CS 229 Problem Set 1 Solutions

Justina Žurauskienė
SUNet:

2023 Summer

Problem 1: Linear Classifiers (logistic regression and GDA)

a) **[10 points]**

In lecture we saw the average empirical loss for logistic regression:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right),$$

where $y^{(i)} \in \{0,1\}$, $h_\theta(x) = g(\theta^T x)$, and $g(z) = \frac{1}{1+e^{-z}}$.

Find the Hessian H of this function, and show that for any vector z, it holds true that $z^T H z \geq 0$.

Hint: You may want to start by showing that

$$\sum_i \sum_j z_i x_i x_j z_j = (x^T z)^2 \geq 0.$$

Recall also that $g'(z) = g(z)(1-g(z))$.

Remark: This is one of the standard ways of showing that the matrix H is positive semidefinite, written as "$H \succeq 0$." This implies that J is convex and has no local minima other than the global one. If you have some other way of showing $H \succeq 0$, you're also welcome to use your method instead of the one above.

Answer: Given the negative *log*-likelihood for logistic regression as,

$$J(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)}\ln[g(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)})] + (1-y^{(i)})\ln[1-g(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)})]\right);$$

the task is to compute the Hessian matrix, H, and show that it fulfills PSD property $z^T H z \geq 0$. For this we will re-arrange the loss function to be in a more convenient form; for simplicity we will use this notation $h_\theta \equiv g(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)})$ and $g(z) = 1/(1+\exp\{-z\})$ is sigmoid function (in bold we denote vectors) thus,

$$J(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)}\ln[h_\theta] - y^{(i)}\ln[1-h_\theta] + \ln[1-h_\theta]\right) =$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)}\ln\left[\frac{h_\theta}{1-h_\theta}\right] + \ln[1-h_\theta]\right).$$

Because

$$1 - h_\theta = \frac{\exp(-\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})}{1 + \exp(-\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})} \quad \Rightarrow \quad \frac{h_\theta}{1 - h_\theta} = \frac{1}{\exp(-\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})} = \exp(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}), \quad \text{and}$$

$$h_\theta = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})} = \frac{\exp(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})}{\exp(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}) + 1} \quad \Rightarrow \quad 1 - h_\theta = \frac{1}{\exp(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}) + 1}.$$

Therefore, using: above facts, logarithm power rule and ln/exp inverse property, we get cost defined as follows,

$$\boxed{J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left(\ln \left[1 + \exp(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}) \right] - y^{(i)} \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} \right).}$$

We first compute gradient followed by the second derivative - Hessian; we use equation (69) from Matrix Cookbook: $\frac{\partial \boldsymbol{x}^T \boldsymbol{a}}{\partial \boldsymbol{x}} = \boldsymbol{a}$,

$$\nabla_\theta J(\boldsymbol{\theta}) := J'(\boldsymbol{\theta}) \underbrace{=}_{eq(69)} \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\exp(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})}{1 + \exp(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})} \boldsymbol{x^{(i)}} - y^{(i)} \boldsymbol{x^{(i)}} \right) = \frac{1}{n} \sum_{i=1}^{n} \left(h_\theta - y^{(i)} \right) \boldsymbol{x}^{(i)}.$$

Provided, that $h'_\theta = h_\theta(1 - h_\theta)\boldsymbol{x^{(i)}}$, and taking derivative of the gradient expression, we get,

$$\boldsymbol{H}_\theta J(\boldsymbol{\theta}) := J''(\boldsymbol{\theta}) \underbrace{=}_{eq(69)} \frac{1}{n} \sum_{i=1}^{n} \left(h_\theta(1 - h_\theta) \boldsymbol{x^{(i)}} \boldsymbol{x^{(i)}}^T \right).$$

Now we will show that $z^T H z \geq 0$.

We will first proof a supporting proposition:

Let $A \in \mathrm{R}^{n \times n}$ and $B \in \mathrm{R}^{n \times n}$ be two matrices. If A and B are positive semi-definite, then their sum, $A + B$, is also positive semi-definite.
Proof: A symmetric matrix M is positive semi-definite, if for all vectors $x \in \mathrm{R}^n$, the quadratic form $x^T M x \geq 0$ (non-negative). Let $x^T A x \geq 0$ and $x^T B x \geq 0$; Then, if $D = A + B$, we have

$$x^T D x = x^T (A + B) x = \underbrace{x^T A x}_{\geq 0} + \underbrace{x^T B x}_{\geq 0} \geq 0$$

since both added terms are non-negative, it follows that $A + B$ is also PSD.

For Hessian matrix defined as,

$$\boxed{H = \frac{1}{n} \sum_{i=1}^{n} \left(h_\theta(1 - h_\theta) \boldsymbol{x^{(i)}} \boldsymbol{x^{(i)}}^T \right),}$$

we notice that it involves two probability terms, since probabilities cannot be negative the following term $h_\theta(1 - h_\theta) \geq 0$ is always non-negative. Further, we have proven before that matrix,

constructed as an outer product is always PSD (see PS0 task 2(a)). Therefore, summing up a number of PSD matrices that are scaled by non-negative terms will result in PSD, meaning: $z^T H z \geq 0$.

b) **[5 points] Coding problem.** Follow the instructions in `src/linearclass/logreg.py` to train a logistic regression classifier using Newton's Method. Starting with $\theta = \vec{0}$, run Newton's Method until the updates to θ are small: Specifically, train until the first iteration k such that $\|\theta_k - \theta_{k-1}\|_1 < \epsilon$, where $\epsilon = 1 \times 10^{-5}$. Make sure to write your model's predicted probabilities on the validation set to the file specified in the code.

Include a plot of the validation data with x_1 on the horizontal axis and x_2 on the vertical axis. To visualize the two classes, use a different symbol for examples $x^{(i)}$ with $y^{(i)} = 0$ than for those with $y^{(i)} = 1$. On the same figure, plot the decision boundary found by logistic regression (i.e., line corresponding to $p(y|x) = 0.5$).

Note: If you want to print the loss during training, you may encounter some numerical instability issues. Recall that the loss function on an example (x, y) is defined as $y \log(h_\theta(x)) + (1-y) \log(1 - h_\theta(x))$, where $h_\theta(x) = (1 + \exp(-x^\top \theta))^{-1}$. Technically speaking, $h_\theta(x) \in (0, 1)$ for any θ, $x \in \mathbb{R}^d$. However, in Python a real number only has finite precision. So it is possible that in your implementation, $h_\theta(x) = 0$ or $h_\theta(x) = 1$, which makes the loss function ill-defined. A typical solution to the numerical instability issue is to add a small perturbation. In this case, you can compute the loss function using $y \log(h_\theta(x) + \epsilon) + (1 - y) \log(1 - h_\theta(x) + \epsilon)$ instead, where ϵ is a very small perturbation (for example, $\epsilon = 10^{-5}$).

Answer:

Below are included results obtained when training logistic regression classifier using Newton's Method. In figures you can see data from validation set plotted, where different colours correspond to two major classes; identified linear decision boundary separating these classes is given in red.

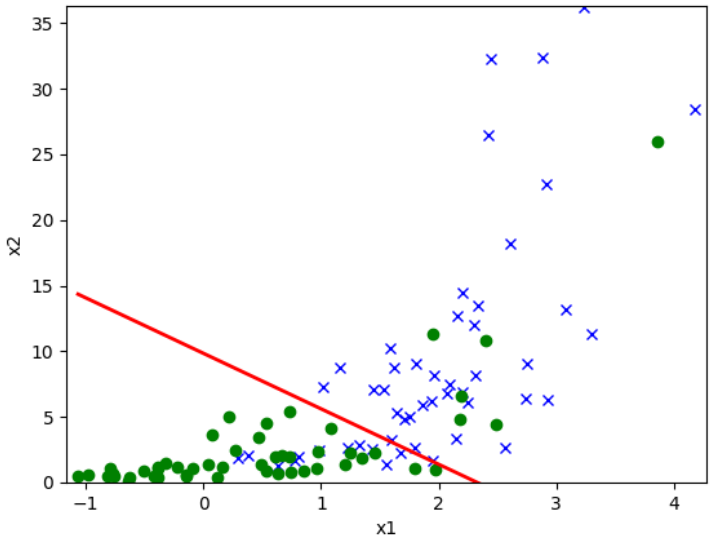

Figure 1: Decision boundary plot corresponding to ds1_{valid} dataset using LR approach.

