Indian Institute of Technology Kanpur
CS771 Introduction to Machine Learning, 2017-18-a

QUESTION 1

Assignment Number: 2
Student Name: Gurpreet Singh
Roll Number: 150259
Date: October 10, 2017

Part 1

No, name is not a useful attribute in learning a decision tree as splitting on this attribute will have no gain. Non-statistically speaking, there is a lot of variance in the attribute itself, and it will be very inefficient to decide on the basis of name as it can often be a newly observed value.

Part 2

It is not possible to classify the data given perfectly. Looking at data #4 and #6, they have the same values for all fields except for 'name', however have different target values. Since we shall not use 'name' as a decision classifier (as discussed in Part A), therefore we can say that is is not possible to classify the data perfectly.

Part 3

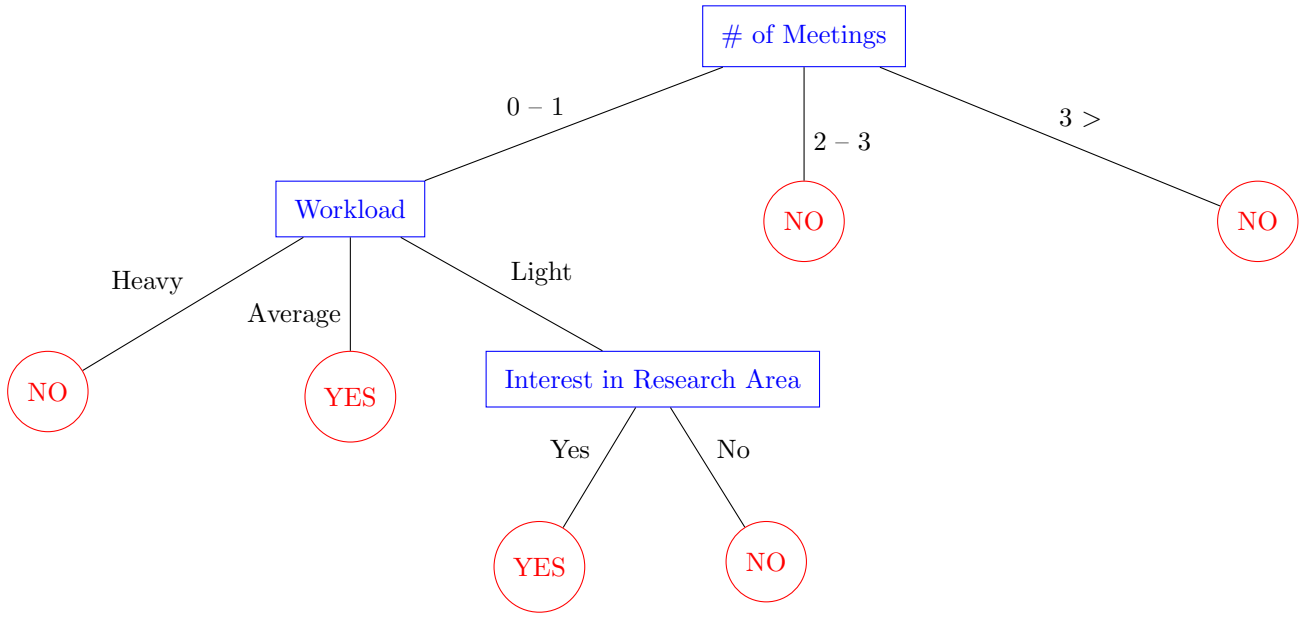


Figure 1: ID3 Tree for the given data

ID3 is a greedy strategy to generate decision trees using Information Gain. The information gain at each level/depth is mentioned below

Level 1

Entropy $S = 0.9183$

IG: Size of Research Group $= S - 0.2600 - 0.2667 - 0.3237 = 0.0679$
IG: Interest in Research Area $= S - 0.2667 - 0.6199 = 0.0317$
IG: Workload $= S - 0.4000 - 0.2163 - 0.2406 = 0.0614$
IG: Number of Meetings $= S - 0.6667 = 0.2516$

Clearly the attribute *Number of Meetings* provides the most IG, and hence we use this attribute to split at the first level.

Level 2 — Number of Meetings

Possible distictions — '$0-1$', '$2-3$', '>3'

Since for '$2-3$' and '>3', all labels are 'No' (**i.e.** entropy is 0), hence we do need to split here, and we add leaf nodes.

For the samples remaining with value '$0-1$' of the attribute *Number of Meetings*, we will further extend the decision tree.

Extropy $S = 1$

IG: Size of Research Group $= S - 0.0000 - 0.4000 - 0.4855 = 0.1145$
IG: Interest in Research Area $= S - 0.2755 - 0.6897 = 0.0348$
IG: Workload $= S - 0.0000 - 0.0000 - 0.3610 = 0.6390$

Clearly the attribute *Workload* gives the highest gain, and hence we use this attribute to further construct the decision tree.

Level 3 — Workload

Possible distinctions — 'Light', 'Average', 'Heavy'

Since there is purity in the samples with 'Average' and 'Heavy' value in the *Workload* attribute (*i.e.* entropy is 0), we do not split further in this case. Therefore we are only left with two samples, which have the value 'Light'.

Entropy $S = 1$

IG: Size of Research Group $= S - 1.0000 = 0.0000$
IG: Interest in Research Area $= S - 0.0000 = 1.0000$

Hence we further split using *Interest in Research Area* attribute. Since there is only one sample per distinction, we cannot split further. Hence we have the decision tree showed in Figure 1

Indian Institute of Technology Kanpur
CS771 Introduction to Machine Learning, 2017-18-a

Assignment Number: 2
Student Name: Gurpreet Singh
Roll Number: 150259
Date: October 10, 2017

QUESTION 2

Notations

L: Represents total number of items

N: Represents total number of users

D: Represents length of feature vector for every user *i.e.* length of $\mathbf{x}^n$

K: 2^L

Part 1

Any user can select any number of items. The probability of selecting an item for a user is represented by a logistic expression.

$$P(\mu_l \,|\, \mathbf{x}^n, \mathbf{w}_l) = \sigma\big(\langle \mathbf{w}_l, \mathbf{x}^n \rangle\big)^{\mu_l} \big(1 - \sigma\big(\langle \mathbf{w}_l, \mathbf{x}^n \rangle\big)\big)^{(1-\mu_l)} \tag{1}$$

Note: $\sigma\big(x\big)$ *represents the sigmoid function*

The equation 1 represents the probability of user n selecting item l *i.e.* $\mu_l = 1$ if item l has been selected, 0 otherwise. We assume that the probability is represented by the same expression for all users and is independent, since all users are independent and are represented by their feature vector $\mathbf{x}^n$.

