Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 2

Student Name: Ajita Shree
Roll Number: 20111262
Date: November 27, 2020

QUESTION 1

Solution

Second Order Optimization for Logistic Regression Logistic Regression model with label $y_n \in \{0, 1\}$ has loss function as $L(w) = -\sum_{n \in N}(y_n w^T x_n - log(1 + exp(w^T x_n)))$

Prove: Second order optimization based update equation $w^{(t+1)} = w^t - H^{(t)-1} g^{(t)}$, where H is Hessian and g is gradient, reduces to solving importance weighted regression problem of the form $w^{(t+1)} = argmin_w \sum_{n \in N} \gamma_n^{(t)} (y_n'^{(t)} - w^T x_n)^2$ where γ_n is the importance of nth training example y_n' denote modified real valued label.

- **Gradient**, $\frac{dL(w)}{dw}$ can be written as $-\sum_{n \in N} y_n x_n + \sum_{n \in N} \frac{exp(w^T x_n) x_n}{1 + exp(w^T x_n)}$
- **Hessian**, $\frac{d^2 L(w)}{dw^2} = \sum_{n \in N} \frac{exp(w^T x_n) x_n^T x_n}{(1 + exp(w^T x_n))^2}$
- Substituting the values in the **Update Equation**,

$$w^{(t+1)} = w^t + \sum_{n \in N} \frac{(1 + exp(w^T x_n))^2}{exp(w^T x_n)} (x_n^T x_n)^{-1} \left\{ y_n x_n - \frac{exp(w^T x_n) x_n}{1 + exp(w^T x_n)} \right\}$$

- Taking H^{-1} term out of the equation

$$w^{(t+1)} = \frac{(1 + exp(w^T x_n)^2}{exp(w^T x_n)} (x_n^T x_n)^{-1} \left\{ \frac{exp(w^T x_n)}{(1 + exp(w^T x_n))^2} (x_n^T x_n) w^t + y_n x_n - \frac{exp(w^T x_n) x_n}{1 + exp(w^T x_n)} \right\}$$

- On further simplification,

$$w^{(t+1)} = \frac{(1 + exp(w^T x_n))^2}{exp(w^T x_n)} (x_n^T x_n)^{-1} \left\{ \frac{exp(w^T x_n)}{(1 + exp(w^T x_n))^2} w^{tT}(x_n) + y_n - \frac{exp(w^T x_n)}{1 + exp(w^T x_n)} \right\} x_n$$

- Alternatively, using $\sigma(w^T x) = \frac{exp(w^T x_n)}{1 + exp(w^T x_n)}$, we have

$$w^{(t+1)} = \frac{(1 + exp(w^T x_n)}{\sigma(w^T x_n)} (x_n^T x_n)^{-1} \left\{ y_n - \sigma(w^T x_n) \{ \frac{w^{tT}(x_n)}{(1 + exp(w^T x_n))} - 1 \} \right\} x_n \qquad (1)$$

- Considering **Importance weighted regression problem**,
- $w^{(t+1)} = argmin_w \sum_{n \in N} \gamma_n^{(t)} (y_n'^{(t)} - w^T x_n)^2$
- Differentiating above equation and equating it to 0, we have

$$w = (\sum_{n \in N} \gamma_n x_n^T x_n)^{-1} (\sum_{n \in N} y_n' x_n) \tag{2}$$

- **Comparing equation 1 and equation 2, we will get the following values**
- $\gamma_n = \frac{\sigma(w^T x_n)}{(1 + exp(w^T x_n)}$; Also,can be written as $\gamma_n = \sigma(w^T x_n)(1 - \sigma(w^T x_n))$ (alternatively, called constant term in the Hessian matrix).
- $y' = y_n - \sigma(w^T x_n)\{\frac{w^{tT}(x_n)}{(1 + exp(w^T x_n))} - 1\}$
- The γ_n makes intuitive sense here because
  - We know, in logistic regression, $\mu_n = \sigma(w^T x)$ and likelihood function for a given point x_n can be defined by $Bernoulli(\mu_n)$ with form $Bernoulli(\mu_n) = \mu_n^{y_n}(1 - \mu_n)^{(1-y_n)}$ and γ has a similar form.
  - More likely/important points will get more importance while predicting real-valued y'in the regression problem because of this form of γ.