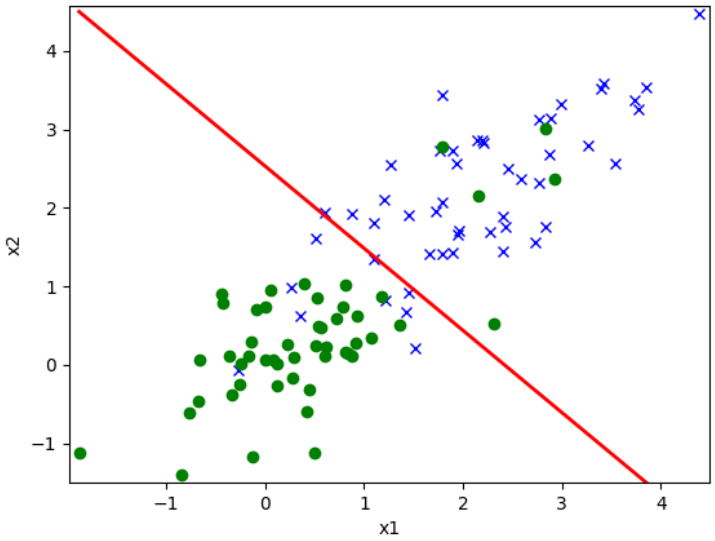

Figure 2: Decision boundary plot corresponding to ds2_{valid} dataset using LR approach.

c) **[5 points]** Recall that in GDA we model the joint distribution of (x, y) by the following equations:

$$p(y) = \begin{cases} \phi & \text{if } y = 1 \\ 1 - \phi & \text{if } y = 0 \end{cases} \quad (1)$$

$$p(x|y=0) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^\top \Sigma^{-1}(x-\mu_0)\right) \quad (2)$$

$$p(x|y=1) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^\top \Sigma^{-1}(x-\mu_1)\right)$$

where ϕ, μ_0, μ_1, and Σ are the parameters of our model.

Suppose we have already fit ϕ, μ_0, μ_1, and Σ, and now want to predict y given a new point x. To show that GDA results in a classifier that has a linear decision boundary, show that the posterior distribution can be written as:

$$p(y=1|x;\phi,\mu_0,\mu_1,\Sigma) = \frac{1}{1+\exp(-(x^\top\theta+\theta_0))},$$

where $\theta \in \mathbb{R}^d$ and $\theta_0 \in \mathbb{R}$ are appropriate functions of ϕ, Σ, μ_0, and μ_1. State the value of θ and θ_0 as a function of ϕ, μ_0, μ_1, and Σ explicitly.

Answer:

Here we will show that, GDA results in a classifier that has a linear decision boundary, by proving that,

$$p(y=1|x;\Sigma,\mu_0,\mu_1) = \frac{1}{1+\exp(-\theta^T x)}.$$

Considering two classes, the posterior probability can be expressed using Bayes rule, i.e. taking joint distribution and dividing it by total probability, $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$. In GDA case, joint distribution is (the "either class" we represent with $\mu_{y_i}, i = \{0, 1\}$) given as,

$$p(y, x|\mu_0, \mu_1, \Sigma, \phi) = p(x|y, \mu_0, \mu_1, \Sigma) \cdot p(y|\phi) =$$

$$= \underbrace{\left[\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_y)^T\Sigma^{-1}(x-\mu_y)\right)\right]}_{=:p(x|y)} \cdot \underbrace{\phi^y}_{=:p(y_1)=\phi;} \cdot \underbrace{(1-\phi)^{1-y}}_{=:p(y_0)=1-\phi}.$$

Using above notations we will express posterior probability for class $y = 1$. For this, we will follow logic presented in C.M.Bishop Pattern recognition book, eq(4.57) to show that decision boundary is linear. Thus,

$$p(y_1|x) = \frac{p(x|y_1)p(y_1)}{p(x|y_1)p(y_1) + p(x|y_0)p(y_0)} = \Bigg| \quad \text{using fact: } \frac{a}{b} = \frac{1}{\frac{b}{a}} \quad \text{we re-arrange right side accordinngly}$$

$$= \frac{1}{\frac{p(x|y_1)p(y_1)+p(x|y_0)p(y_0)}{p(x|y_1)p(y_1)}} = \frac{1}{\frac{p(x|y_1)p(y_1)}{p(x|y_1)p(y_1)} + \frac{p(x|y_0)p(y_0)}{p(x|y_1)p(y_1)}} = \frac{1}{1 + \frac{p(x|y_0)p(y_0)}{p(x|y_1)p(y_1)}}.$$

To put $\frac{p(x|y_0)p(y_0)}{p(x|y_1)p(y_1)}$ under exponent, we can use this re-arrangement,

$$\frac{a}{b} = \exp\ln\frac{a}{b} = \exp\ln\left(\frac{b}{a}\right)^{-1} = \exp\ln\left(\frac{b}{a}\right)^{-1} = \exp\left(-\ln\frac{b}{a}\right);$$

leading to,

$$p(y_1|x) = \frac{1}{1 + \frac{p(x|y_0)p(y_0)}{p(x|y_1)p(y_1)}} = \frac{1}{1 + \exp(-z)}, \quad \text{where } z = \ln\frac{p(x|y_1)p(y_1)}{p(x|y_0)p(y_0)} \quad \text{(also see eq(4.58)).}$$

Thus, under the assumption of GDA the posterior distribution $p(y = 1|x; \Sigma, \mu_0, \mu_1)$ takes the form of a logistic sigmoid function. Now, we will show that $z = \theta^T x + \theta_0$,

$$\ln\frac{p(x|y_1)p(y_1)}{p(x|y_0)p(y_0)} = \ln\left[\frac{\frac{1}{(2\pi)^{d/2}|\Sigma|^{0.5}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)}{\frac{1}{(2\pi)^{d/2}|\Sigma|^{0.5}} \exp\left(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right)} \cdot \frac{\phi}{1-\phi}\right] =$$

$$= \frac{1}{2}\left[(x-\mu_0)^T\Sigma^{-1}(x-\mu_0) - (x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right] + \ln\frac{\phi}{(1-\phi)} =$$

$$= \frac{1}{2}\left[(x^T\Sigma^{-1}x - 2\mu_0^T\Sigma^{-1}x + \mu_0\Sigma^{-1}\mu_0) - (x^T\Sigma^{-1}x - 2\mu_1^T\Sigma^{-1}x + \mu_1\Sigma^{-1}\mu_1)\right] + \ln\frac{\phi}{(1-\phi)} =$$

$$= \frac{1}{2}\left[x^T\Sigma^{-1}x - 2\mu_0^T\Sigma^{-1}x + \mu_0\Sigma^{-1}\mu_0 - x^T\Sigma^{-1}x + 2\mu_1^T\Sigma^{-1}x - \mu_1\Sigma^{-1}\mu_1\right] + \ln\frac{\phi}{(1-\phi)} =$$

$$= \underbrace{(m_1-\mu_0)^T\Sigma^{-1}}_{=:\theta^T} x + \underbrace{0.5\left[(\mu_0\Sigma^{-1}\mu_0 - \mu_1\Sigma^{-1}\mu_1) + \ln\frac{\phi}{(1-\phi)}\right]}_{=:\theta_0} = \theta^T x + \theta_0,$$

where $\boxed{\theta^T = (m_1-\mu_0)^T\Sigma^{-1}}$ and $\boxed{\theta_0 = 0.5\left[(\mu_0\Sigma^{-1}\mu_0 - \mu_1\Sigma^{-1}\mu_1) + \ln\frac{\phi}{(1-\phi)}\right].}$

d) **[7 points]** Given the dataset, we claim that the maximum likelihood estimates of the parameters are given by:

$$\phi = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y^{(i)} = 1\} \quad (3)$$

$$\mu_0 = \frac{\sum_{i=1}^{n} \mathbf{1}\{y^{(i)} = 0\} x^{(i)}}{\sum_{i=1}^{n} \mathbf{1}\{y^{(i)} = 0\}} \quad (4)$$

$$\mu_1 = \frac{\sum_{i=1}^{n} \mathbf{1}\{y^{(i)} = 1\} x^{(i)}}{\sum_{i=1}^{n} \mathbf{1}\{y^{(i)} = 1\}} \quad (5)$$

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^\top$$

The log-likelihood of the data is:

$$\ell(\phi, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^{n} p(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \quad (6)$$

$$= \log \prod_{i=1}^{n} p(x^{(i)} | y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi)$$

By maximizing ℓ with respect to the four parameters, prove that the maximum likelihood estimates of ϕ, μ_0, μ_1, and Σ are indeed as given in the formulas above. (You may assume that there is at least one positive and one negative example, so that the denominators in the definitions of μ_0 and μ_1 above are non-zero.)

