Machine Learning Assignment 1

Yihong Chen April 3, 2018

1 Mathematics Basics

1.1 Optimization

Put the problem into the standard form of constrained optimization

$$\begin{aligned} \min_{x_1,x_2} \quad & x_1^2 + x_2^2 - 1 \\ s.t. \quad & x_1 + x_2 - 1 = 0 \\ & -x_1 + 2x_2 \leq 0 \end{aligned} \tag{1}$$

The *Lagrangian* of the problem is given as follows

$$L(x_1, x_2, \lambda, \mu) = x_1^2 + x_2^2 - 1 + \lambda(x_1 + x_2 - 1) + \mu(-x_1 + 2x_2)$$

Applying the KKT condition, we obtain

$$\begin{aligned} 2x_1 + \lambda - \mu &= 0 \\ 2x_2 + \lambda + 2\mu &= 0 \\ \mu(x_1 - 2x_2) &= 0 \\ -x_1 + 2x_2 &\leq 0 \\ x_1 + x_2 - 1 &= 0 \\ \mu &\geq 0 \end{aligned} \tag{2}$$

which gives the solution

$$\begin{aligned} x_1 &= \frac{2}{3} \\ x_2 &= \frac{1}{3} \\ \mu &= \frac{2}{9} \\ \lambda &= -\frac{10}{9} \end{aligned} \tag{3}$$

The optimal objective is $-\frac{4}{9}$

1.2 Calculus

(1) Applying integration by parts, we obtain

$$\begin{aligned}\Gamma(x+1) &= \int_0^\infty u^x e^{-u} du \\ &= -u^x e^{-u}|_{u=0}^\infty - \int_0^\infty (-x u^{x-1} e^{-u}) du \\ &= 0 + x\int_0^\infty u^{x-1} e^{-u} du \\ &== x\Gamma(x)\end{aligned} \tag{4}$$

(2)

$$\begin{aligned}\Gamma(a)\Gamma(b) &= \int_0^\infty u^{a-1} e^{-u} du \int_0^\infty v^{b-1} e^{-v} dv \\ &= \int_{v=0}^\infty \int_{u=0}^\infty u^{a-1} v^{b-1} e^{-(u+v)} du dv\end{aligned} \tag{5}$$

Changing variables by $u = zt$ and $v = z(1-t)$, we obtain

$$\begin{aligned}\Gamma(a)\Gamma(b) &= \int_{z=0}^\infty \int_{t=0}^1 e^{-z} (zt)^{a-1} (z(1-t))^{b-1} z dt dz \\ &= \int_{z=0}^\infty e^{-z} z^{a+b-1} dz \int_{t=0}^1 t^{a-1} (1-t)^{b-1} dt \\ &= \Gamma(a+b) \int_{\mu=0}^1 \mu^{a-1} (1-\mu)^{b-1} d\mu\end{aligned} \tag{6}$$

1.3 Probability

$$\begin{aligned}p(\lambda|x) &\propto p(\lambda) p(x|\lambda) \\ &\propto \frac{\beta^\alpha \lambda^{\alpha-1} e^{-\beta\lambda}}{\Gamma(\alpha)} \frac{e^{-\lambda} \lambda^x}{x!} \\ &\propto \lambda^{\alpha+x-1} e^{-(\beta+1)\lambda}\end{aligned} \tag{7}$$

which means $\lambda|x \sim \Gamma(\lambda|\alpha+x, \beta+1)$ Gamma distribution can serve as a conjugate prior to the Poisson distribution.