It is clear that there are K possible subsets of the items. Hence the user can choose any one of the subsets. Let these subsets be denoted by S_k. Hence the powerset of the items set is $\big\{S_k\big\}_{k=1}^{K}$. Now, we can write the expression for a user selecting a subset of items.

$$P(S_k \,|\, \mathbf{x}^n, \big\{\mathbf{w}_l\big\}_{l=1}^{L}) = \prod_{l \in S_k} P(\mu_l = 1 \,|\, \mathbf{x}^n) \prod_{l \notin S_k} P(\mu_l = 0 \,|\, \mathbf{x}^n) \tag{2}$$

Here, $\big\{\mathbf{w}_l\big\}_{l=1}^{L}$ is a parameter for the model.

Now we define our latent variable $z^n \in [K]$. This is the index of the subset of items, in the powerset, that the user n has chosen *i.e.* $z^n = k$ represents the subset S_k. Hence, we can define the conditional probability of z^n as follows

$$P(z^n \,|\, \mathbf{x}^n, \big\{\mathbf{w}_l\big\}_{l=1}^{L}) = \prod_{k=1}^{K} P(S_k \,|\, \mathbf{x}^n, \big\{\mathbf{w}_l\big\}_{l=1}^{L})^{\mathbb{I}\left[z^n = k\right]} \tag{3}$$

We also define a function $h : [K] \to \mathbb{R}$ such that

$$h(z^n) = \sum_{l \in S_{z^n}} c_l \tag{4}$$

We can now write the expression for the conditional probablity of our output variable b^n as follows

$$b^n \,|\, z^n, \sigma \sim \mathcal{N}\big(b^n \,|\, h(z^n), \sigma^2\big) \tag{5}$$

Note: $b^n \,|\, z^n, \sigma \sim b^n \,|\, z^n, \mathbf{x}^n, \sigma$. This is because once we assume a value of z^n, $\mathbf{x}^n$ no longer plays a role in the conditional probability

Note: *We can collectively define* $\sigma, \{\mathbf{w}_l\}_{l=1}^{L}$ *as* $\mathbf{\Theta}$

Part 2

We can represent the Complete Likelihood (CLE) as follows

$$CLE \sim \mathbf{b}, \mathbf{z} \,|\, \mathbf{X}, \mathbf{\Theta}$$

Since all users are independent, we can write the above as follows

$$P(\mathbf{b}, \mathbf{z} \,|\, \mathbf{X}, \mathbf{\Theta}) = \prod_{n=1}^{N} P(b^n, z^n \,|\, \mathbf{x}^n, \mathbf{\Theta}) \tag{6}$$

Using bayes rule, we can further expand this probability term

$$P(b^n, z^n \,|\, \mathbf{x}^n, \mathbf{\Theta}) = P(b^n \,|\, z^n, \mathbf{\Theta}) P(z^n \,|\, \mathbf{x}^n, \mathbf{\Theta})$$
$$\implies P(\mathbf{b}, \mathbf{z} \,|\, \mathbf{X}, \mathbf{\Theta}) = \prod_{n=1}^{N} P(b^n \,|\, z^n, \mathbf{\Theta}) P(z^n \,|\, \mathbf{x}^n, \mathbf{\Theta})$$

We can also write the Complete Log Likelihood (CLL) from the above exporession

$$CLL = \sum_{n=1}^{N} \log\big(P(b^n \,|\, z^n, \sigma)\big) + \sum_{n=1}^{N} \log\big(P(z^n \,|\, \mathbf{x}^n, \{\mathbf{w}_l\}_{l=1}^{L})\big) \tag{7}$$

Part 3

Since directly finding $\mathbf{\Theta}_{MLE}$ is a NP hard problem, therefore, we use alternating optimization to find out an approximate MLE estimate of $\mathbf{\Theta}$. The algorithm for alternating optimization is as follows

Time Complexity Analysis

We need to compute $\langle \mathbf{w}_l, \mathbf{x}^n \rangle$ for all items and users. It is optimal to save this and store it before starting the updation steps. This can be done in $\mathcal{O}(NLD)$ steps. We can also compute the values

of the isomorphism h for all subsets *i.e.* compute $h(k)$ for all k. This needs to be done only once, and can be done in the preprocessing part. Hence we need not add the time for this in the iteration time.

Therefore, for all z^n updations, we need to check for all subsets, and we need to compute probability of the user choosing that subset. Since we have already computed some of the terms, we will only require $\mathcal{O}(NKL)$ time for this updation.

The MLE update of σ requires only $\mathcal{O}(N)$.

For the MLE estimate of $\mathbf{w}_l$, we need to find the MLE estimate of a logistic expression, using function approximation. As mentioned, we can assume that each FA takes $\mathcal{O}(ND)$ time, therefore,

we can do this step in $\mathcal{O}(NLD)$.

Algorithm 1: *Alternating Optimization — Hard Assignment*

Initialize $\mathbf{\Theta}_{MLE}$ to $\mathbf{\Theta}^0_{MLE}$
while **no convergence**:

Update z

$$\begin{aligned}
\forall \quad n \in [N]: \quad z^n &= \arg\max_k P(z^n = k \,|\, b^n, \mathbf{x}^n, \mathbf{\Theta}) \\
&= \arg\max_k \log\big(P(b^n, z^n = k \,|\, \mathbf{x}^n, \mathbf{\Theta})\big) \\
&= \arg\min_k \big(b^n - h(k)\big)^2 - \sum_{l \in S_k} \log\big(\sigma\big(\langle \mathbf{w}_l, \mathbf{x}^n \rangle\big)\big) - \sum_{l \notin S_k} \log\big(1 - \sigma\big(\langle \mathbf{w}_l, \mathbf{x}^n \rangle\big)\big)
\end{aligned}$$