Answer:

To derive formulas for the maximum likelihood (ML) estimates of the parameters used in the Gaussian Discriminant Analysis, we need to differentiate the *log*-likelihood function with respect to the parameters and set obtained derivatives to zero. Thus, the *log*-likelihood is given as,

$$\begin{aligned}
\mathscr{L}(\theta) &= \ln \prod_{i=1}^{n} p(x^{(i)} | y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi) = \\
&= \ln \prod_{i=1}^{n} \left[\frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (x^{(i)} - \mu_{y^{(i)}})^T \Sigma^{-1} (x^{(i)} - \mu_{y^{(i)}}) \right) \right] \phi^{y^{(i)}} (1-\phi)^{1-y^{(i)}} = \\
&= \sum_{i=1}^{n} \ln \left[\frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (x^{(i)} - \mu_{y^{(i)}})^T \Sigma^{-1} (x^{(i)} - \mu_{y^{(i)}}) \right) \phi^{y^{(i)}} (1-\phi)^{1-y^{(i)}} \right] = \\
&= -N - \frac{1}{2} \sum_{i=1}^{n} (x^{(i)} - \mu_{y^{(i)}})^T \Sigma^{-1} (x^{(i)} - \mu_{y^{(i)}}) + \sum_{i=1}^{n} y^{(i)} \ln \phi + (1 - y^{(i)}) \ln(1-\phi),
\end{aligned}$$

where we denoted $N = \frac{nd}{2} \ln(2\pi) + \frac{n}{2} \ln |\Sigma|$ and $\theta = (\phi, \mu_0, \mu_1, \Sigma)$

To derive update rules we will collect all terms involving parameter of interest from $\mathscr{L}(\theta)$, compute derivatives and solve it for 0 (we will ignore irrelevant terms as their derivative with respect to parameter of interest will be 0). Thus,

- **Update rule for ϕ.**

$$\frac{\partial}{\partial\phi}\left[\sum_{i=1}^{n} y^{(i)}\ln\phi + (1-y^{(i)})\ln(1-\phi)\right] = 0$$
$$\frac{1}{\phi}\sum_{i=1}^{n} y^{(i)} - \frac{1}{1-\phi}\sum_{i=1}^{n}(1-y^{(i)}) = 0,$$

Denoting $a := \sum_{i=1}^{n} y^{(i)}$ and $b := \sum_{i=1}^{n}(1-y^{(i)})$, we get,

$$\frac{a}{\phi} = \frac{b}{1-\phi} \quad \Rightarrow \quad a - a\phi = b\phi \quad \Rightarrow \quad \phi = \frac{a}{b+a} = \frac{\sum_{i=1}^{n} y^{(i)}}{\sum_{i=1}^{n} y^{(i)} + 1 - y^{(i)}} = \frac{1}{n}\sum_{i=1}^{n} y^{(i)}; \quad \Rightarrow$$

$$\boxed{\phi = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{y^{(i)}=1}}$$

- **Update rule for μ_0 and μ_1.**

$$\frac{\partial}{\partial\mu_{y^{(i)}}}\left[-\frac{1}{2}\sum_{i=1}^{n}(x^{(i)}-\mu_{y^{(i)}})^T\Sigma^{-1}(x^{(i)}-\mu_{y^{(i)}})\right] = 0$$
$$-\frac{1}{2}\sum_{i=1}^{n}\frac{\partial}{\partial\mu_{y^{(i)}}}(x^{(i)}-\mu_{y^{(i)}})^T\Sigma^{-1}(x^{(i)}-\mu_{y^{(i)}}) = 0$$

From [PS0q1a] we recall that $\frac{1}{2}x^TAx = Ax$ therefore

$$-\sum_{i=1}^{n}\Sigma^{-1}(x^{(i)}-\mu_{y^{(i)}})(-1) = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} x^{(i)} = \sum_{i=1}^{n}\mu_{y^{(i)}} \quad \Rightarrow$$

Re-arranging we have

$$\boxed{\mu_0 = \frac{\sum_{i=1}^{n} x^{(i)}\mathbf{1}_{y^{(i)}=0}}{\sum_{i=1}^{n}\mathbf{1}_{y^{(i)}=0}} \quad \text{when } y=0;} \quad \text{and} \quad \boxed{\mu_1 = \frac{\sum_{i=1}^{n} x^{(i)}\mathbf{1}_{y^{(i)}=1}}{\sum_{i=1}^{n}\mathbf{1}_{y^{(i)}=1}} \quad \text{when } y=1;}$$

- **Update rule for Σ.**

$$-\frac{\partial}{\partial\Sigma}N - \frac{1}{2}\sum_{i=1}^{n}\frac{\partial}{\partial\Sigma}(x^{(i)}-\mu_{y^{(i)}})^T\Sigma^{-1}(x^{(i)}-\mu_{y^{(i)}}) = 0$$

We will break this down by firstly computing derivatives for N term, that contain Σ. We will use result form Matrix Cookbook (under 2.1.4 Other nonlinear forms section), $\frac{\partial\ln|det(X)|}{\partial X} = (X^{-1})^T = (X^T)^{-1}$;

$$\frac{\partial}{\partial\Sigma}N = \frac{n}{2}\frac{\partial}{\partial\Sigma}\ln|\Sigma| = \frac{n}{2}\Sigma^{-1}.$$

We next compute derivative of second term; since Σ is square and convertible, we will use this result, $\frac{\partial a^T X^{-1} b}{\partial X} = -X^{-T} a b^T X^{-T}$, from matrix cookbook, see page 8. Therefore,

$$\sum_{i=1}^{n} \frac{\partial}{\partial \Sigma} (x^{(i)} - \mu_{y^{(i)}})^T \Sigma^{-1} (x^{(i)} - \mu_{y^{(i)}}) = -\sum_{i=1}^{n} \Sigma^{-T} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T \Sigma^{-T} \quad \Rightarrow$$

Combining we have,

$$\frac{1}{2} \sum_{i=1}^{n} \Sigma^{-1} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T \Sigma^{-1} = \frac{n}{2} \Sigma^{-1}$$

$$\Sigma^{-1} = \frac{1}{n} \Sigma^{-1} \Sigma^{-1} \sum_{i=1}^{n} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T \quad | \text{ multiply both sides by } \Sigma \text{ twice,}$$

$$\Rightarrow \quad \boxed{\Sigma = \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T.}$$

e) **[5 points] Coding problem.** In `src/linearclass/gda.py`, fill in the code to calculate ϕ, μ_0, μ_1, and Σ, use these parameters to derive θ, and use the resulting GDA model to make predictions on the validation set. Make sure to write your model's predictions on the validation set to the file specified in the code.

Include a plot of the validation data with x_1 on the horizontal axis and x_2 on the vertical axis. To visualize the two classes, use a different symbol for examples $x^{(i)}$ with $y^{(i)} = 0$ than for those with $y^{(i)} = 1$. On the same figure, plot the decision boundary found by GDA (i.e., line corresponding to $p(y|x) = 0.5$).

