Homework 1 Solutions
Estimation, Naive Bayes, Convexity, Deep Learning

CMU 10-715: Machine Learning (Fall 2015)
http://www.cs.cmu.edu/~bapoczos/Classes/ML10715_2015Fall/
OUT: Sep 21, 2015
DUE: Oct 5, 2015, 10:20 AM

1 Estimating Parameters [Eric; 30 pts]

1.1 Closed Form Estimation

1. (3pts) An exponential distribution with parameter λ follows a distribution $p(x) = \lambda e^{-\lambda x}$. Given some i.i.d. data $\{x_i\}_{i=1}^n \sim \text{Exp}(\lambda)$, derive the maximum likelihood estimate (MLE) $\hat{\lambda}_{MLE}$. Is this estimator biased?

   Solution: The log likelihood is

   $$l(\lambda) = \sum_i \log \lambda - \lambda x_i = n \log \lambda - \lambda \sum_i x_i$$

   Set the derivative to 0:

   $$n/\lambda - \sum_i x_i = 0 \Rightarrow \lambda = \frac{1}{\bar{x}}$$

   This is biased.

2. (5pts) A gamma distribution with parameters α, β has density function $p(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$ where $\Gamma(t)$ is the gamma function (see https://en.wikipedia.org/wiki/Gamma_distribution).

   If the posterior distribution is in the same family as the prior distribution, then we say that the prior distribution is the conjugate prior for the likelihood function. Show that the Gamma distribution (that is $\lambda \sim \text{Gamma}(\alpha, \beta)$) is a conjugate prior of the $\text{Exp}(\lambda)$ distribution. In other words, show that if $x \sim Exp(\lambda)$ and $\lambda \sim \text{Gamma}(\alpha, \beta)$, then $P(\lambda|x) \sim \text{Gamma}(\alpha^*, \beta^*)$ for some values α^*, β^*.

   Solution:

   $$\begin{aligned} P(\lambda|X) &\propto P(X|\lambda)P(\lambda) \\ &\propto \lambda^n e^{-\lambda \sum_i x_i} \lambda^{\alpha-1} e^{-\beta\lambda} \\ &\propto e^{-\lambda(\sum_i x_i - \beta)} \lambda^{n+\alpha-1} \end{aligned}$$

   so $P(\lambda|X) \propto \text{Gamma}(\alpha + n, \sum_i x_i + \beta)$

   Derive the maximum a posteriori estimator (MAP) $\hat{\lambda}_{MAP}$ as a function of α, β. What happens as n gets large?

   Solution: The log posterior is

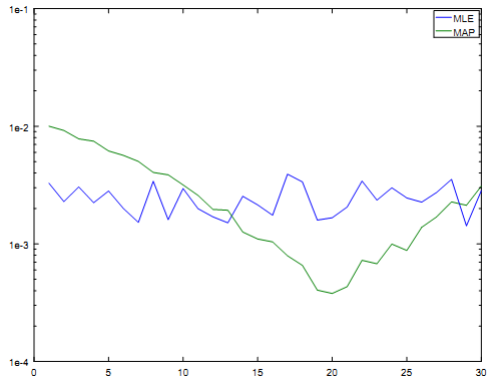
   $$\log P(\lambda|X) \propto -\lambda \left(\sum_i x_i + \beta \right) + (n + \alpha - 1) \log \lambda$$

   Set the derivative to 0:

   $$0 = -\sum_i x_i - \beta + \frac{n + \alpha - 1}{\lambda} \rightarrow \lambda = \frac{n + \alpha - 1}{\sum_i x_i + \beta}$$

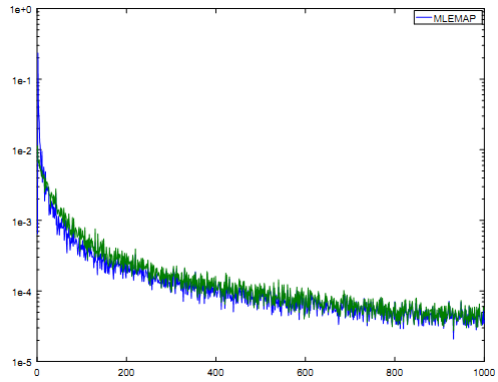
3. (4pt) Let's perform an experiment in the above setting. Generate $n = 20$ random variables drawn from Exp($\lambda = 0.2$). Fix $\beta = 100$ and vary α over the range $(1, 30)$ using a stepsize of 1. Compute the corresponding MLE and MAP estimates for λ. For each α, repeat this process 50 times and compute the mean squared error of both estimates compared against the true value. Plot the mean squared error as a function of α. (Note: Octave parameterizes the Exponential distribution with $\Theta = 1/\lambda$. https://www.gnu.org/software/octave/doc/interpreter/Random-Number-Generation.html may be helpful)

   Solution:



4. (2pt) Now, fix $(\alpha, \beta) = (30, 100)$ and vary n up to 1000. Plot the MSE for each n of the corresponding estimates.