Update MLE of $\mathbf{\Theta}$

$$\begin{aligned}
\sigma_{MLE} &= \arg\max_\sigma CLL \\
&= \arg\max_\sigma \log\big(P(\mathbf{b} \,|\, \mathbf{z}, \sigma)\big) \\
&= \sqrt{\frac{1}{N}\sum_{n=1}^{N} (b^n - h(z^n))^2}
\end{aligned}$$

$$\begin{aligned}
\forall \quad l \in [L]: \quad \mathbf{w}_{lMLE} &= \arg\max_{\mathbf{w}_l} CLL \\
&= \arg\max_{\mathbf{w}_l} \log\big(P(\mathbf{z} \,|\, \mathbf{X}, \{\mathbf{w}_l\}_{l=1}^{L})\big) \\
&= \arg\max_{\mathbf{w}_l} \sum_{n=1}^{N} \Big(\mathbb{I}\big[l \in S_{z^n}\big] \log\big(\sigma\big(\langle \mathbf{w}_l, \mathbf{x}^n \rangle\big)\big) + \mathbb{I}\big[l \notin S_{z^n}\big] \log\big(1 - \sigma\big(\langle \mathbf{w}_l, \mathbf{x}^n \rangle\big)\big) \Big)
\end{aligned}$$

Hence the total time for each updation is $\mathcal{O}(NLD + N + NKL)$. Since $L \leq D$, we can clearly say that this is bounded by $\mathcal{O}(NKD)$ or $\mathcal{O}(2^L ND)$.

Part 4

In case of soft assignment, we cannot directly use the mode of the posterior (point estimate) as an estimate of $\mathbf{z}$. Therefore, we take the expected value of $\mathbf{z}$ and use EM algorithm to compute the MLE estimate of $\mathbf{\Theta}$.

It will be easier to use the one-hot representation of the z^n, as we will simply assign the expected probability in the k^{th} index of the user buying the k^{th} subset of items. This is similar to the soft k-means algorithm.

Note: $\mathbf{z}^n$ *represents the one-hot representation, whereas* z^n *represents the numerical value*

Algorithm 2: *Alternating Optimization — Soft Assignment*

Initialize $\boldsymbol{\Theta}_{MLE}$ to $\boldsymbol{\Theta}^0_{MLE}$
while **no convergence**:

Update z

$\forall \quad n \in [N]:$

$$\forall \quad k \in [K]: \qquad \mathbb{E}\left[\mathbf{z}_k^n\right] = \frac{P(S^k|\mathbf{x}^n, \left\{\mathbf{w}_l\right\}_{l=1}^L)\mathcal{N}\left(b^n|h(k), \sigma^2\right)}{\sum_{k'=1}^K P(S^{k'}|\mathbf{x}^n, \left\{\mathbf{w}_l\right\}_{l=1}^L)\mathcal{N}\left(b^n|h(k'), \sigma^2\right)}$$

Update MLE of $\boldsymbol{\Theta}$

$$\begin{aligned}
\sigma_{MLE} &= \arg\max_{\sigma} \mathbb{E}\left[CLL\right] \\
&= \arg\max_{\sigma} \mathbb{E}\left[\log\left(P(\mathbf{b} \mid \mathbf{Z}, \sigma)\right)\right] \\
&= \sqrt{\frac{1}{N}\sum_{n=1}^{N} \mathbb{E}\left[\left(b^n - h(z^n)\right)^2\right]} \\
&= \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left((b^n)^2 - 2\sum_{k=1}^{K}\mathbf{z}_k^n\, h(k) + \sum_{k=1}^{K}\mathbf{z}_k^n\, h(k)^2\right)}
\end{aligned}$$

$$\begin{aligned}
\forall \quad l \in [L]: \qquad \mathbf{w}_{lMLE} &= \arg\max_{\mathbf{w}_l} \mathbb{E}\left[CLL\right] \\
&= \arg\max_{\mathbf{w}_l} \mathbb{E}\left[\log\left(P(\mathbf{Z} \mid \mathbf{X}, \left\{\mathbf{w}_l\right\}_{l=1}^L)\right)\right] \\
&= \arg\max_{\mathbf{w}_l} \mathbb{E}\left[\log\left(\prod_{n=1}^{N}\prod_{k=1}^{K}\left[\prod_{l'\in S_k}\sigma\left(\langle \mathbf{w}_{l'}, \mathbf{x}^n\rangle\right)\prod_{l'\notin S_k}\left(1-\sigma\left(\langle \mathbf{w}_{l'}, \mathbf{x}^n\rangle\right)\right)\right]^{\mathbf{z}_k^n}\right)\right] \\
&= \arg\max_{\mathbf{w}_l} \sum_{k:l\in S_k}\sum_{n=1}^{N}\mathbf{z}_k^n\sigma\left(\langle \mathbf{w}_{l'}, \mathbf{x}^n\rangle\right) + \sum_{k:l\notin S_k}\sum_{n=1}^{N}\mathbf{z}_k^n\left(1-\sigma\left(\langle \mathbf{w}_{l'}, \mathbf{x}^n\rangle\right)\right)
\end{aligned}$$

QUESTION 3

Assignment Number: 2
Student Name: Gurpreet Singh
Roll Number: 150259
Date: October 10, 2017

Part 1

We have the optimization problem $(P1)$ as follows

$$\min_{\mathbf{w},\{\xi_i\}} \|\mathbf{w}\|_2^2 + \sum_{i=1}^{n} \xi_i^2 \qquad \text{(P1)}$$
$$s.t. \forall i \in [n] \quad 1 - y^i \langle \mathbf{w}, \mathbf{x}^i \rangle \leq \xi_i$$
$$\xi_i \geq 0$$

Now consider an optimization problem $(P2)$, which is similar to the optimization problem given in equation (P1), except that we do not have the condition $\xi_i \geq 0$ for any $i \in [n]$. Clearly, the convexity still holds, and hence we will have a solution for $(P2)$.

Consider this solution to be $\mathbf{w}_0, \{\xi_i^0\}$. If we do not have any $k \in [n]$ such that $\xi_k^0 < 0$. Then we can safely claim that the solution for both the optimization problems $(P1)$ and $(P2)$ is the same. Otherwise, we will have at least one $k \in [n]$ such that the condition holds, *i.e.* $\xi_k^0 < 0$. Since this is a solution of the $(P2)$, we can say that this satisfies the condition $1 - y^k \langle \mathbf{w}, \mathbf{x}^k \rangle \leq \xi_k^0$.
Now consider the pair $\mathbf{w}, \{\xi_i^1\}$ where

$$\xi_i^1 = \begin{cases} \xi_i^0 & if i \neq k \\ 0 & if i = k \end{cases}$$

Since $0 > \xi_k^0$, we can say that it follows all constraints of $(P2)$ and $0 < (\xi_k^0)^2$. For all other ξ_i^1, they are the same and hence do not change anything (since all ξ_i are independent). Hence, we can say that $w, \{\xi_i^1\}$ is a solution of $(P2)$, and $val(P2, (w, \{\xi_i^1\})) < val(P2, (w, \{\xi_i^1\}))$.

