CSCI-567: Assignment #4

Due on Monday, November 9, 2015

Saket Choudhary
2170058637

Contents

Problem 1

Problem 1: (a) Gradient Calculation

$$L(y_i, \hat{y_i}) = \log(1 + \exp(-y_i\hat{y_i}))$$

$$g_i = \frac{\partial L(y_i, \hat{y_i})}{\partial \hat{y_i}}$$

$$\boxed{g_i = \frac{-y_i \exp(-y_i\hat{y_i})}{1 + \exp(-y_i\hat{y_i})}}$$

Problem 1: (b) Weak Learner Section

$$h^* = \min_{h \in H} \big(\min_{\gamma \in R} \sum_{i=1}^{n} (-g_i - \gamma h(x_i))^2 \big)$$

$$\implies \frac{\partial h^*}{\partial \gamma} = 0$$

$$\implies 2\sum_{i=1}^{n}(-g_i - \gamma h(x_i))(-h(x_i)) = 0$$

$$\boxed{\hat{h} = -\frac{\sum_{i=1}^{n} g_i h(x_i)}{\sum_{i=1}^{n} h(x_i)^2}}$$

Also check if it is indeed minimum with a second derivative test:

$$\frac{\partial^2 h^*}{\partial \gamma^2} = 2\sum_{i=1}^{n} h(x_i)^2 > 0$$

Since the second derivative is positive definite, $ga\hat{m}ma$ is indeed where the minima occurs.

Problem 1 continued on next page...

Problem 1: (c) Step Size Selection

$$\alpha^* = \arg\min_{\alpha \in R} \sum_{1}^{n} L(y_i, \hat{y}_i + \alpha h^*(x_i))$$

Newton's approximation:

$$\alpha_1 = \alpha_0 - \frac{f'(\alpha_0)}{f''(\alpha_0)}$$

We start from $\alpha_0 = 0$ and hence:

$$\begin{aligned}
f(\alpha_0) &= \sum_{i=1}^{n} \log(1 + \exp(-y_i \hat{y}_i)) \\
f'(\alpha) &= \sum_{i=1}^{n} \frac{\partial L}{\partial \alpha} \\
&= \sum_{i=1}^{n} \frac{-y_i h^*(x_i) \exp\left(-y_i(\hat{y}_i + \alpha h^*(x_i))\right)}{1 + \exp\left(-y_i(\hat{y}_i + \alpha h^*(x_i))\right)}
\end{aligned}$$

$$\boxed{f'(\alpha = \alpha_0) = -\sum_{i=1}^{n} \frac{y_i h^*(x_i) \exp(-y_i \hat{y}_i)}{1 + \exp(-y_i \hat{y}_i)}}$$

And,

$$\begin{aligned}
f''(\alpha) &= \sum_{i=1}^{n} \frac{\partial^2 L}{\partial \alpha^2} \\
&= \sum_{i=1}^{n} \frac{\left\{\left(1 + \exp\left(-y_i(\hat{y}_i + \alpha h^*(x_i))\right)\right)(y_i h^*(x_i))^2 + y_i h^*(x_i)\right\} \exp\left(-y_i(\hat{y}_i + \alpha h^*(x_i))\right)}{\left(1 + \exp\left(-y_i(\hat{y}_i + \alpha h^*(x_i))\right)\right)^2}
\end{aligned}$$

$$\boxed{f''(\alpha_0) = \sum_{i=1}^{n} \frac{\left\{\left(1 + \exp\left(-y_i \hat{y}_i\right)\right)(y_i h^*(x_i))^2 + y_i h^*(x_i)\right\} \exp\left(-y_i \hat{y}_i\right)}{\left(1 + \exp\left(-y_i \hat{y}_i\right)\right)^2}}$$