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 2

Student Name: Ajita Shree
Roll Number: 20111262
Date: November 27, 2020

QUESTION 2

Solution

Perceptron with Kernels Perceptron updates have the form $w = w + y_n x_n$, the weight learned by perceptron can be written as $w = \sum_{n \in N} \alpha_n y_n x_n$ where α_n is the number of times perceptron makes a mistake on example x_n

Goal Enable perceptron to learn non-linear boundaries, using kernel k with feature map ϕ. Give the kernalized variant of perceptron algorithm.

Kernel perceptron algorithm is stochastic gradient descent on the perceptron kernel loss.Let ϕ defines the kernel mapping for data point x and kernel function $K(x_m, x_n) = \phi(x_m)^T \phi(x_n)$ that is guaranteed to be always in the finite dimension.

- Step 1: **Initialization**: Initialize w = $\sum_{n \in N} \alpha_n y_n x_n$ and $\alpha_m = 0 \forall m \in N$.
- Step 2: Pick a point (x_m, y_m) at random
- Step 3: **Mistake Condition**: $y_m w^{(t)T} \phi(x_m) < 0$
  - $\sum_{n \in N} (y_m \alpha_n y_n \phi^T(x_n) \phi(x_m)) < 0$
  - $\sum_{n \in N} (y_m \alpha_n y_n K(x_n, x_m) < 0$
- Step 4: **Update Equation** $\alpha_m + = 1$ if mistake condition holds true.
- Step 5: Go to step 2 until no converged.

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 2

Student Name: Ajita Shree
Roll Number: 20111262
Date: November 27, 2020

QUESTION 3

Solution

SVM with unequal class importance The primal formulation of SVM with unequal class importance is given by the following equation

$$min_{w,b,\zeta} \frac{||w||^2}{2} + \sum_{n \in N} C_{y_n} \zeta_n$$
$$s.t. y_n(w^T x_n + b) >= 1 - \zeta_n$$
$$\zeta_n >= 0, \forall n$$

- **Lagrangian Problem of this modified SVM is**

$$min_{w,b,\zeta} max_{\alpha >= 0, \beta >= 0} L(w, b, \zeta, \alpha, \beta)$$
$$= \frac{||w||^2}{2} + \sum_{n \in N} C_{y_n} \zeta_n + \sum_{n \in N} \alpha_n (1 - y_n(w^T x_n + b) - \zeta_n) - \sum_{n \in N} \beta_n \zeta_n$$

- Differentiating w.r.t. w, b, ζ_n, C_{y_n}, we get
  - setting $\frac{dL}{dw} = 0$, we get w = $\sum_{n \in N} \alpha_n y_n x_n$
  - setting $\frac{dL}{db} = 0$, we get $\sum_{n \in N} \alpha_n y_n = 0$
  - setting $\frac{dL}{d\zeta_n} = 0, weget \mathrm{C}_{y_n} - \alpha_n - \beta_n = 0$ and we know that $\beta_n >= 0$. We will have $\alpha_n <= C_{y_n}$
  - setting $\frac{dL}{dC_{y_n}} = 0$, we get $\zeta_n = 0$

- Substituting all the values in the Lagrangian equation, we get

$$1/2 \sum_{m,n \in N} \alpha_m \alpha_n y_m y_n (x_m^T x_n) + \sum_{n \in N} (\alpha_n \zeta_n + \beta_n \zeta_n) + \sum_{n \in N} \alpha_n - \sum_{m,n \in N} \alpha_m \alpha_n y_m y_n (x_m^T x_n)$$
$$+ \sum_{n \in N} \alpha_n y_n b - \sum_{n \in N} \alpha_n \zeta_n - \sum_{n \in N} \beta_n \zeta_n$$