1.4 Stochastic Process

Tossing a fair coin for a number of times, the outcome of tosses follows *Bernoulli* Process. Let us denote as $E[N]$ the expected number of tosses needed to observe the required pattern. The outcome of tosses include the following cases

- If we get a H at the 1st toss:

$$E[N|Z_1 = 1] = 1 + E[N]$$

- If we get a T at the 1st toss and H at the 2nd toss:

$$E[N|Z_1 = 0, Z_2 = 1] = 2 + E[N]$$

- If we get a T at the 1st toss and T at the 2nd toss ... H at the kth:

$$E[N|Z_1... = Z_{k-1} = 0, Z_k = 1] = k + E[N]$$

- If we get T from 1st to kth toss:

$$E[N|Z_1... = Z_k = 0] = k$$

Summing over all the cases, we obtain

$$E[N] = \frac{1}{2} \times (1 + E[N]) + \frac{1}{2^2} \times (2 + E[N]) ... \frac{1}{2^k} \times (k + E[N]) + \frac{1}{2^k} \times k$$

The solution to the equation

$$E[N] = 2^{k+1} - 2$$

2 SVM

This problem requires us to derive the dual form of the support vector regression. The primal *Lagrangian* is

$$\begin{aligned} L_p(\omega, b, \xi_i, \hat{\xi}_i, \alpha_i, \hat{\alpha_i}) =& \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_i(\xi + \hat{\xi}_i) \\ &+ \sum_i \alpha_i(\boldsymbol{\omega}^\intercal \boldsymbol{x_i} + b - y_i - \epsilon - \xi_i) \\ &+ \sum_i \hat{\alpha_i}(y_i - \boldsymbol{\omega}^\intercal \boldsymbol{x_i} - b - \epsilon - \hat{\xi}_i) \\ &+ \sum_i \eta_i \xi_i \\ &+ \sum_i \hat{\eta_i} \hat{\xi}_i \end{aligned} \tag{8}$$

Applying KKT conditions and taking derivatives of L with respect to the primal variables, we obtain

$$\begin{aligned} \boldsymbol{\omega} &= \sum_i (\hat{\alpha_i} - \alpha_i)\boldsymbol{x_i} \\ \sum_i (\alpha_i - \hat{\alpha_i}) &= 0 \\ C - \alpha_i - \eta_i &= 0 \\ C - \hat{\alpha_i} - \hat{\eta_i} &= 0 \end{aligned} \tag{9}$$

Subtituing back into the primal problem, we obtain the following dual problem

$$\begin{aligned} \max_{\alpha_i, \hat{\alpha_i}} \quad & -\frac{1}{2}\sum_{ij}(\hat{\alpha_i} - \alpha_i)(\hat{\alpha_j} - \alpha_j)\boldsymbol{x_i}^\intercal \boldsymbol{x}_j - \epsilon\sum_i(\hat{\alpha_i} + \alpha_i) + \sum_i y_i(\hat{\alpha_i} - \alpha_i) \\ s.t. \quad & \sum_i(\hat{\alpha_i} - \alpha_i) = 0 \\ & \alpha_i, \hat{\alpha_i} \in [0, C] \end{aligned} \tag{10}$$

3 IRLS for Logistic Regression

Let $\boldsymbol{\mu}$ denotes the output predictions for all samples, $\boldsymbol{X}$ denotes inputs for all samples, where each row corresponds to each sample, $\boldsymbol{R}$ denotes $diag(\mu_i(1-\mu_i))$. The negative log-likelihood is

$$NLL(\boldsymbol{\omega}) = -\log \prod_{i=1}^{N} P_{\boldsymbol{\omega}}(y_i|\boldsymbol{x}_i) = -\log \prod_{i=1}^{N} \frac{\exp(y_i \boldsymbol{\omega}^\intercal \boldsymbol{x}_i)}{1+\exp(\boldsymbol{\omega}^\intercal \boldsymbol{x}_i)} = -\sum_{i=1}^{N} [y_i \boldsymbol{\omega}^\intercal \boldsymbol{x}_i - \log(1 + exp(\boldsymbol{\omega}^\intercal \boldsymbol{x}_i))]$$

Taking the gradient of the negative log-likelihood function with respect to $\boldsymbol{\omega}$, we obtain

$$\boldsymbol{g}(\boldsymbol{\omega}) = \nabla NLL(\boldsymbol{\omega}) = -\sum_{i=1}^{N} [y_i \boldsymbol{x}_i - \frac{\exp(\boldsymbol{\omega}^\intercal \boldsymbol{x}_i)}{1+\exp(\boldsymbol{\omega}^\intercal \boldsymbol{x}_i)} \boldsymbol{x}_i] = -\sum_{i=1}^{N} [(y_i - \boldsymbol{\mu}_i)\boldsymbol{x}_i] = \boldsymbol{X}^\intercal(\boldsymbol{\mu} - \boldsymbol{y})$$