   Solution:



5. (4pts) Under what conditions is the MLE estimator better? Under what conditions is the MAP estimator better? Explain the behavior in the two above plots.

   Solution: The MLE is better when prior information is incorrect. The MAP is better with low sample size and good prior information. Asymptotically they are the same.

1.2 Non-closed Form Estimation

For this question, please make use of the digamma and trigamma functions. You can find them in any scientific computing package (e.g. Octave, Matlab, Python...). This question requires some coding but is not being submitted to Autolab.

1. (3pts) Sometimes we don't have closed forms for the estimators. Given some data $\{x_i\}_{i=1}^n \sim \text{Gamma}(\alpha, \beta)$, use gradient descent to maximize the likelihood and derive the steps to calculate the MLE estimators $\hat{\alpha}_{MLE}, \hat{\beta}_{MLE}$.

Solution: The log likelihood is

$$l(x) = n\alpha \log \beta - n \log \Gamma(\alpha) + (\alpha - 1) \sum_i \log x_i - \beta \sum_i x_i$$

Take the derivative with respect to β results in a simple expression for β in terms of α:

$$\frac{\partial l}{\partial \beta}(x) = \frac{n\alpha}{\beta} - \sum_i x_i = 0$$

so $\beta = \frac{\alpha}{\bar{x}}$. Same with α, plugging in β:

$$l(x) = n\alpha \log \frac{\alpha}{\bar{x}} - n \log \Gamma(\alpha) + (\alpha - 1) \sum_i \log x_i - \frac{\alpha}{\bar{x}} \sum_i x_i$$

$$\frac{\partial l}{\partial \alpha}(x) = n \log \alpha - n \log \bar{x} - n\psi(\alpha) + \sum_i \log x_i$$

So the gradient descent rule would be to subtract this gradient at every step.

2. (3pts) We can also use Newton's method to calculate the same estimate. Provide the Newton's method updates to calculate the above MLE estimators for the Gamma distribution.

   Solution: The second derivative of the likelihood is:

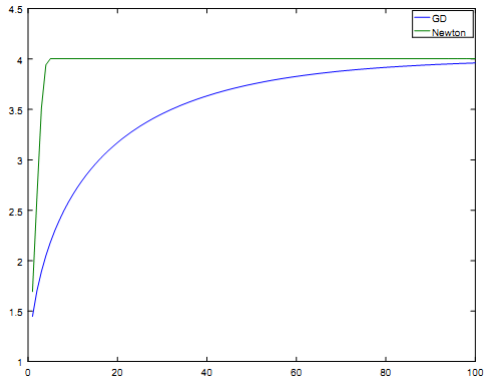
   $$\frac{\partial^2 l}{\partial \alpha^2}(x) = \frac{n}{\alpha} - n\psi'(\alpha)$$

   So the Newton step would be

   $$\alpha = \alpha - \frac{l'(\alpha)}{l''(\alpha)}$$

3. (6pts) Inside the handout, estimators.mat contains a vector drawn from a Gamma distribution. Run your implementation of gradient descent and Newton's method to obtain the MLE estimators for this distribution. Create a plot showing the convergence of the two above methods. How do they compare? Which took more iterations? Lastly, provide the actual estimated values obtained.

   Solution: You should have gotten $\alpha \approx 4, \beta \approx 0.5$.



2 Naive Bayes [Eric; 30 pts]

In this question you will implement a variation on the Naive Bayes classification algorithm. You will be asked to fill in function stubs in the code handout. To submit, please tar the folder named code (e.g. with

tar -cvf code.tar code) and upload it to the Autolab submission site. Your datasets will have class labels $\{1 \dots k\}$ and have real valued features. Assume the features are normally distributed.

Your code will be run on various datasets following the above description. Provided are two datasets: the iris dataset and the forests dataset (as mentioned in recitation). You can find the function stubs in the **code/avg** folder and the datasets in the **data** folder. As a reference, our solution code runs on Autolab in less than 2 minutes.

2.1 Simple Naive Bayes

1. (6pts) First, implement [model] = NaiveBayes(XTrain, yTrain) and [predictions] = NaiveBayesClassify(model, XTest) with an implementation of the naive Bayes classifier for warmup. *model* should contain any precomputed values necessary for classification, which is passed directly to the classifier function. *Predictions* is a $m \times 1$ vector of predicted labels for the datapoints in *XTest*.