Thus,

$$\boxed{\alpha_1 = \frac{\sum_{i=1}^{n} \frac{y_i h^*(x_i) \exp(-y_i \hat{y}_i)}{1 + \exp(-y_i \hat{y}_i)}}{\sum_{i=1}^{n} \frac{\left\{\left(1 + \exp\left(-y_i \hat{y}_i\right)\right)(y_i h^*(x_i))^2 + y_i h^*(x_i)\right\} \exp\left(-y_i \hat{y}_i\right)}{\left(1 + \exp\left(-y_i \hat{y}_i\right)\right)^2}}}$$

Problem 2

Problem 2: (a)

Primal form:

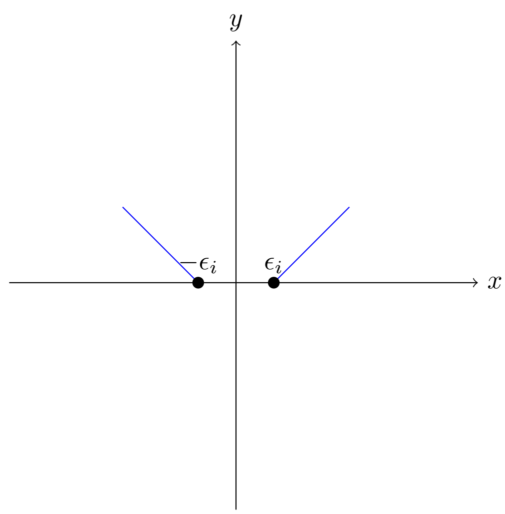
$$\begin{aligned}
&\min_{w} ||w||^2 \\
\text{such that } &|y_i - (w^T x_i + b)| \leq \epsilon
\end{aligned}$$

Problem 2: (b)

$$\min_{w,\epsilon_i} \frac{1}{2}||w||^2 + C\sum_i \epsilon_i$$
$$\text{such that } (w^T x_i + b) - y_i \leq n_i + \epsilon_i \text{ (positive deviation)}$$
$$\text{and } y_i - (w^T x_i + b) \leq p_i + \epsilon_i \text{ (negative deviation)}$$
$$n_i \geq 0$$
$$p_i \geq 0$$

Also, the slackness loss needs further constraints:

$$n_i = \begin{cases} 0 & |n_i| < \epsilon_i, \\ |n_i| - \epsilon & otherwise \end{cases}$$
$$p_i = \begin{cases} 0 & |p_i| < \epsilon_i, \\ |p_i| - \epsilon & otherwise \end{cases}$$



So essentially n_i, p_i are non zero, only above the two blue lines

Problem 2: (c)

$$L = \frac{1}{2}||w||^2 + C\sum_i (p_i + n_i)$$
$$- \sum_i (\lambda_i p_i + \lambda_i' n_i)$$
$$- \sum_i \alpha_i(\epsilon + p_i - (y_i - (w^T x_i + b)))$$
$$- \sum_i \beta_i(\epsilon + n_i + (y_i - (w^T x_i + b)))$$

Conditions :

$$\alpha_i \geq 0$$
$$\beta_i \geq 0$$
$$\lambda_i \geq 0$$
$$\lambda_i' \geq 0$$

Dual Form(all summations are from 1 to n)::

$$\Delta_w L = 0$$
$$= w - \sum_i \alpha_i x_i + \sum_i \beta_i x_i = 0$$
$$= w - \sum_i (\alpha_i - \beta_i) x_i = 0$$
$$\Delta_b L = 0$$
$$= \sum_i \alpha_i - \sum_i \beta_i = 0$$
$$\Delta_{p_i} L = 0$$
$$= C - \sum \lambda_i - \sum_i \alpha_i = 0$$
$$\Delta_{n_i} L = 0$$
$$= C - \sum \lambda_i' - \sum_i \beta_i = 0$$