- The final form of the equation is:

$$max_{\alpha_n <= C_{y_n} : \forall n \in N} \sum_{n \in N} \alpha_n - 1/2 \sum_{m,n \in N} \alpha_m \alpha_n y_m y_n (x_m^T x_n)$$
$$s.t. \sum_{n \in N} \alpha_n y_n = 0$$
$$\zeta_n = 0, \forall n \in N$$

- Intuitively, the key difference between this SVM and standard SVM is as follows:
  - The dual formulation of this SVM has an additional constraint on ζ_n i.e. $\zeta_n = 0$ which is making this formulation similar to that of hard margin SVM conceptually.
  - However, the dual formulation is exactly matching with soft margin SVM Only **difference is that here nth term of α is constraint by the respective C_{y_n} where as in standard SVM all α terms are upper bounded by constant c**
  - Intuitvely, if nth example is +, only possible prediction are - and +. + would be correct prediction and - would fall under the category of mis-classification of positive examples. Hence, it makes more sense if α_n is bounded the C_+. Same logic applied for - class.

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 2

Student Name: Ajita Shree
Roll Number: 20111262
Date: November 27, 2020

QUESTION 4

Solution

SGD for k-means objective The kmeans objective is given by $\sum_{n\in N}\sum_{k\in K} z_{nk}||x_n - \mu_k||^2$. Assume $\mu_{k\in K}$ are initialized randomly at the beginning of the algorithm. K-means can be made online by using following steps: 1) Taking random example x_n at a time and greedily assign it to best cluster.2) Updating the cluster means using SGD on objective L.

- **How to solve step 1?**
  - Randomly select point $x_{n'}$ from the data points (**It can be a point that is not assigned to any cluster or in case of no new data points, it can be any point that is already being assigned to a cluster**)
  - Select the cluster greedily k' for $x_{n'}$ which is the closest i.e. $k' = min_k||x_{n'} - \mu_k||^2$
  - Update z matrix for $x_{n'}$
- What will be the SGD based cluster mean update equations for step 2? Intuitively, why does the update make sense.
  - **SGD for updating the mean $\mu_{k'}$**

$$\frac{dL}{d\mu_{k'}} = \frac{d\sum_{n\in N} z_{nk'}||x_n - \mu_{k'}||^2}{d\mu_{k'}}$$

$$\frac{dL}{d\mu_{k'}} = -2\sum_{n\in N} z_{nk'}(x_n - \mu_{k'})$$

  - **Update equation:**

$$\mu_{k'+1} = \mu_{k'} + \eta_{k'} 2\sum_{n\in N} z_{nk'}(x_n - \mu_{k'})$$

  - **The update make sense because** the gradient term signifies the sum of euclidean distance of the current assignment of data points to that cluster. Given right η term, it will be exactly equal to the mean formula (typically used in k-means algorithms to compute the mean).
- **SGD requires a step size.** For your derived SGD update, suggest a good choice of step size (mention why it is a good choice).
  - In each iteration of SGD, cluster assignment of only one point x'_n is changed as per the algorithm. Hence, $n_{k'}$ can be selected based on following scenarios.

* **Case 1: $x_{n'}$ is added to k' cluster**

$$\eta_{k'} = \frac{1}{2\sum_{n \in N} z_{nk'}} - \frac{\sum_{n \neq n'} z_{nk'}(x_n - \mu'_k)}{2\sum_{n \neq n'} z_{nk'}}$$

First term of the $\eta_{k'}$ will ensure the **mean calculation for the updated assignments** and **second term** will be the **mean calculation for previous assignment**(will be canceled out by $\mu_{k'}$ in the update equation). Note that,in case of new data points, previous assignment will automatically not be considered as z_n vector will be all 0.