However, since $w, \{\xi_i^0\}$ was claimed to be a valid solution, this is not possible. Hence the claim that can exist a $k \in [n]$ such that $\xi_k^0 < 0$ is false. Hence, the solution of $(P2)$ is always the same as that of $(P1)$, which suggests that the second constraint is vacuous.

Part 2

$$\min_{\mathbf{w},\{\xi_i\}} \|\mathbf{w}\|_2^2 + \sum_{i=1}^{n} \xi_i^2 \qquad \text{(P1)}$$
$$s.t. \forall i \in [n] \quad 1 - y^i \langle \mathbf{w}, \mathbf{x}^i \rangle \leq \xi_i \quad \text{and} \quad -\xi_i \leq 0$$

We can convert this problem to an unconstrained optimization problem using Langrange Multipliers.

$$\min_{\mathbf{w},\{\xi_i\}} \max_{\boldsymbol{\lambda},\boldsymbol{\gamma}\geq\mathbf{0}} \|\mathbf{w}\|_2^2 + \sum_{i=1}^{n} \xi_i^2 + \sum_{i=1}^{n} \lambda_i\big(1 - y^i\langle \mathbf{w}, \mathbf{x}^i\rangle - \xi_i\big) - \sum_{i=1}^{n} \gamma_i\xi_i \tag{P1}$$

Part 3

We can convert the optimization problem $(P1)$ given in equation (P1) to a dual problem. Assuming strong duality, we can switch the optimization variables.

$$\max_{\boldsymbol{\lambda},\boldsymbol{\gamma}\geq\mathbf{0}} \min_{\mathbf{w},\{\xi_i\}} \|\mathbf{w}\|_2^2 + \sum_{i=1}^{n} \xi_i^2 + \sum_{i=1}^{n} \lambda_i\big(1 - y^i\langle \mathbf{w}, \mathbf{x}^i\rangle - \xi_i\big) - \sum_{i=1}^{n} \gamma_i\xi_i \tag{P1}$$

Now using this, we can first optimize based on the inner optimization (using differentiation). That is, we can partially differentiate w.r.t. to $\mathbf{w}$ and $\{\xi_i\}$ separately, in order to obtain a dual problem.

$$\begin{aligned} \frac{\partial P1}{\partial \mathbf{w}} &= 2\mathbf{w} - \sum_{i=1}^{n} \lambda_i y^i \mathbf{x}^i = 0 \\ \implies \quad \mathbf{w}' &= \frac{1}{2}\sum_{i=1}^{n} \lambda_i y^i \mathbf{x}^i \end{aligned}$$

$$\begin{aligned} \frac{\partial P1}{\partial \xi_k} &= 2\xi_k - \lambda_k - \gamma_i = 0 \\ \implies \quad \xi_k' &= \frac{\lambda_k + \gamma_i}{2} \end{aligned}$$

Hence, we can replace these values in the optimization problem, while solving the inner optimization problem.

$$\begin{aligned} &\max_{\boldsymbol{\lambda},\boldsymbol{\gamma}\geq 0} \quad \big\|\mathbf{w}'\big\|_2^2 + \sum_{i=1}^{n} \xi_i'^2 + \sum_{i=1}^{n} \lambda_i\big(1 - y^i\langle \mathbf{w}', \mathbf{x}^i\rangle - \xi_i'\big) - \sum_{i=1}^{n} \gamma_i\xi_i' \\ &= \max_{\boldsymbol{\lambda},\boldsymbol{\gamma}\geq 0} \quad \left\|\frac{1}{2}\sum_{i=1}^{n} \lambda_i y^i \mathbf{x}^i\right\|_2^2 + \frac{1}{4}\sum_{i=1}^{n} \big(\lambda_i + \gamma_i\big)^2 - \sum_{i=1}^{n} \lambda_i - \sum_{i=1}^{n} \lambda_i y^i \langle \frac{1}{2}\sum_{j=1}^{n} \lambda_j y^j \mathbf{x}^j, \mathbf{x}^i\rangle - \frac{1}{2}\sum_{i=1}^{n} \big(\lambda_i + \gamma_i\big)^2 \\ &= \max_{\boldsymbol{\lambda},\boldsymbol{\gamma}\geq 0} \quad \left\|\frac{1}{2}\sum_{i=1}^{n} \lambda_i y^i \mathbf{x}^i\right\|_2^2 - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i\lambda_j y^i y^j \langle \mathbf{x}^j, \mathbf{x}^i\rangle - \frac{1}{4}\sum_{i=1}^{n} \big(\lambda_i + \gamma_i\big)^2 - \sum_{i=1}^{n} \lambda_i \end{aligned}$$

We can expand the first term as

$$\begin{aligned} \left\|\frac{1}{2}\sum_{i=1}^{n} \lambda_i y^i \mathbf{x}^i\right\|_2^2 &= \frac{1}{4}\left\|\sum_{i=1}^{n} \lambda_i y^i \mathbf{x}^i\right\|_2^2 \\ \implies \left\|\frac{1}{2}\sum_{i=1}^{n} \lambda_i y^i \mathbf{x}^i\right\|_2^2 &= \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \boldsymbol{\lambda}_j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j\rangle \end{aligned}$$

Replacing back in the original equation

$$= \max_{\boldsymbol{\lambda},\boldsymbol{\gamma}\geq\mathbf{0}} \quad -\frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y^i y^j \langle \mathbf{x}^j, \mathbf{x}^i\rangle - \frac{1}{4}\sum_{i=1}^{n}\left(\lambda_i+\gamma_i\right)^2 - \sum_{i=1}^{n}\lambda_i$$

We can alter this optimization problem by multiplying with -4 and inverting the optimizing condition. Hence the dual can be given as follows

$$\min_{\boldsymbol{\lambda},\boldsymbol{\gamma}\geq\mathbf{0}} \boldsymbol{\lambda}^T Q\boldsymbol{\lambda} + \sum_{i=1}^{n}\left(\lambda_i+\gamma_i\right)^2 - 4\lambda_i \tag{D1}$$

where

$$Q_{ij} = y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j\rangle$$

Note that $\lambda_i, \gamma_i \geq 0$ for all $i \in [n]$. Hence, $\left(\lambda_i+\gamma_i\right)^2 \geq \left(\lambda_i\right)^2$, hence it is very trivial to see that $\gamma_i = 0$ for all $i \in [n]$. Therefore, we can further simplify the dual as follows

$$\min_{\boldsymbol{\lambda}\geq\mathbf{0}} \boldsymbol{\lambda}^T Q\boldsymbol{\lambda} + \sum_{i=1}^{n}\left(\lambda_i^2 - 4\lambda_i\right) \tag{D1}$$

where

$$Q_{ij} = y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j\rangle$$

Part 4

Let us state the original dual problem.

$$\min_{\boldsymbol{\alpha}\in[0,C]^n} \boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \sum_{i=1}^{n}\boldsymbol{\alpha}_i$$

The main difference that seems to be there between the two objectives is that the dual variable (each value) in the original problem is upper bounded by a constant C.

