= 1**

**Now, it’s an optimization problem with a convex quadratice objective and linear constraints. It’s solution is optimal margin classifier.**

**The above optimization can be described by Lagrangian with alpha(lagrange multiplier)**

* **Support vector : In a maximum margin separating hyperplane, the points with the smallest margins (the ones closest to the decision boundary). In Lagrangian optimization, non-zero lagrangian multipliers.**
* **After solving Lagrangian duality, w\*x + b = ( sum\_alpha\*y\*x^i )^T \*x + b**

**= sum\_alpha\*y<x^i, x> +b**

**Therefore, we can rewrite the entire algorithm in terms of inner products between input features.**

**Kernels (let SVM to learn in the high dimensional feature space)**

**Attribute: original input value**

**Phai : feature mapping, which maps from the attributes to the features.**

**Valid(Mercer) Kernel : Kernel Matrix is symmetric positive semi-definite.**

**Regularization**

**Since it’s still susceptible to outliers, we can reformulate the optimization with regularization.**

**Part 6: Learning Theory**

**Training error (empirical risk or error) : fraction of trainging examples that h misclassifies.**

**Empirical Risk Minimization (ERM) : Minimize the training error.**

**e.g. Logistic regression : Approximations to empirical risk minimization.**

**Perceptron and Large margin classifiers**

* **Batch Learning : learning on entire training data set**
* **Online Learning : It is used when it is computationally infeasible to train over the entire dataset, and data is sequential that enables us to update best predictor for future data at each step. (theta = theta + y\*x)**
* **Algorithm has to make predictions continuously even while it’s learning.**

**K-means**

1. **Un-supervised learning**
2. **Methods**
   1. **Initialize cluster centroids (k) randomly**
   2. **Repeat 2 procedures ( 1. Assigning each training example to the closest cluster centroids, 2. Moving each cluster centroid to the mean of the points assigned to it ) until convergence**
3. **Time Complexity : O(n\*k\*t) : iterations, clusters, data points**
4. **Cost function : J(c,mu) = sum\_i=1toM || x^i – mu\_c^i ||^2.** 
   1. **Repeatedly minimizes J w.r.t. c while holding mu fixed.**
   2. **And then, minimizes J w.r.t. mu while holding c fixed.**
   3. **Therefore, J must monotonically decrease, and the value of J must converge.**
   4. **Since J is non-convex function, and so coordinate descent on J is not guaranteed to converge to global minimum. 🡪 K-means is susceptible to local minima.**
   5. **Solutions : Run K-means multiple times, and choose one that gives the lowest distortion**
   6. **Assumptions** 
      1. first that the clusters are spherical and second that the clusters are of similar size.
      2. Non-Spherical data 🡪 doesn’t work well

**Solution 1 (Using K-means): Transform our Cartesian (x,y) coordinate to polar(arc vs. radius) coordinate. Such data will be completely partitioned by their radius.**

**Solution 2 (Using DBSCAN) : 1. Clusters are constructed based on the density, geometrically featured data can be clustered well. 2. No need to set the number of clusters. 3. Possible to find outlier using noise point.**

**What is DBSCAN?**

**It recognizes cluster if there exist minimum number of points(minpts) within distance(epsilon) from a certain point. If a point constructs a border line, then it’s called border point. If a point is not involved in any cluster, then it’s noise point**

**Input : distance of a certain point to cluster edge(epsilon) and (minPts)**

* DBSCAN does not require the analyst to select the number of clusters a priori — the algorithm determines this based on the parameters it's given.
* It excels at clustering non-spherical data.
* It does not force every data point into a cluster — if a data point lies outside of the specified starting parameters, it will be classified as "noise" and not included in any resulting cluster.
* An evolved version of DBSCAN, called **"HDBSCAN"** (the **H** for "hierarchical"), attempts to allow for clusters of differing variances and densities. [HDBSCAN](http://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html) really only requires us to provide one parameter: minimum cluster size. 🡪 Varying density clusters