* **Case 2: $x_{n'}$ is removed from k' cluster**

$$\eta_{k'} = \frac{1}{2\sum_{n \in N} z_{nk'}} - \frac{\sum_{n} z_{nk'}(x_n - \mu'_k) + (x_{n'} + \mu_{k'})}{2(\sum_{n} z_{nk'} + 1)}$$

First term is similar to case 1 denoting mean of cluster with updated assignment; Second term is the mean of cluster when $x_{n'}$ was the part of this previous cluster (2 in the denominator is to cancel terms in update equation). **Note: This case will arise in case of there are no new data points and some already assigned $x_n s$ got picked at random in the iterations.**

* **Case 3: No update needed for cluster k' if the points are neither removed nor added in the cluster**

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 2

Student Name: Ajita Shree
Roll Number: 20111262
Date: November 27, 2020

QUESTION 5

Solution

Kernel K-means: Assume a kernel with infinite dimensional feature map, We can neither store the kernel induced feature set nor can store the cluster means in the kernel induced feature space.

- **Sketch the K-means algorithms** (initialization, cluster assignment and mean computation)
  - The Loss function for k-means is given by $\sum_{n\in N}\sum_{k\in K} z_{nk}||x_n - \mu_k||^2$.
  - The Loss function in the feature induced space can be written as
$$\sum_{n\in N}\sum_{k\in K} z_{nk}||\phi(x_n) - \phi(\mu_k)||^2$$
  - On further solving
$$\sum_{n\in N}\sum_{k\in K} z_{nk}(\phi(x_n)^T\phi(x_n) + \phi(\mu_k)^T\phi(\mu_k) - 2\phi(x)\phi(\mu_k))$$
$$\sum_{n\in N}\sum_{k\in K} z_{nk}K(\phi(x_n),\phi(x_n)) + K(\phi(\mu_k),\phi(\mu_k)) - 2K(\phi(x_n),\phi(\mu_k))$$
  - Substituting the RBF kernal value, $K(x,z) = exp[-\gamma||x-z||^2]$
$$\sum_{n\in N}\sum_{k\in K} z_{nk}(exp(-\gamma||x_n - x_n||^2) + exp(-\gamma||\mu_k - \mu_k||^2) - 2exp(-\gamma||x_n - \mu_k||^2))$$
  - **Loss function for kernal k-means** can be written as
$$L(X,Z,\mu) = \sum_{n\in N}\sum_{k\in K} z_{nk}(-2)exp(-\gamma||x_n - \mu_k||^2))$$
  - The initialization , cluster assignments and mean computation will remain same as k-means function using **ALT-OPT technique over z and μ** to optimize above loss function as follows:
    - **Step1:** Fix μ as μ' find optimal z assignment as $Z' = argmin_z L(X,Z,\mu')$
    - **Step 2:** Fix z as z' find optimal μ as $\mu' = argmin_\mu L(X,Z',\mu)$
    - **Step 3:** Go to step1 until not converged
- **What is the difference between how cluster means are stored in kernel k means vs standard k-means**

  For kernel k means, we need not store kernel induced feature space mean for μ as the loss function got reduced to the form with only x_n and μ_k. We need to store μ_k for k clusters similar to k-means.

- **How does kernel k-means compares with standard k-means in terms of cost of input to cluster mean distance calculations**

  The cost of computing input to cluster mean distance is same as k-means O(NK). (Reason: Kernal k-means loss function is similar to loss function of k-means in terms of computation).

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 2

Student Name: Ajita Shree
Roll Number: 20111262
Date: November 27, 2020

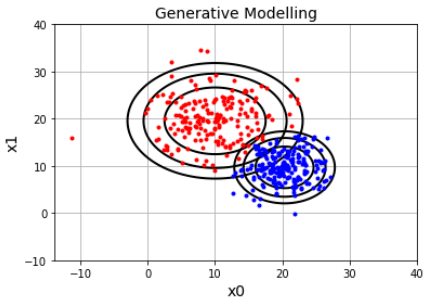
QUESTION 6

Solution - Part 1

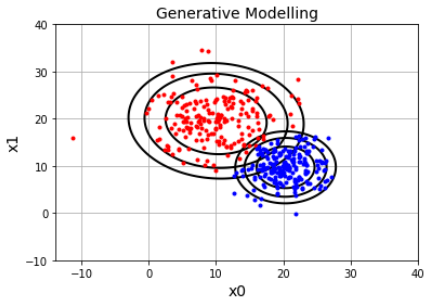
Implement the generative classification model assuming Gaussian class-conditional distributions of positive and negative class examples. Class prior are given as 0.5 in the question and also being verified for the given dataset.