It seems that the dual variable in $(D1)$ has no upper bound, however there is still a tradeoff between the high and low values of our dual variable. Consider $\boldsymbol{\lambda} > 2$. In this case, the second term will become positive. Hence, we do not want very hight values of $\boldsymbol{\lambda}$. Looking at the primal problem of the original SVM, we can say that in our case (by similarity of terms), C = 2. Hence we are trying to keep the value of λ within the range $[0, C]^n$, however it is a loose bound.

Therefore, the only difference is that in the original SVM problem, we have a strict upper bound on the value of the dual variable, whereas in our case, the upper bound is loose, but still existant.

Part 5

No. From our approach, we have proved that $\xi_i < 0$ only increases the value of the objective function while $\xi_i = 0$ is a solution, and hence the lower bound is vacuous, however, this is not the case with the original SVM. In the original problem, we are hoping to get negative values of ξ_i which would mean that we are giving weightage to the points which are far from the margin, which is not a correct optimization problem, as it essentials ignores the large margin problem. Hence, in that case, we need to explicitly add the lower bound.

QUESTION 4

Assignment Number: 2
Student Name: Gurpreet Singh
Roll Number: 150259
Date: October 10, 2017

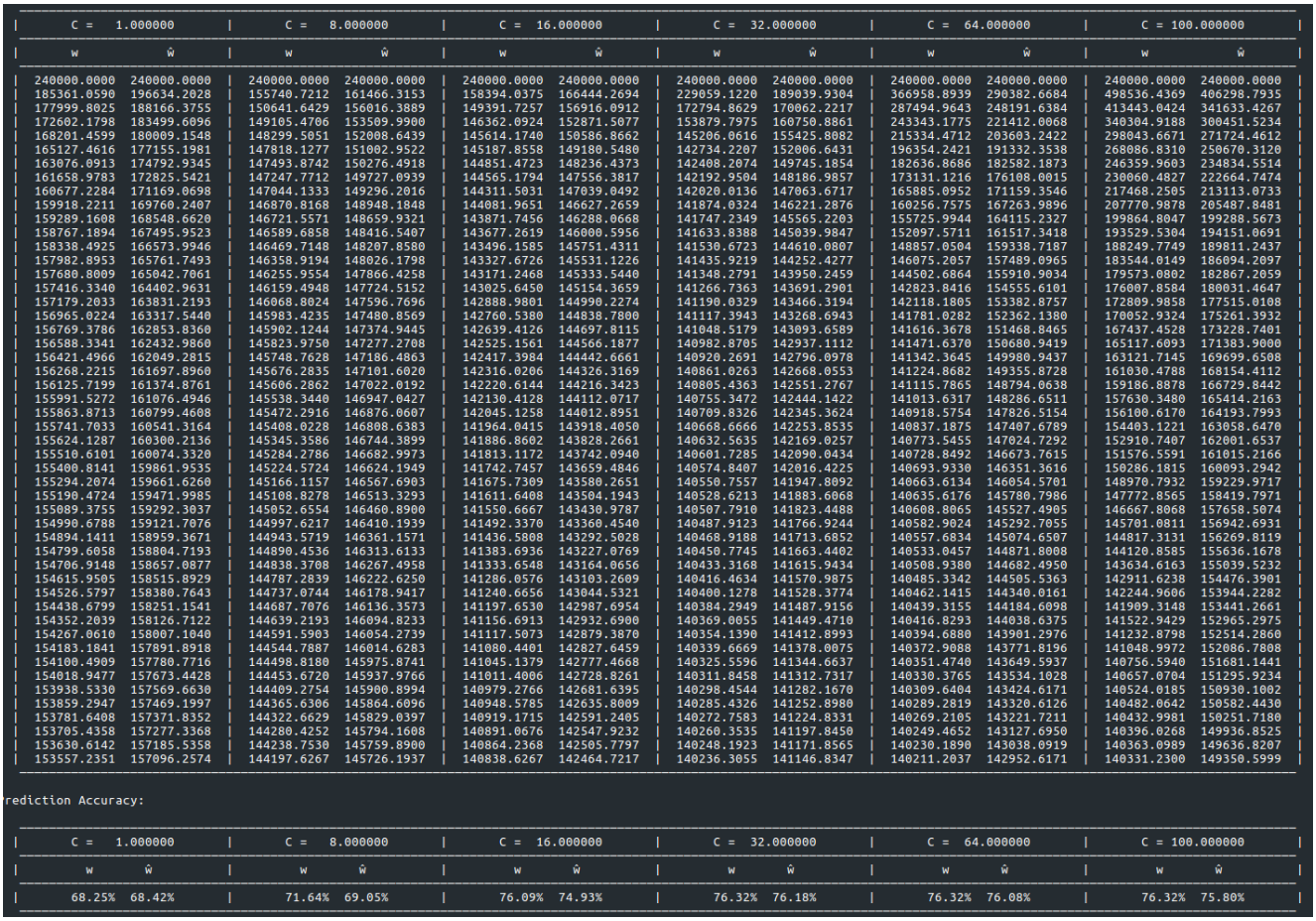
Part 3

The averaging trick did not give better results in the case of Vanilla Gradient Descent. Determined on the basis of held-out validation, the model with w_bar gave slightly lesser accuracy as well as higher value of the objective function for all iterations. This is evident in the validation data provided in figure 2

Part 4

I took $h(n)$ as $\frac{1}{n}$, as it allows for averaging the sum of the support vectors. As for the constant multiplied to $\frac{h(n)}{\sqrt{t+1}}$, I used held-out validation to find the fastest converging as well as the one with the best accuracy. Amongst the set of values $\{1, 8, 16, 32, 64, 100\}$, the best value came out to be 32.