Answer:

Here we have implemented GDA model using dataset, ds1_{train,valid}. Results are summarized in the following Figure 3

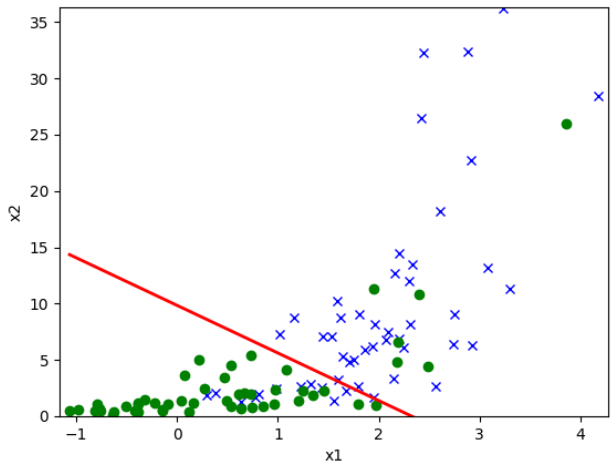

Figure 3: Decision boundary plot corresponding to ds1_{valid} dataset using GDA approach.

f) **[2 points]** For Dataset 1, compare the validation set plots obtained in part (b) and part (e) from logistic regression and GDA respectively, and briefly comment on your observation in a couple of lines.

Answer:

Here we see that for ds1 - GDA shows poorer performance when compared to LR, as modelling assumptions made by GDA do not represent data characteristics (see corresponding plots given in first column of Figure 4).

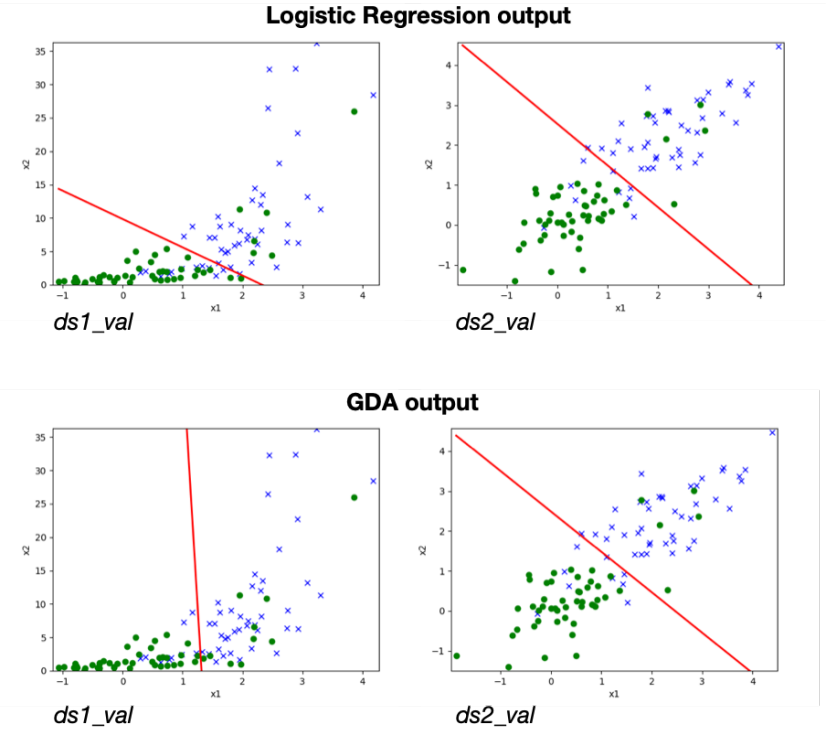

Figure 4: Decision boundary plots corresponding to ds{1, 2}_{valid} datasets using LR and GDA methods.

g) **[5 points]** Repeat the steps in part (b) and part (e) for Dataset 2. Create similar plots on the validation set of Dataset 2 and include those plots in your writeup. On which dataset does GDA seem to perform worse than logistic regression? Why might this be the case?

Answer:

Results from both LR and GDA models using second dataset are summarize in Figure 4 (see corresponding plots in second column).

From plots, we can see that performance of (GDA) and (LR) depend on specific characteristics of data.

GDA makes stronger modeling assumptions (the inputs for each class are drawn from a Gaussian distribution with a common covariance matrix). When these assumptions hold true, GDA can perform very well. However, when these assumptions are violated (underlying data is non-Gaussian), GDA's performance can suffer.

Logistic Regression makes weaker assumptions, i.e. there are no assumptions that inputs follow Gaussian patterns. Therefore, LR can be more robust to inaccuracies of modeling assumptions, and in turn achieve better performance on data that is non-Gaussian (see main notes, 4.1.3 Discussion: GDA and logistic regression).

In the analyzed scenario, it is evident that dataset ds1 deviated from the Gaussian distribution assumption (this can be tested using the Shapiro–Wilk test of normality), resulting in superior performance of the LR model over GDA. Whereas for dataset ds2, the data more closely met GDA's modeling assumptions, leading to comparable performance between both - GDA and LR.

h) **[1 points]** For the dataset where GDA performed worse in parts (f) and (g), can you find a transformation of the $x^{(i)}$'s such that GDA performs significantly better? What might this transformation be?

Answer:

We could apply Box-Cox transformation of input features, which belongs to power transform family. Box-Cox transformation has a parameter lambda (λ) that can be tuned to find the transformation that makes original data look more like Gaussian. We would apply this transformation separately to each feature (i.e. to each dimension in X).

Inspecting plots, we see that our ds1_valid has negative values in x_1 dimension, we would need to shift it to a positive range (e.g. by adding a constant value greater than $\min(x_1)$), const=2 would work here) for Box-Cox to work.

Transformation parameters, λ, we would estimate using training data and then apply identified transformation to the validation data. This could be done for validation set to "mimic" unseen data and also to avoid using data twice (for model building and validation).

We then could re-apply GDA on transformed data to inspect if the outcome improved.

Please see plots below that illustrate the effects of Box-Cox transformation applied on validation dataset only.

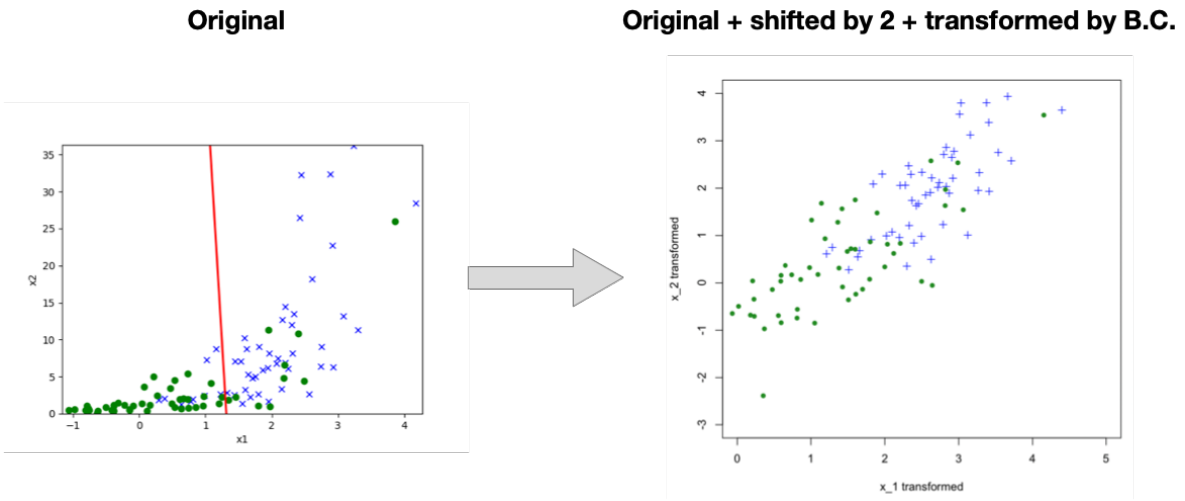

Figure 5: Illustration of Box-Cox transformation using ds1_valid dataset.

Plot in Figure 5 on the right resembles data pattern/trend observed in ds2_valid dataset, where

we know that GDA performs reasonably well. Thus, we could reason, that this would apply to transformed dataset too, meaning we would significantly boost GDA's performance.

Problem 2: Poisson Regression

a) **[5 points]** Consider the Poisson distribution parameterized by λ:

$$p(y;\lambda) = \frac{e^{-\lambda}\lambda^y}{y!}.$$

(Here y has positive integer values and $y!$ is the factorial of y.) Show that the Poisson distribution is in the exponential family, and clearly state the values for $b(y)$, η, $T(y)$, and $a(\eta)$.