Thus, w is given by:

$$w = \sum_i \alpha_i x_i - \sum_i \beta_i x_i$$

depends only on the support vectors.
This reduces the optimisation to:

$$\begin{aligned}
\max f &= \frac{1}{2}\sum_{i,j}(\alpha_i - \beta_i)x_i^T x_j(\alpha_j - \beta_j) + p_i(C - \sum_i \lambda_i - \sum_i \alpha_i) \\
&+ n_i(C + \sum_i \lambda_i' - \sum_i \beta_i) \\
&+ \epsilon(-\sum_i \alpha_i - \sum_i \beta_i) \\
&+ \sum_i y_i(\alpha_i - \beta_i) - \sum_i(\alpha_i w^T x_i - \beta_i w^T x_i) \\
&= -\frac{1}{2}\sum_{i,j}(\alpha_i - \beta_i)x_i^T x_j(\alpha_j - \beta_j) - \epsilon(\sum_i(\alpha_i + \beta_i)) \\
&+ \sum_i y_i(\alpha_i - \beta_i)
\end{aligned}$$

$$\text{such that} \sum_i(\alpha_i - \beta_i) = 0$$

$$\text{and } \alpha_i, \beta_i \in [0, C]$$

Problem 2: (d)

Using Kernel transformation, we simply replace $x_i^T x_j$ with $k(x_i, x_j)$:

$$w = \sum_i(\alpha_i - \beta_i)\phi(x_i)$$

this happens because $x_i' x_j$ gets mapped onto by an equivalent kernel function $k(x_i, x_j) = \phi^T(x_i)\phi(x_j)$ and the objective function is:

$$\max_f = \frac{1}{2}\sum_{i,j}(\alpha_i - \beta_i)k(x_i, x_j)(\alpha_j - \beta_j) - \epsilon(\sum_i(\alpha_i + \beta_i)) + \sum_i y_i(\alpha_i - \beta_i)$$

Problem 3.3

Problem 3.3: (a)

C	Tr. Dataset 1(t)	Tr. Dataset 2(t)	Training Dataset 3(t)	CV	Avg. t
4^{-6}=0.000244	0.606156	0.400791	0.346078	0.578976	0.451
4^{-5}=0.000977	0.379495	0.477559	0.498054	0.907001	0.451
4^{-4}=0.003906	0.529940	0.495841	0.534946	0.926001	0.520
4^{-3}=0.015625	0.516236	0.577324	0.561874	0.935501	0.551
4^{-2}=0.062500	0.512287	0.529517	0.554510	0.945006	0.532
4^{-1}=0.250000	0.630195	0.663459	0.657651	0.943010	0.650
4^{0}=1.000000	0.746649	0.601710	0.563063	0.939003	0.637
4^{1}=4.000000	0.633370	0.572011	0.595552	0.942501	0.600
4^{2}=16.000000	0.674677	0.681010	0.698041	0.943503	0.684

The time shown(t) is in seconds for three partitions, the last column being the average time

As seen from the table. the time seems to increase with C and the CV increases too. C determines the tradeoff between objective function complexity and the overall loss. When C is small, there are chances of overfitting, this is evident from low CV values for lower C(because the generalisation error is high, and this is where cross validation is helpful)
The larger the value of C, the more is the penalisation and hence smaller the ϵ_i would be

Problem 3.3: (b)

Based on lowest cross validation error. $C = 4^2$

Problem 3.3: (c)

With $C = 16$, test accuracy = 0.943500

Problem 3.4

Problem 3.4: (a)

Platform Used: *Ubuntu* 12.04, *x*86_64
'libsvm' gives 0.9655 as its accuracy which is pretty close to 0.943 that my code gives.

C	Training Time	CV
4^{-6}	0.829624	0.5575
4^{-5}	0.827846	0.5575
4^{-4}	0.831586	0.5575
4^{-3}	0.831599	0.7295
4^{-2}	0.649459	0.9195
4^{-1}	0.425499	0.934
4^{0}	0.281219	0.949
4^{1}	0.210071	0.955
4^{2}	0.201989	0.9655

Problem 3.4 continued on next page...