Observation on data: binclass.txt

- On the 2-dimensional plane, the examples and the learned decision boundary using class conditionals to be to be $N(\mu_+, \sigma_+^2 I_2)$ and $N(\mu_-, \sigma_-^2 I_2)$ are as follows

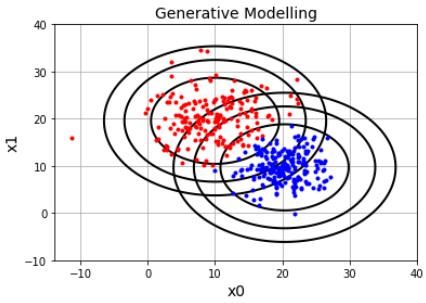


Diagonal Co-variance Matrix

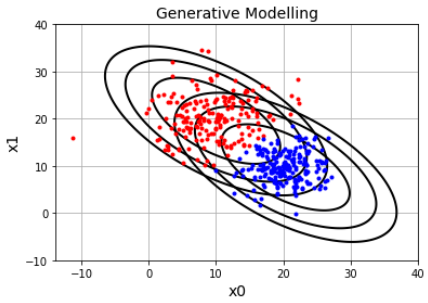


Full Co-variance Matrix

- The Results will be as follows assuming the Gaussian class-conditional distributions of the positive and negative examples to be $N(\mu_+, \sigma^2 I_2)$ and $N(\mu_-, \sigma^2 I_2)$

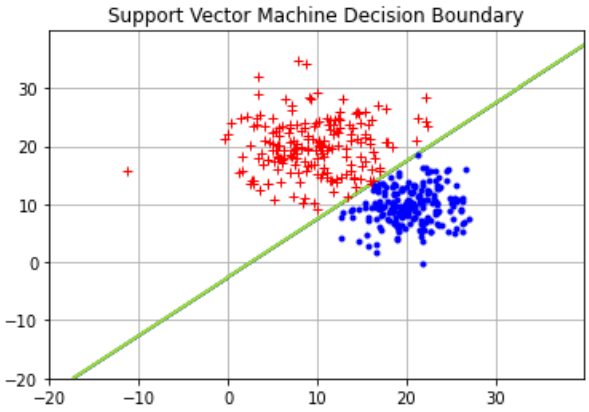


Diagonal Co-variance Matrix



Full Co-variance Matrix

- The decision boundary corresponding to linear SVM ($svm.LinearSVC(C = 1, max_iter = 5000).fit(X, Y)$ from the sklearn library) is as follows:

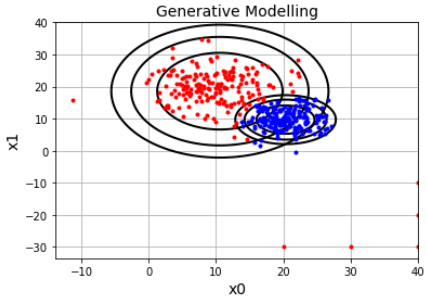


SVM Classifier: Dataset 1

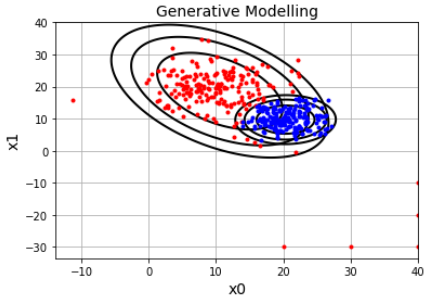
Solution - Part 2

Observation on data: binclassv2.txt

- On the 2-dimensional plane, the examples and the learned decision boundary using class-conditional to be to be $N(\mu_+, \sigma_+^2 I_2)$ and $N(\mu_-, \sigma_-^2 I_2)$ are as follows