Answer:

General form of exponential family is given by,

$$p(y;\eta) = b(y)\exp\{\eta \cdot T(y) - a(\eta)\},$$

where η is natural parameter; $b(y)$ is base measure; $T(y)$ is sufficient statistics of data; and $a(\eta)$ is *log*-partition function. We will re-arrange Poisson distribution to be in this format; thus,

$$p(y;\lambda) = \frac{e^{-\lambda}\lambda^y}{y!} = \frac{1}{y!}\exp\{-\lambda\}\cdot\exp\{\ln(\lambda^y)\} = \boxed{\underbrace{\frac{1}{y!}}_{=:b(y)}\exp\{\underbrace{\ln\lambda}_{=:\eta}\cdot\underbrace{y}_{=:T(y)} - \underbrace{\lambda}_{=:a(\eta)}\}.}$$

b) **[3 points]** Consider performing regression using a GLM model with a Poisson response variable. What is the canonical response function for the family? (You may use the fact that a Poisson random variable with parameter λ has mean λ.)

Answer:

From GLM key assumptions we have:

- Given x and θ, the distribution of y follows Poison exponential family distribution, with parameter η:

$$y|x,\theta \sim b(y)\exp\{\eta\cdot T(y) - a(\eta)\}$$

- Hypothesis h satisfies: $h(\theta) = E(y|x) = \lambda$
- Natural parameters η and inputs x are related linearly: $\eta = \theta^T x$.

Further, we have shown that $\eta = \ln\lambda$; therefore, re-arranging it we get that $\lambda = \exp\{\eta\}$ and in turn canonical response function is given by:

$$h(\theta) = E(y|x) = \boxed{\lambda = \exp\{\theta^T x\}.}$$

c) **[7 points]** For a training set $\{(x^{(i)}, y^{(i)}); i = 1,\ldots,n\}$, let the log-likelihood of an example be $\log p(y^{(i)}|x^{(i)};\theta)$. By taking the derivative of the log-likelihood with respect to θ_j, derive the stochastic gradient ascent update rule for learning using a GLM model with Poisson responses y and the canonical response function.

Answer:

A Poisson distribution with mean λ has probability mass function (PMF) given by:

$$p(y;\lambda) = \frac{e^{-\lambda}\lambda^y}{y!}$$

Because we will use stochastic gradient ascent, we should compute the gradient and update model parameters using only single training example at a time (as compared to summing over the whole training set using regular (batch) gradient ascent). Thus, for a single training example $(x^{(i)}, y^{(i)})$, the *log*-likelihood of the Poisson distribution can be written as:

$$\mathscr{L}(\theta) \equiv -\lambda + y^{(i)} \ln \lambda - \ln(y^{(i)}!).$$

Recalling that $\lambda = \exp\{\theta^T x\}$, we rewrite above *log*-likelihood using θ's,

$$\mathscr{L}(\theta) \equiv -\exp\{\theta^T x^{(i)}\} + y^{(i)} \ln(\exp\{\theta^T x^{(i)}\}) - \ln(y^{(i)}!) = y^{(i)}\theta^T x^{(i)} - \exp\{\theta^T x^{(i)}\} - \ln(y^{(i)}!)$$

We can now derive the update rule for a single component θ_j; for this we begin by first computing gradient with respect to θ_j, which is,

$$\frac{\partial}{\partial \theta_j}\mathscr{L}(\theta) = y^{(i)}x_j^{(i)} - \exp\{\theta^T x^{(i)}\} \cdot x_j^{(i)} = x_j^{(i)}\left[y^{(i)} - \exp\{\theta^T x^{(i)}\}\right]$$

Finally, the stochastic gradient ascent update rule will be,

$$\boxed{\theta_j := \theta_j + \alpha \cdot \left[y^{(i)} - \exp\{\theta^T x^{(i)}\}\right] x_j^{(i)}.}$$

d) **[10 points] Coding problem**

Consider a website that wants to predict its daily traffic. The website owners have collected a dataset of past traffic to their website, along with some features which they think are useful in predicting the number of visitors per day. The dataset is split into train/valid sets, and the starter code is provided in the following files:

- `src/poisson/{train,valid}.csv`
- `src/poisson/poisson.py`

We will apply Poisson regression to model the number of visitors per day. Note that applying Poisson regression, in particular, assumes that the data follows a Poisson distribution whose natural parameter is a linear combination of the input features (i.e., $\eta = \theta^T x$). In `src/poisson/poisson.py`, implement Poisson regression for this dataset and use full batch gradient ascent to maximize the log-likelihood of θ. For the stopping criterion, check if the change in parameters has a norm smaller than a small value such as 10^{-5}.

Using the trained model, predict the expected counts for the validation set, and create a scatter plot between the true counts vs. predicted counts on the validation set. In the scatter plot, let the x-axis be the true count and the y-axis be the corresponding predicted expected count. Note that the true counts are integers while the expected counts are generally real values.

Answer:

Here we implemented Poisson regression model and obtained parameter estimates using batch gradient ascent algorithm; results are summarised in Figure 6.

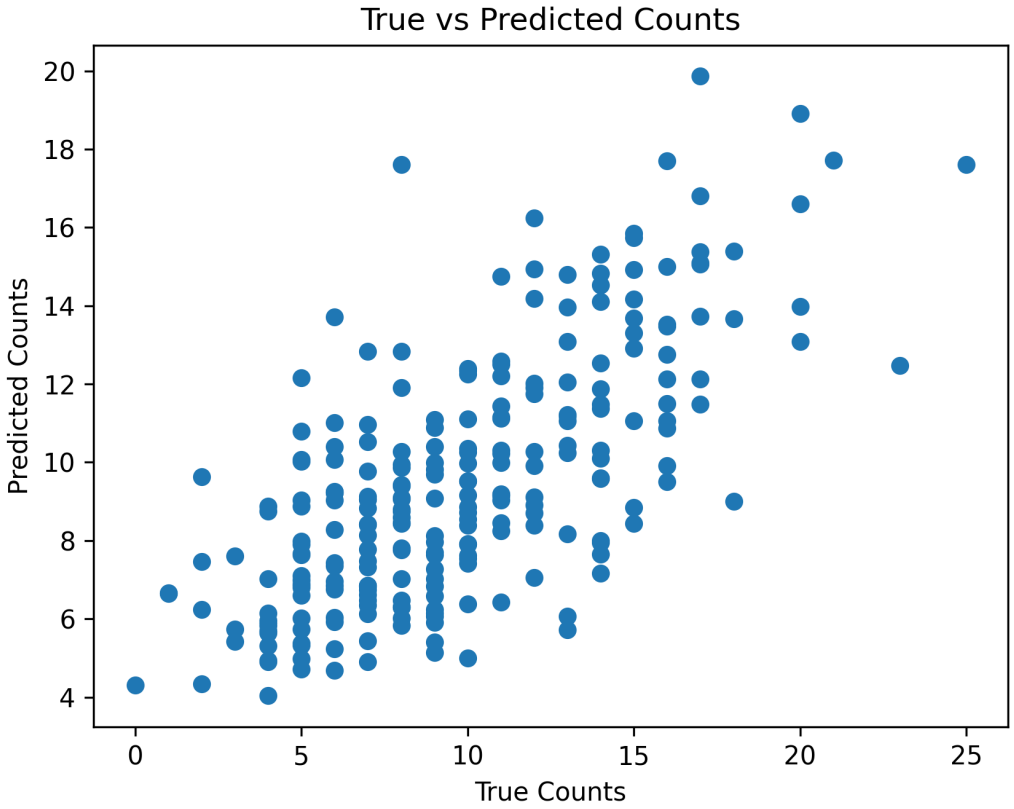

Figure 6: Illustration of Poisson regression model performance using "the number of visitors per day" validation dataset. On x-axis we plot true counts, whereas on y-axis - our predictions.

Problem 3: Convexity of Generalized Linear Models

a) **[5 points]** Derive an expression for the mean of the distribution. Show that $E[Y;\eta] = \frac{\partial}{\partial \eta} a(\eta)$ (note that $E[Y;\eta] = E[Y|X;\theta]$ since $\eta = \theta^\top x$). In other words, show that the mean of an exponential family distribution is the first derivative of the log-partition function with respect to the natural parameter.

Hint: Start by observing that $\frac{\partial}{\partial \eta} \int p(y;\eta) dy = \int \frac{\partial}{\partial \eta} p(y;\eta) dy$.