Problem 3.4: (b)

‘libsvm‘ gives 0.9455 as its accuracy which is pretty close to

Problem 3.5

Problem 3.5: (a)

RBF

C	γ	Training Time	CV
0.015625	0.000061	0.799739	55.750000
0.015625	0.000244	0.799145	55.750000
0.015625	0.000977	0.798638	55.750000
0.015625	0.003906	0.801114	55.750000
0.015625	0.015625	0.802529	64.750000
0.015625	0.062500	0.805714	73.750000
0.015625	0.250000	0.822912	55.750000
0.062500	0.000061	0.799229	55.750000
0.062500	0.000244	0.798608	55.750000
0.062500	0.000977	0.799373	55.750000
0.062500	0.003906	0.800940	83.350000
0.062500	0.015625	0.648832	91.200000
0.062500	0.062500	0.634587	91.850000
0.062500	0.250000	0.824124	63.600000
0.250000	0.000061	0.798746	55.750000
0.250000	0.000244	0.799292	55.750000
0.250000	0.000977	0.799286	86.350000
0.250000	0.003906	0.587110	92.000000
0.250000	0.015625	0.426160	93.200000
0.250000	0.062500	0.423444	94.750000
0.250000	0.250000	0.713477	92.300000
1.000000	0.000061	0.800186	55.750000
1.000000	0.000244	0.797172	86.900000
1.000000	0.000977	0.571743	91.850000
1.000000	0.003906	0.381565	93.050000
1.000000	0.015625	0.281917	94.650000
1.000000	0.062500	0.285756	96.150000
1.000000	0.250000	0.568394	96.100000
4.000000	0.000061	0.798443	86.950000
4.000000	0.000244	0.569461	91.850000
4.000000	0.000977	0.371507	93.000000
4.000000	0.003906	0.259033	94.250000
4.000000	0.015625	0.203123	95.550000
4.000000	0.062500	0.230975	96.650000
4.000000	0.250000	0.539952	96.100000
16.000000	0.000061	0.570222	91.850000
16.000000	0.000244	0.369962	93.050000
16.000000	0.000977	0.261383	93.950000
16.000000	0.003906	0.203001	95.050000
16.000000	0.015625	0.195201	96.500000
16.000000	0.062500	0.225650	96.900000
16.000000	0.250000	0.541735	95.950000
64.000000	0.000061	0.369279	93.050000
64.000000	0.000244	0.262142	93.950000
64.000000	0.000977	0.210888	94.750000
64.000000	0.003906	0.191527	94.950000
64.000000	0.015625	0.183208	96.650000
64.000000	0.062500	0.229521	96.550000
64.000000	0.250000	0.541598	95.950000
256.000000	0.000061	0.257230	93.900000
256.000000	0.000244	0.214556	94.700000
256.000000	0.000977	0.191176	94.900000
256.000000	0.003906	[illegible]	[illegible]

Problem 3.5: (b)

C	Training Time	CV
0.015625	0.806863	0.5575
0.062500	0.804094	0.5575
0.250000	0.804537	0.5575
1.000000	0.807285	0.5575
4.000000	0.810119	0.6475
16.000000	0.815702	0.7375
64.000000	0.828215	0.55750
256.000000	0.803329	0.5575
1024.000000	0.803978	0.5575
4096.000000	0.806347	0.5575
16384.000000	0.816474	0.8335

Based on the RBF, polykernel:
Polynomal Kernel train accuracy: 96.600000
RBF Kernel train accuracy: 96.900000
Better kernel: RBF
Polynomial Kernel optimal C: 64.000000
Polynomial Kernel optimal degree: 2.000000
Polynomal Kernel test accuracy: 95.150000
RBF Kernel optimal g: 0.062500
RBF Kernel optimal c: 16.000000
RBF Kernel test accuracy: 96.500000