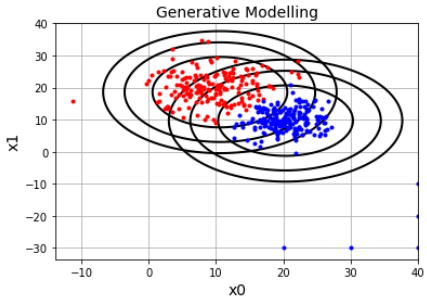


Diagonal Co-variance Matrix

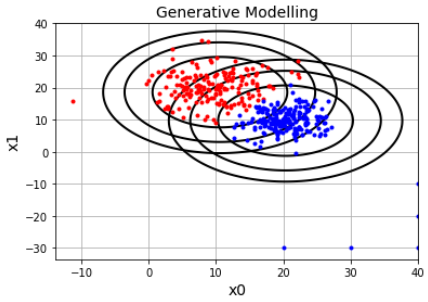


Full Co-variance Matrix

- The Results will be as follows assuming the Gaussian class-conditional distributions of the positive and negative examples to be $N(\mu_+, \sigma^2 I_2)$ and $N(\mu_-, \sigma^2 I_2)$

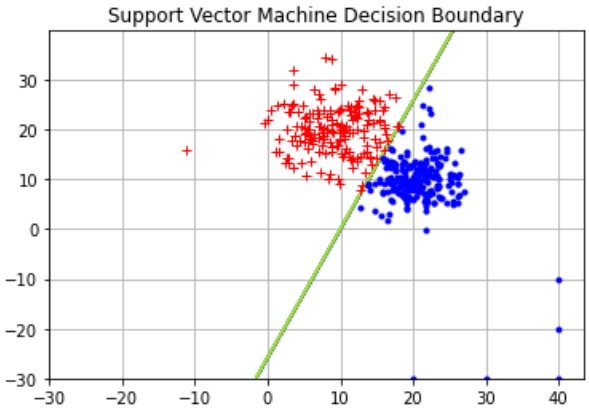


Diagonal Co-variance Matrix



Full Co-variance Matrix

- The decision boundary corresponding to linear SVM ($svm.LinearSVC(C = 1, max_iter = 5000).fit(X, Y)$ from the sklearn library) is as follows:



SVM Classifier Dataset 2

Conclusion: Generative classification vs Linear-SVM

- **Dataset: binclass.txt**
  - $\{\sigma_+, \sigma-\}$ **for both classes**: The contours for both the classes are more compact and there is a nice separation between the positive points (red) class versus the negative points (blue class). The shape of contour is similar in both diagonal co-variance matrix and full co-variance matrix. **The accuracy achieved is 95.25**
  - σ **for both classes:** If σ is learned taking both classes together, the mean differs, the shape and breadth of both the classes matches. **The accuracy achieved is 95**
  - **SVM**: The linear decision boundary is able to separate the points well. **The accuracy achieved is 94.75**
  - As it can be seen that, the accuracy achieved on the train data for both generative classification as well SVM seems to be same.
- **Dataset: binclassv2.txt**
  - $\{\sigma_+, \sigma-\}$ **for both classes**:In this case, significantly smaller contour size is learned for negative samples compared to positive samples. If we take full-co-variance matrix, we can see the diagonal Gaussian is learned for positive class. **The best accuracy achieved is 95.5**
  - σ **for both classes:** As σ is same for both classes, the spread is similar and similar Gaussian is learnt irrespective of diagonal matrix or full covariance matrix. **The best accuracy achieved is 93.75**
  - **SVM**: The linear boundary is shown in the diagram above. **The accuracy achieved through linear SVM is 93.5.**

  Observing results on both the data, it can be concluded that generative classification will be slightly better than linear SVM in case of 2 class classification. **It is also clearly evident that learning different σ_+, sigma for positive class and σ_-, sigma for negative class is able to learn probability distribution better, specially in the situations when spread of the data is very different.**

Implementation:

- The code will do generative classification and can run for multiple features at the time i.e. $D > 0$
- Co-variance matrices capable of handling any shape of Gaussian i.e. all entries non-zero: Line 50-57 can be un commented to visualize this
- The Code can be easily scaled up for multiple classes at a time.
- Input path and file to be provided (main function).