Answer:

Here we show the relationship between expectation and *log*-partition function. For GLM defined as exponential family $p(y;\eta) = b(y)\exp\{\eta y - a(\eta)\}$, if we assume p is a full probability density function (up to a constant), it should integrate to 1,

$$\int p(y;\eta) dy := \int b(y)\exp\{\eta y - a(\eta)\} dy = 1,$$

Taking $\frac{\partial}{\partial \eta}$ on both sides, and using the following fact,

$$\frac{\partial}{\partial \eta} \int p(y;\eta)\, dy = \int \frac{\partial}{\partial \eta} p(y;\eta)\, dy$$

we get

$$\int b(y) \frac{\partial}{\partial \eta} \exp\{\eta y - a(\eta)\} dy = 0,$$

$$\int b(y) \exp\{\eta y - a(\eta)\} \cdot \left(y - \frac{\partial}{\partial \eta} a(\eta) \right) dy = 0,$$

$$\int y b(y) \exp\{\eta y - a(\eta)\} dy - \int \frac{\partial}{\partial \eta} a(\eta) \cdot y b(y) \exp\{\eta y - a(\eta)\} dy = 0,$$

$$\int y b(y) \exp\{\eta y - a(\eta)\} dy = \frac{\partial}{\partial \eta} a(\eta) \underbrace{\int y b(y) \exp\{\eta y - a(\eta)\} dy}_{=1},$$

$$\underbrace{\int y p(y;\eta)}_{=:\mathrm{E}[y;\eta]} = \frac{\partial}{\partial \eta} a(\eta) \quad \Rightarrow \quad \boxed{\mathrm{E}[y;\eta] = \frac{\partial}{\partial \eta} a(\eta).}$$

b) **[5 points]** Next, derive an expression for the variance of the distribution. In particular, show that $\mathrm{Var}(Y;\eta) = \frac{\partial^2}{\partial \eta^2} a(\eta)$ (again, note that $\mathrm{Var}(Y;\eta) = \mathrm{Var}(Y|X;\theta)$). In other words, show that the variance of an exponential family distribution is the second derivative of the log-partition function with respect to the natural parameter.

Hint: Building upon the result in the previous sub-problem can simplify the derivation.

Answer:

To show the relationship between variance and *log*-partition function, we take results from computing expectation, and apply again $\frac{\partial}{\partial \eta}$ on both sides,

$$\int y b(y) \frac{\partial}{\partial \eta} \exp\{\eta y - a(\eta)\} dy = \frac{\partial^2}{\partial \eta^2} a(\eta)$$

$$\int y^2 b(y) \exp\{\eta y - a(\eta)\} dy - \frac{\partial}{\partial \eta} a(\eta) \int y b(y) \exp\{\eta y - a(\eta)\} dy = \frac{\partial^2}{\partial \eta^2} a(\eta)$$

$$\mathrm{E}[y^2;\eta] - (\mathrm{E}[y;\eta])^2 = \frac{\partial^2}{\partial \eta^2} a(\eta) \quad \Rightarrow \boxed{\mathrm{V}ar[y;\eta] = \frac{\partial^2}{\partial \eta^2} a(\eta).}$$

c) **[5 points]** Finally, write out the loss function $\ell(\theta)$, the negative log-likelihood (NLL) of the distribution, as a function of θ. Then, calculate the Hessian of the loss with respect to θ and show that it is always positive semidefinite (PSD). This concludes the proof that the NLL loss of GLM is convex.

Hint 1: Use the chain rule of calculus along with the results of the previous parts to simplify your derivations.

Hint 2: Recall that the variance of any probability distribution is non-negative.

Remark: The main takeaways from this problem are:

- Any GLM model is convex in its model parameters.
- The exponential family of probability distributions is mathematically nice. While calculating the mean and variance of distributions in general involves integrals (hard), surprisingly, we can calculate them using derivatives (easy) for exponential family.

Answer:

Here, we will write out the loss function $\mathscr{L}(\theta)$, the NLL of the distribution, as a function of (θ). Then, we will calculate the Hessian of the loss w.r.t (θ), and show that it is always PSD. This will conclude the proof that NLL loss of GLM is convex. Thus, for this we first recall that $\eta = \theta^T x$, giving us expression for a single example,

$$p(y^{(i)}, \eta = \theta^T x^{(i)}) = b(y^{(i)}) \exp\left\{\theta^T x^{(i)} \cdot y^{(i)} - a(\theta^T x^{(i)})\right\}.$$

Considering a whole data set $X = \left\{x^{(i)}\right\}_{i=1}^n$, we can write joint distribution by multiplying individual likelihoods; then taking ln and multiplying by (-1), we arrive at NLL, which is given below,

$$\boxed{\mathscr{L}(\theta; X, y) = -\sum_{i=1}^n \ln b(y^{(i)}) - \sum_{i=1}^n \left[\theta^T x^{(i)} \cdot y^{(i)} - a(\theta^T x^{(i)})\right],}$$

which is our loss function. We will now compute Hessian of $\mathscr{L}(\theta; X, y)$, for this we will disregard terms that do not involve parameters θ. Thus, recalling chain rule and useful fact from Matrix Cookbook - eq(69): $\frac{\partial x^T a}{\partial x} = a$, gives us,

$$\nabla_\theta \mathscr{L}(\theta; X, y) = \nabla_\theta \left(\sum_{i=1}^n \left[\theta^T x^{(i)} \cdot y^{(i)} - a(\theta^T x^{(i)})\right]\right) = \sum_{i=1}^n \left(\underbrace{-y^{(i)} x^{(i)}}_{const} + \frac{\partial a(\theta^T x)}{\partial \theta} \cdot x^{(i)}\right) \quad \Rightarrow$$

$$\text{because of chain rule:} \quad \frac{\partial}{\partial \theta} a(\theta^T x) = \left(\frac{\partial a(\theta^T x)}{\partial \underbrace{\theta^T x}_{\eta}} \frac{\partial \theta^T x}{\partial \theta}\right) = \frac{\partial a(\theta^T x)}{\partial \eta} x^{(i)} \quad \Rightarrow$$

$$H_\theta \mathscr{L}(\theta; X, y) = \frac{\partial}{\partial \theta}\left(\sum_{i=1}^n \frac{\partial a(\theta^T x)}{\partial \eta} \cdot x^{(i)}\right) = \sum_{i=1}^n \underbrace{\frac{\partial^2 a(\theta^T x^{(i)})}{\partial \eta^2}}_{=:Var} x^{(i)} x^{(i)^T}.$$

We know from previous exercises that outer product is PSD, and sum of PSD is PSD. We also notice, that computing Hessian uncovered definition of variance, which by itself is non-negative term. Thus, Hessian defined in such a way will be PSD for GLM's. This concludes the proof.

Problem 4: Locally weighted linear regression

a) **[10 points]** Consider a linear regression problem in which we want to "weight" different training examples differently. Specifically, suppose we want to minimize

$$J(\theta) = \frac{1}{2} \sum_{i=1}^n w^{(i)} (\theta^\top x^{(i)} - y^{(i)})^2.$$

In class, we worked out what happens for the case where all the weights (the $w^{(i)}$'s) are the same. In this problem, we will generalize some of those ideas to the weighted setting. We will assume $w^{(i)} > 0$ for all i.

(i.) **[2 points]** Show that $J(\theta)$ can also be written as

$$J(\theta) = (X\theta - y)^\top W(X\theta - y)$$

for an appropriate matrix W, and where X and y are as defined in class. Clearly specify the value of each element of the matrix W.

Answer:

Here we will show that this holds true,

$$J(\theta) := \frac{1}{2}\sum_{i=1}^{n} \omega^{(i)}(\theta^T x^{(i)} - y^{(i)})^2 \stackrel{?}{=} \frac{1}{2}(X\theta - y)^T W(X\theta - y)$$

Because $\omega^{(i)}$ is a single-number weight associated with a single example $x^{(i)}$; and, because X matrix in columns contains example vectors $x^{(i)}$, the only way to represent weights in a matrix form would be via diagonal matrix. This means, here we will consider $W := diag(W)$, i.e. W's diagonal elements are $W_{ii} = \omega^{(i)}, i = 1, .., n$, whereas all other elements are 0's.