Hence, the best value of η for me was $\frac{32}{n\sqrt{t+1}}$. This is inspite of the fact that initially the value of the objective function increases *i.e.* the objective function first diverges, then converges. Using a smaller value of the constant (5) avoids this problem, however gives lower final accuracy and slower convergence.



C = 1.000000		C = 8.000000		C = 16.000000		C = 32.000000		C = 64.000000		C = 100.000000	
w	ŵ	w	ŵ	w	ŵ	w	ŵ	w	ŵ	w	ŵ
240000.0000	240000.0000	240000.0000	240000.0000	240000.0000	240000.0000	240000.0000	240000.0000	240000.0000	240000.0000	240000.0000	240000.0000
185361.0590	196634.2028	155740.7212	161466.3153	158394.0375	166444.2694	229059.1220	189039.9304	366958.8939	290382.6684	498536.4369	406298.7935
177999.8025	188166.3755	150641.6429	156016.3889	149391.7257	156916.0912	172794.8629	170062.2217	287494.9643	248191.6384	413443.0424	341633.4267
172602.1798	183499.6096	149105.4706	153509.9900	146362.0924	152871.5077	153879.7975	160750.8861	243343.1775	221412.0068	340304.9188	300451.5234
168201.4599	180009.1548	148299.5051	152008.6439	145614.1740	150586.8662	145206.0616	155425.8082	215334.4712	203603.2422	298043.6671	271724.4612
165127.4616	177155.1981	147818.1277	151002.9522	145187.8558	149180.5480	142734.2207	152006.6431	196354.2421	191332.3538	268086.8310	250670.3120
163076.0913	174792.9345	147493.8742	150276.4918	144851.4723	148236.4373	142408.2074	149745.1854	182636.8686	182582.1873	246359.9603	234834.5514
161658.9783	172825.5421	147247.7712	149727.0939	144565.1794	147556.3817	142192.9504	148186.9857	173131.1216	176108.0015	230060.4827	222664.7474
160677.2284	171169.0698	147044.1333	149296.2016	144311.5031	147039.0492	142020.0136	147063.6717	165885.0952	171159.3546	217468.2505	213113.0733
159918.2211	169760.2407	146870.8168	148948.1848	144081.9651	146627.2659	141874.0324	146221.2876	160256.7575	167263.9896	207770.9878	205487.8481
159289.1608	168548.6620	146721.5571	148659.9321	143871.7456	146288.0668	141747.2349	145565.2203	155725.9944	164115.2327	199864.8047	199288.5673
158767.1894	167495.9523	146589.6858	148416.5407	143677.2619	146000.5956	141633.8388	145039.9847	152097.5711	161517.3418	193529.5304	194151.0691
158338.4925	166573.9946	146469.7148	148207.8580	143496.1585	145751.4311	141530.6723	144610.0807	148857.0504	159338.7187	188249.7749	189811.2437
157982.8953	165761.7493	146358.9194	148026.1798	143327.6726	145531.1226	141435.9219	144252.4277	146075.2057	157489.0965	183544.0149	186094.2097
157680.8009	165042.7061	146255.9554	147866.4258	143171.2468	145333.5440	141348.2791	143950.2459	144502.6864	155910.9034	179573.0802	182867.2059
157416.3340	164402.9631	146159.4948	147724.5152	143025.6450	145154.3659	141266.7363	143691.2901	142823.8416	154555.6101	176007.8584	180031.4647
157179.2033	163831.2193	146068.8024	147596.7696	142888.9801	144990.2274	141190.0329	143466.3194	142118.1805	153382.8757	172809.9858	177515.0108
156965.0224	163317.5440	145983.4235	147480.8569	142760.5380	144838.7800	141117.3943	143268.6943	141781.0282	152362.1380	170052.9324	175261.3932
156769.3786	162853.8360	145902.1244	147374.9445	142639.4126	144697.8115	141048.5179	143093.6589	141616.3678	151468.8465	167437.4528	173228.7401
156588.3341	162432.9860	145823.9750	147277.2708	142525.1561	144566.1877	140982.8705	142937.1112	141471.6370	150680.9419	165117.6093	171383.9000
156421.4966	162049.2815	145748.7628	147186.4863	142417.3984	144442.6661	140920.2691	142796.0978	141342.3645	149980.9437	163121.7145	169699.6508
156268.2215	161697.8960	145676.2835	147101.6020	142316.0206	144326.3169	140861.0263	142668.0553	141224.8682	149355.8728	161030.4788	168154.4112
156125.7199	161374.8761	145606.2862	147022.0192	142220.6144	144216.3423	140805.4363	142551.2767	141115.7865	148794.0638	159186.8878	166729.8442
155991.5272	161076.4946	145538.3440	146947.0427	142130.4128	144112.0717	140755.3472	142444.1422	141013.6317	148286.6511	157630.3480	165414.2163
155863.8713	160799.4608	145472.2916	146876.0607	142045.1258	144012.8951	140709.8326	142345.3624	140918.5754	147826.5154	156100.6170	164193.7993
155741.7033	160541.3164	145408.0228	146808.6383	141964.0415	143918.4050	140668.6666	142253.8535	140837.1875	147407.6789	154403.1221	163058.6470
155624.1287	160300.2136	145345.3586	146744.3899	141886.8602	143828.2661	140632.5635	142169.0257	140773.5455	147024.7292	152910.7407	162001.6537
155510.6101	160074.3320	145284.2786	146682.9973	141813.1172	143742.0940	140601.7285	142090.0434	140728.8492	146673.7615	151576.5591	161015.2166
155400.8141	159861.9535	145224.5724	146624.1949	141742.7457	143659.4846	140574.8407	142016.4228	140693.9330	146351.3616	150286.1815	160093.2942
155294.2074	159661.6260	145166.1157	146567.6903	141675.7309	143580.2651	140550.7557	141947.8092	140663.6134	146054.5701	148970.7932	159229.9717
155190.4724	159471.9985	145108.8278	146513.3293	141611.6408	143504.1943	140528.6213	141883.6068	140635.6176	145780.7986	147772.8565	158419.7971
155089.3755	159292.3037	145052.6554	146460.8900	141550.6667	143430.9787	140507.7910	141823.4488	140608.8065	145527.4905	146667.8068	157658.5074
154990.6788	159121.7076	144997.6217	146410.1939	141492.3370	143360.4540	140487.9123	141766.9244	140582.9024	145292.7055	145701.0811	156942.6931
154894.1411	158959.3671	144943.5719	146361.1571	141436.5808	143292.5028	140468.9188	141713.6852	140557.6834	145074.6507	144817.3131	156269.8119
154799.6058	158804.7193	144890.4536	146313.6133	141383.6936	143227.0769	140450.7745	141663.4402	140533.0457	144871.8008	144120.8585	155636.1678
154706.9148	158657.0877	144838.3708	146267.4958	141333.6548	143164.0656	140433.3168	141615.9434	140508.9380	144682.4950	143634.6163	155039.5232
154615.9505	158515.8929	144787.2839	146222.6250	141286.0576	143103.2609	140416.4634	141570.9875	140485.3342	144505.5363	142911.6238	154476.3901
154526.5797	158380.7643	144737.0744	146178.9417	141240.6656	143044.5321	140400.1278	141528.3774	140462.1415	144340.0161	142244.9606	153944.2282
154438.6799	158251.1541	144687.7076	146136.3573	141197.6530	142987.6954	140384.2949	141487.9156	140439.3155	144184.6098	141909.3148	153441.2661
154352.2039	158126.7122	144639.2193	146094.8233	141156.6913	142932.6900	140369.0055	141449.4710	140416.8293	144038.6375	141522.9429	152965.2975
154267.0610	158007.1040	144591.5903	146054.2739	141117.5073	142879.3870	140354.1390	141412.8993	140394.6880	143901.2976	141232.8798	152514.2860
154183.1841	157891.8918	144544.7887	146014.6283	141080.4401	142827.6459	140339.6669	141378.0075	140372.9088	143771.8196	141048.9972	152086.7808
154100.4909	157780.7716	144498.8180	145975.8741	141045.1379	142777.4668	140325.5596	141344.6637	140351.4740	143649.5937	140756.5940	151681.1441
154018.9477	157673.4428	144453.6720	145937.9766	141011.4006	142728.8261	140311.8458	141312.7317	140330.3765	143534.1028	140657.0704	151295.9234
153938.5330	157569.6630	144409.2754	145900.8994	140979.2766	142681.6395	140298.4544	141282.1670	140309.6404	143424.6171	140524.0185	150930.1002
153859.2947	157469.1997	144365.6306	145864.6096	140948.5785	142635.8009	140285.4326	141252.8980	140289.2819	143320.6126	140482.0642	150582.4430
153781.6408	157371.8352	144322.6629	145829.0397	140919.1715	142591.2405	140272.7583	141224.8331	140269.2105	143221.7211	140432.9981	150251.7180
153705.4358	157277.3368	144280.4252	145794.1608	140891.0676	142547.9232	140260.3535	141197.8450	140249.4652	143127.6950	140396.0268	149936.8525
153630.6142	157185.5358	144238.7530	145759.8900	140864.2368	142505.7797	140248.1923	141171.8565	140230.1890	143038.0919	140363.0989	149636.8207
153557.2351	157096.2574	144197.6267	145726.1937	140838.6267	142464.7217	140236.3055	141146.8347	140211.2037	142952.6171	140331.2300	149350.5999