Similarly, the hessian of the negative log-likelihood function with respect to $\boldsymbol{\omega}$ is given by following

$$\boldsymbol{H}(\boldsymbol{\omega}) = \sum_{i=1}^{N} [\boldsymbol{\mu}_i(1-\boldsymbol{\mu}_i)\boldsymbol{x}_i \boldsymbol{x}_i^\intercal] = \boldsymbol{X}^\intercal \boldsymbol{R} \boldsymbol{X}$$

As for the l_2-norm regularized logistic regress, the gradient and hessian of the negative log-likelihood function are given by

$$\boldsymbol{g}(\boldsymbol{\omega}) = \boldsymbol{X}^\intercal(\boldsymbol{\mu} - \boldsymbol{y}) + \lambda \boldsymbol{\omega}$$

$$\boldsymbol{H}(\boldsymbol{\omega}) = \boldsymbol{X}^\intercal \boldsymbol{R} \boldsymbol{X} + \lambda \boldsymbol{I}$$

The Newton-Raphson update is as follows

$$\begin{aligned} \boldsymbol{\omega}_{k+1} &= \boldsymbol{\omega}_k - \boldsymbol{H}_k^{-1} g_k \\ &= \boldsymbol{\omega}_k + (\boldsymbol{X}^\intercal \boldsymbol{R} \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal (\boldsymbol{y} - \boldsymbol{\mu}) \\ &= (\boldsymbol{X}^\intercal \boldsymbol{R} \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal (\boldsymbol{R}\boldsymbol{X}\boldsymbol{\omega}_k + \boldsymbol{y} - \boldsymbol{\mu}) \\ &= (\boldsymbol{X}^\intercal \boldsymbol{R} \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{R} \boldsymbol{z}_k \end{aligned} \tag{11}$$

where $\boldsymbol{z}_k = \boldsymbol{X}\boldsymbol{\omega}_k + \boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{\mu})$ As for the l_2-norm regularized logistic regress, the update formula is as follows

$$\begin{aligned} \boldsymbol{\omega}_{k+1} &= \boldsymbol{\omega}_k - \boldsymbol{H}_k^{-1} g_k \\ &= \boldsymbol{\omega}_k + (\boldsymbol{X}^\intercal \boldsymbol{R} \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} (\boldsymbol{X}^\intercal (\boldsymbol{y} - \boldsymbol{\mu}) + \lambda \boldsymbol{\omega}_k) \end{aligned} \tag{12}$$

The IRLS is implemented in Python using numpy for matrix computations. Please refer to the source code if your are interested in the implementation. Here we present the experiment results.

Experiment settings: The l2 penalty $\lambda = 0.1$ was chosen after running 5-fold cross validations over $\lambda \in [0.001, 0.01, 0.1, 1, 10, 100]$ on the training set. The stopping criteria was set as $\|\boldsymbol{\omega}\|^2 < 0.001$. We also adopt some techniques to ensure the numerical stability of the implemented IRLS, which includes *the log-sum-exp trick* and pseudo inverse of the hessian matrices (Although they are theoretically positive semi-definite, the numerical matrix computations make them singular in practice.)

The following table shows the performances of the two models on the given dataset.

	train accuracy	test accuracy	number of iterations	l2-norm of weights
LR	0.84914	0.84995	29	53.91309
LR with l2 penalty	0.84911	0.84988	8	7.90651

As shown in the table, both models achieve promising training accuracy and generalize well. However the l2 regularized LR takes less iterations to converges and the l2-norm of the learned weights are smaller. The converging processes are shown as follows