For simplicity we will use the following notation,

$$a := X\theta - y \quad \text{(here } a \text{ is a vector)}$$

then, having this in mind and expanding the right hand side of our equation, we have

$$\begin{aligned}(X\theta - y)^T W(X\theta - y) = a^T W a = a^T \cdot diag(W) \cdot a = \quad \text{(using sum notation)} \\ = \left(\sum_{i=1}^{n} a_i \omega^{(i)}\right)^T \cdot a = \sum_{i=1}^{n} a_i \omega^{(i)} a_i = \\ = \sum_{i=1}^{n} \omega^{(i)} a_i^2 = \boxed{\sum_{i=1}^{n} \omega^{(i)}(\theta^T x^{(i)} - y^{(i)})^2.}\end{aligned}$$

With this we can conclude that derived expression equates to the left-hand side of $J(\theta)$. This demonstrates that the two expressions are equivalent.

(ii.) **[4 points]** If all the $w^{(i)}$'s equal 1, then we saw in class that the normal equation is $X^\top X\theta = X^\top y$, and that the value of θ that minimizes $J(\theta)$ is given by $(X^\top X)^{-1}X^\top y$. By finding the derivative $\nabla_\theta J(\theta)$ and setting it to zero, generalize the normal equation to this weighted setting and give the new value of θ that minimizes $J(\theta)$ in closed form as a function of X, W, and y.

Answer:

For simplicity we will use results from matrix cookbook (see 2.3.2 Second Order part, page 10 at the top)

$$\frac{\partial}{\partial s}(x - As)^T W(x - As) = -2A^T W(x - As), \quad \text{(provided } W \text{ is symmetric).}$$

Since diagonal matrices are symmetric, adapting this to our $J(\theta)$ will give us,

$$\frac{\partial}{\partial\theta}J(\theta) = \frac{1}{2}\cdot 2X^TW(X\theta - y) = X^TW(X\theta - y) \quad \Rightarrow$$

Setting this derivative to zero, will generalize the normal equation to weighted setting,

$$X^TW(X\theta - y) = 0 \quad \Rightarrow \quad X^TWX\theta = X^TWy \quad \Rightarrow \quad \boxed{\theta = \left[X^TWX\right]^{-1}X^TWy.}$$

(assuming that the matrix (X^TWX) inverse exists).

(iii.) **[4 points]** Suppose we have a dataset $\{(x^{(i)}, y^{(i)}); i = 1, \ldots, n\}$ of n independent examples, but we model the $y^{(i)}$'s as drawn from conditional distributions with different levels of variance $(\sigma^{(i)})^2$. Specifically, assume the model

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi\sigma^{(i)}}}\exp\left(-\frac{(y^{(i)} - \theta^\top x^{(i)})^2}{2(\sigma^{(i)})^2}\right).$$

That is, each $y^{(i)}$ is drawn from a Gaussian distribution with mean $\theta^\top x^{(i)}$ and variance $(\sigma^{(i)})^2$ (where the $\sigma^{(i)}$'s are fixed, known constants). Show that finding the maximum likelihood estimate of θ reduces to solving a weighted linear regression problem. State clearly what the $w^{(i)}$'s are in terms of the $\sigma^{(i)}$'s.

In other words, this suggests that if we have prior knowledge on the noise levels (the variance of the label $y^{(i)}$ conditioned on $x^{(i)}$) of all the examples, then we should use weighted least squares with weights depending on the variances.

Answer:

The joint likelihood can be given as,

$$L(\theta) = \prod_{i=1}^{n} p(y^{(i)}|x^{(i)};\theta) = \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^{(i)2}}}\exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^{(i)2}}\right);$$

For maximum likelihood we will define *log*-likelihood as,

$$\mathscr{L}(\theta) = \sum_{i=1}^{n}\ln p(y^{(i)}|x^{(i)};\theta) = \sum_{i=1}^{n}\left[-\frac{1}{2}\ln(2\pi\sigma^{(i)2}) - \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^{(i)2}}\right];$$

The negative *log*-likelihood is then defined as,

$$-\mathscr{L}(\theta) = \sum_{i=1}^{n}\ln p(y^{(i)}|x^{(i)};\theta) = \sum_{i=1}^{n}\left[\frac{1}{2}\ln(2\pi\sigma^{(i)2}) + \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^{(i)2}}\right];$$

We can convert the negative log-likelihood into a form that resembles the cost function of the weighted linear regression. For this we collect and re-arrange only those terms that include parameters of interest, θ, (this is because computing gradient would result in 0 of other terms), i.e.

$$\sum_{i=1}^{n}\left[\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^{(i)2}}\right] = \frac{1}{2}\sum_{i=1}^{n}\underbrace{\frac{1}{\sigma^{(i)2}}}_{=:\omega^{(i)}}\cdot(y^{(i)} - \theta^T x^{(i)})^2 = \frac{1}{2}\sum_{i=1}^{n}\omega^{(i)}(y^{(i)} - \theta^T x^{(i)})^2$$

Thus, obtained expression is similar to previously defined cost function of LWLR model.

b) **[10 points] Coding problem.**

We will now consider the following dataset (the formatting matches that of Datasets 1-4, except $x^{(i)}$ is 1-dimensional):

`src/lwr/{train,valid,test}.csv`

In `src/lwr/lwr.py`, implement locally weighted linear regression using the normal equations you derived in Part (a) and using

$$w^{(i)} = \exp\left(-\frac{\|x^{(i)} - x\|_2^2}{2\tau^2}\right).$$

Train your model on the train split using $\tau = 0.5$, then run your model on the valid split and report the mean squared error (MSE). Finally, plot your model's predictions on the validation set. Plot the training set with blue 'x' markers and the predictions on the validation set with red 'o' markers.

Does the model seem to be under- or overfitting?

Answer:

Here we trained LWLR model on provided dataset, and obtained mean squared error, MSE = 0.331 (rounded to three decimal places). Below is given a plot (please see Fig. 7) that shows training set in blue crosses, whereas red circles here illustrate predicted response variable, y_{pred} at query/input points, x, from the validation set.

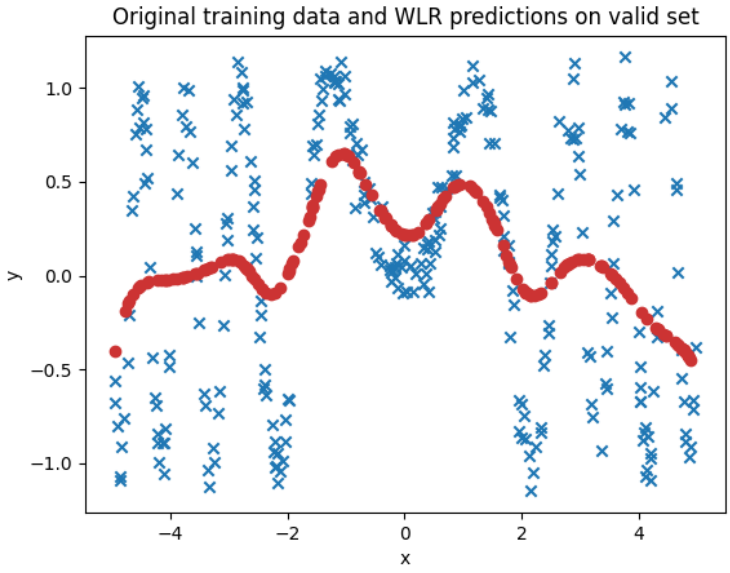

Figure 7: Illustration of LWLR model performance, for fixed $\tau = 0.5$

We can see that obtained model is not able to explain the underlying signal/trend, suggesting that current model is under-fitting; this indicates that our model's complexity/flexibility is insufficient (i.e. model is too simple) to properly capture the desired patterns.

c) **[5 points] Coding problem.**

We will now tune the hyperparameter τ. In `src/lwr/tau.py`, find the MSE value of your model on the validation set for each of the values of τ specified in the code. For each τ, plot your model's predictions on the validation set in the format described in part (b). Report the value of τ which achieves the lowest MSE on the valid split, and finally, report the MSE on the test split using this τ-value.