rediction Accuracy:

C = 1.000000		C = 8.000000		C = 16.000000		C = 32.000000		C = 64.000000		C = 100.000000	
w	ŵ	w	ŵ	w	ŵ	w	ŵ	w	ŵ	w	ŵ
68.25%	68.42%	71.64%	69.05%	76.09%	74.93%	76.32%	76.18%	76.32%	76.08%	76.32%	75.80%

Figure 2: Cross validation data for some values of η

Part 5 and Part 6

The plots have been generated after running GD and SCD on the complete training dataset *i.e.* 300K examples.

For GD, I have taken $n\_iter = 2000$ and $spacing = 10$, whereas for SCD, $n\_iter = 100000$ and $spacing = 5000$.

From the figures, it is evident that Gradient Descent gives lower objective function value, and thus a better solution, wheras SCD gives a lot of fluctuations. However, this is not actually the case. In the case of SCD, there is obviously a tradeoff between the objective value and margin, and the speed. The SCD method very quickly gives good accuracy, wheras the GD takes much more time to give the same accuracy.
Also, looking at the theoretical plots, it seems that the work done by SCD is much smaller than that of GDHowever, this is not in relevance to the observed time for the same set of training data. The theoretical time and wall-clock time are definitely correlated, however, it seems that there is a lot more overhead on $\mathcal{O}(1)$ operations than expected by the work done per iteration in the SCD method. The difference in the methods might become more relevant in case of much larger values of the size of the dataset used.