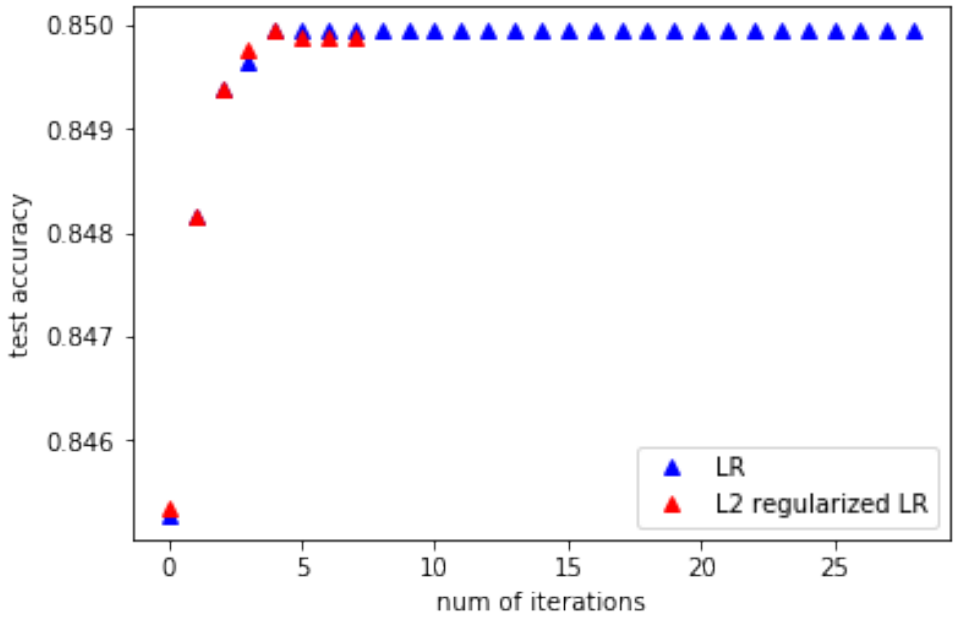

Figure 1: Test Accuracy V.S. Number of Iterations

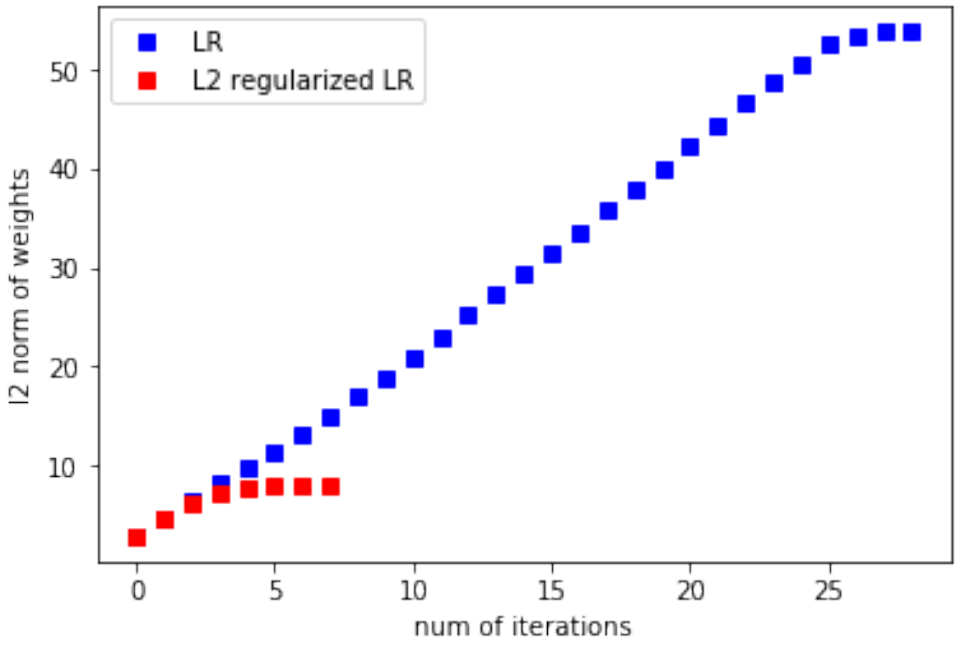

Figure 2: L2 Norm of the Learned Weights V.S. Number of Iterations