Answer:

In previous exercise, we worked with a fixed hyper-parameter $\tau = 0.5$. Here, we will perform hyper-parameter tuning to optimize the performance of our LWLR model. We will use the following steps:

- Fitting the LWLR model to training data and then evaluating it on a validation set for different values of τ_i, $i = 1..6$.
- Selecting the value of τ that results in the min(MSE) as the optimal hyperparameter for our LWLR model.
- Finally, testing this model with the optimal τ on the test set.

Following these steps, we identified the optimal value of $\tau_{opt} = 0.05$, with $\mathbf{MSE = 0.012}$. For each τ, Fig.8 illustrates the model's predictions on the validation set. Using the model that corresponds to τ_{opt}, we obtained an $\mathbf{MSE = 0.017}$ on the provided test split.

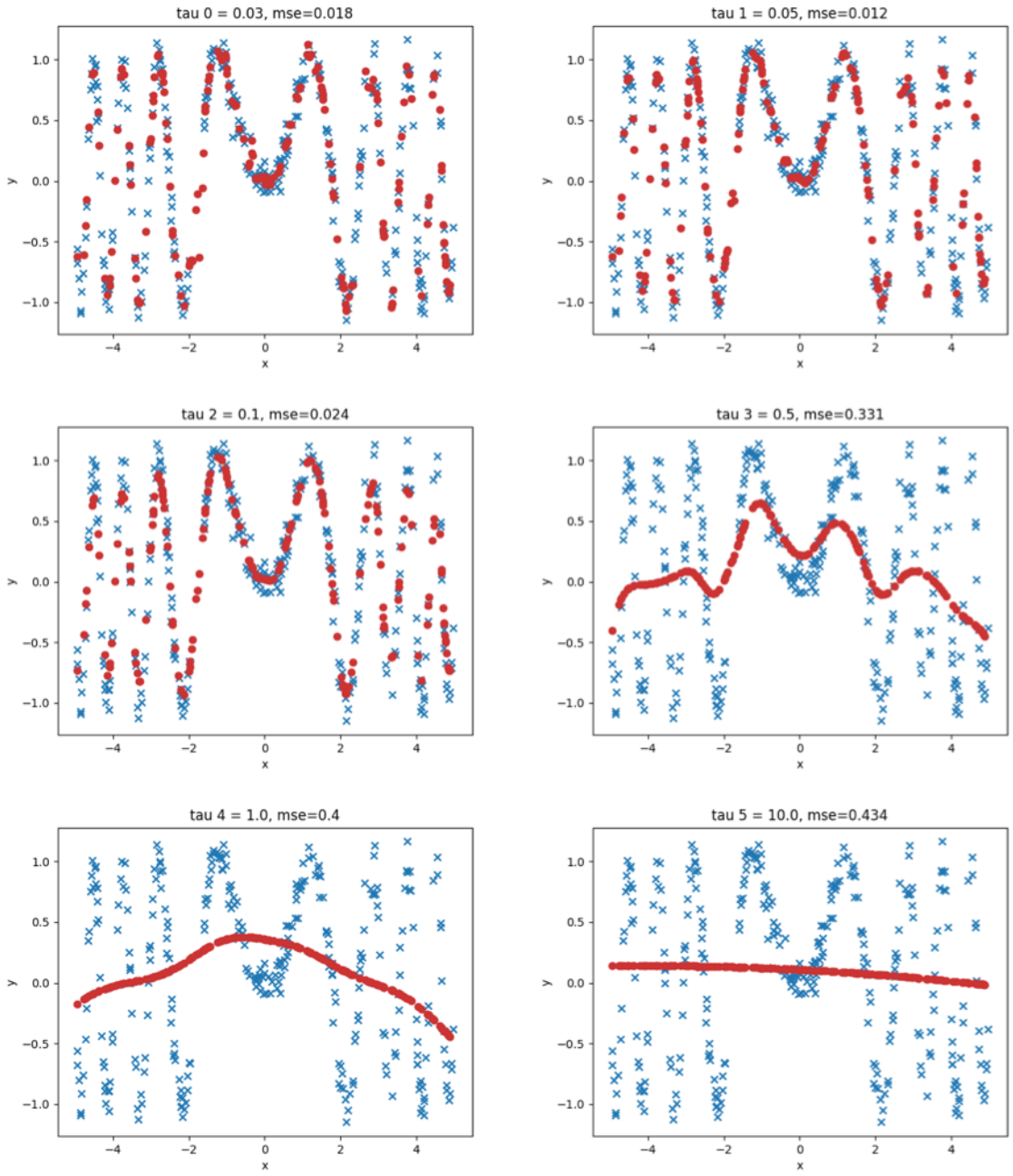

Figure 8: Illustration of LWLR model performance for different values of the τ hyper-parameter.

To conclude, from plots we observe here that MSE can be useful indicator of the quality of a model in the context of over-fitting and under-fitting phenomena.

Problem 5: Linear invariance of optimization algorithms

a) **[7 points]** Show that Newton's method (applied to find the minimum of a function) is invariant to linear reparameterizations. Note that since $z^{(0)} = \vec{0} = A^{-1}x^{(0)}$, it is sufficient to show that if Newton's method applied to $f(x)$ updates $x^{(i)}$ to $x^{(i+1)}$, then Newton's method applied to $g(z)$ will update $z^{(i)} = A^{-1}x^{(i)}$ to $z^{(i+1)} = A^{-1}x^{(i+1)}$.

Answer: ***follow logic given below; rewrite this using multivariate formulas for complete marks*****

Here we firstly state our assumptions:

- For $f(x)$, we have $x^{(0)}, x^{(1)}, \ldots$ iterations using Newton's method.
- For matrix $A \in \mathbb{R}^{d\times d}$, there exists a matrix A^{-1}.
- If the new function is $g(z) = f(Az)$, then $x = Az$ and $z = A^{-1}x$.

Newton's update formula for x can be expressed as:

$$x^{new} := x^{old} - \frac{f'(x)}{f''(x)};$$

We need to show that,

$$z^{new} := z^{old} - \frac{g'(z)}{g''(z)} \stackrel{?}{=} A^{-1}x^{new};$$

Thus, to demonstrate this, we will compute expressions for gradient and Hessian; here we will make use of chain rule:

$$\nabla \equiv \frac{\partial g(z)}{\partial z} = \frac{\partial}{\partial z}\left[f(Az)\right] = \frac{\partial f(Az)}{\partial Az}\cdot\frac{\partial Az}{\partial z} = A\frac{\partial f(Az)}{\partial Az};$$

$$Hess \equiv \frac{\partial^2 g(z)}{\partial z^2} = \frac{\partial}{\partial z}\left[A\frac{\partial f(Az)}{\partial Az}\right] = A^2\frac{\partial^2 f(Az)}{\partial (Az)^2};$$

Therefore, plugging these and above terms from assumption statements into Newton's formula for z^{new}, we get

$$z^{new} := z^{old} - \underbrace{A\frac{\partial f(Az)}{\partial Az}}_{=:g'(z)}\cdot\left[\underbrace{A^2\frac{\partial^2 f(Az)}{\partial (Az)^2}}_{=:g''(z)}\right]^{-1} = A^{-1}\left(x^{old} - \frac{\partial f(x)}{\partial x}\cdot\left[\frac{\partial^2 f(x)}{\partial x^2}\right]^{-1}\right) =$$

$$= A^{-1}\left(x^{old} - \frac{f'(x)}{f''(x)}\right); \quad \Rightarrow \quad \boxed{z^{new} := A^{-1}x^{new}.}$$

Meaning that Newton's method is invariant to linear reparameterizations.

b) **[3 points]** Is gradient descent invariant to linear reparameterizations? Justify your answer.

Answer:

Here we can follow similar logic to see if gradient descent is also invariant to linear reparameterizations. Thus, given gradient descent parameter update rule,

$$x^{new} = x^{old} - \alpha f'(x);$$

we have,

$$z^{new} := z^{old} - \alpha g'(z) = A^{-1}x^{old} - \alpha \cdot A\frac{\partial f(Az)}{\partial Az} = A^{-1}x^{old} - \alpha \cdot Af'(x).$$

Since we cannot reduce this expression further, and $z^{new} :\neq A^{-1}x^{new}$, we conclude that gradient descent is not invariant to linear reparameterizations.