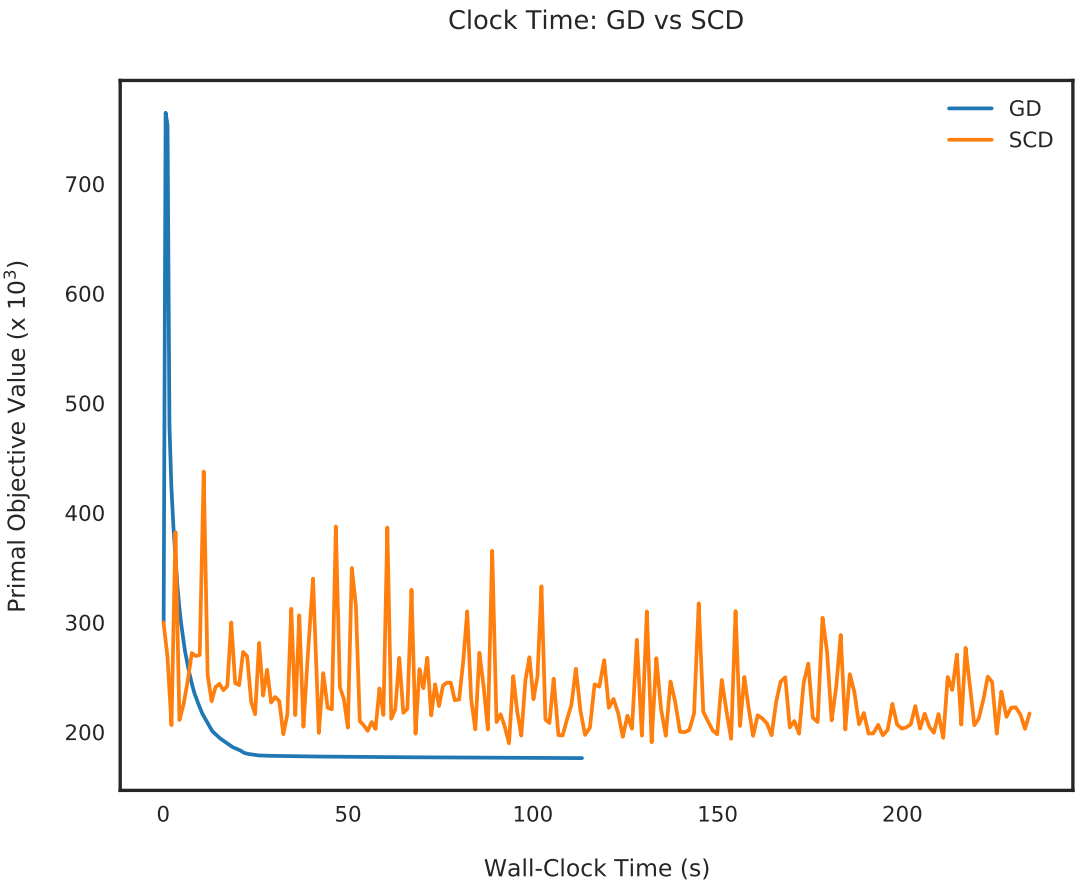


Figure 3: Clock Time Analysis of Gradient Descent vs Stochastic Coordinate Descent

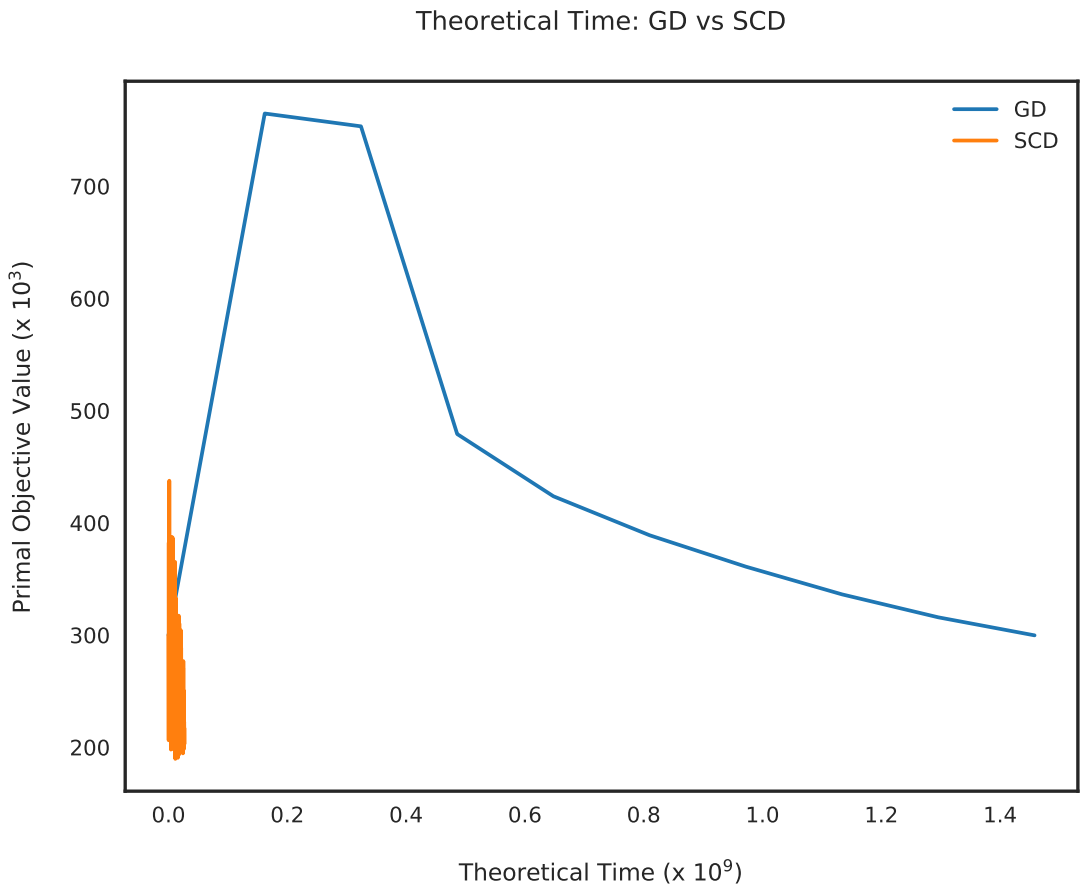


Figure 4: Theoretical Time Analysis of Gradient Descent vs Stochastic Coordinate Descent

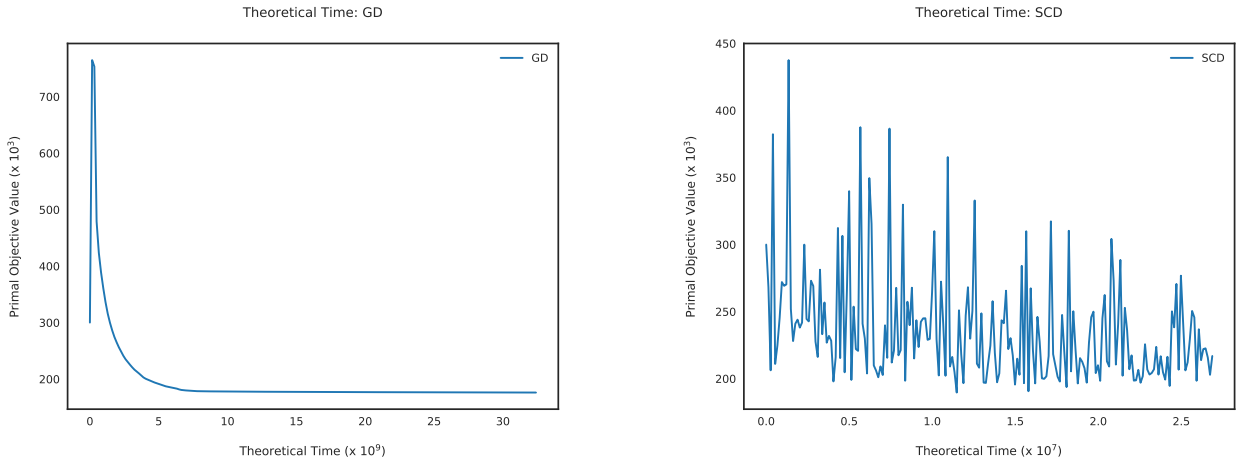


Figure 5: Theoretical Time Analysis of (a) Gradient Descent and (b) Stochastic Coordinate Descent