2.2 Bayes Model Averaging

Recall the naive Bayes assumption. A naive Bayes classifier may not perform as well on datasets with redundant or excessively large numbers of features. To deal with this, one option is to reduce the number of features and choose a smaller subset based on some criterion (e.g. mutual information, [1]). A different way of dealing with this is to not remove any features, but to take an average over many possible feature subsets. This procedure of averaging over many models instead of explicitly selecting one is more commonly known as Bayes model averaging [2].

In this scenario, let F be the set of all possible feature subsets, where $f = (f_1, \dots, f_k) \in F$ represents a possible subset with $f_i \in \{0, 1\}$. $f_i = 1$ denotes that feature i is used. Assume the following prior on $P(f_i)$:

$$P(f_i) \propto \begin{cases} \frac{1}{\beta} & \text{if } f_i = 1 \\ 1 & \text{if } f_i = 0 \end{cases}$$

As usual, let $D = \{x^{(i)}, y^{(i)}\}$ be the set of training data. Lastly, we define what it means for a model to use feature k as follows:

$$P(x_j^{(i)} | f_j, y_i) = \begin{cases} P(x_j^{(i)} | y_i) & \text{if } f_j = 1 \\ P(x_j^{(i)}) & \text{if } f_j = 0 \end{cases}$$

In other words, if the feature is used, the the probability depends on the output y_i. If a feature is not used, it is reduced to a constant that does not depend on y_i.

For a new observation x^*, the Bayes classifier relies on choosing the label that maximizes the conditional probability of the label given the data, $P(x^*|y, D)P(y|D)$. Now, we want to choose a label that maximizes the same quantity marginalized over all possible models:

$$\underset{y}{\text{argmax}} \sum_f P(x^*, f|y, D)P(y|D)$$

At first glance, this sum looks terrible: there are exponentially many feature subsets! However, in the following questions, you will derive an equivalent classifier that runs in time linear to the number of features.

1. (3pts) Using Bayes rule, rewrite the classifier in terms of $P(x|f, y, D)$, $P(y|D)$, and $P(f|D)$.

   Solution: Let D be the training data, and let x be a new point. In naive Bayes we use the following classifier:

   $$\underset{y}{\text{argmax}}\, P(y|x, D) = P(x|y, D)P(y|D)$$

   To use the Bayes model average framework, instead we marginalize over various feature subsets f:

   $$\underset{y}{\text{argmax}} \sum_f P(x, f|y, D)P(y|D) = \underset{y}{\text{argmax}} \sum_f P(x|f, y, D)P(y|D)P(f|D)$$

2. (3pts) Write an expression for $P(f|D)$ using Bayes rule, and substitute into the classifier. Assume that each feature is selected independently from one another, and relabel the new observation (x, y) as $(x^{(N+1)}, y^{(N+1)})$ to simplify.

   Solution: This is where we would like to know $P(f|D)$. Features are assumed independent so we get $P(f) = \prod_j P(f_j)$:

$$P(f|D) \propto P(D|f)P(f) = \left[\prod_i P(x_i|f, y_i)P(y_i)\right]\left[\prod_j P(f_j)\right]$$

   Substitute this into the above to get

$$\operatorname*{argmax}_y \sum_f P(x|f, y, D)P(y|D)\left[\prod_i P(x_i|f, y_i)P(y_i)\right]\left[\prod_j P(f_j)\right]$$

   We move $P(f_j)$ to the front. Relabeling x, y as the $N+1$th datapoint makes this simpler

$$\operatorname*{argmax}_y \sum_f \left[\prod_j P(f_j)\right]\left[\prod_i^{N+1} P(x_i|f, y_i)P(y_i)\right] = \operatorname*{argmax}_y \sum_f \left[\prod_j P(f_j)\right]\left[\prod_i^{N+1} P(y_i)\prod_{k=1}^{K} P(x_{i,k}|f_k, y_i)\right]$$

3. (6pts) Finally, derive the following form of the classification rule that runs in time linear to the number of features. You will need to exploit the properties of sums and products.

$$\operatorname*{argmax}_y \left[\prod_{i=1}^{N+1} P(y^{(i)})\right]\prod_{k=1}^{K}\left[\prod_{i=1}^{N+1} P(x_k^{(i)}) + \frac{1}{\beta}\prod_{i=1}^{N+1} P(x_k^{(i)}|y)\right]$$

   Solution: Exploit sums and products, and sum explicitly over all features f_j to get

$$\operatorname*{argmax}_y \left[\prod_i^{N+1} P(y_i)\right]\sum_{f_1}\cdots\sum_{f_j}\left[\prod_{k=1}^{K} P(f_j)\prod_i^{N+1} P(x_{i,k}|f_k, y_i)\right]$$

   Exploiting more sums and products, we get:

$$\operatorname*{argmax}_y \left[\prod_i^{N+1} P(y_i)\right]\prod_{j=1}^{J}\sum_{f_j} P(f_j)\prod_i^{N+1} P(x_i|f, y_i)$$

   Finally we plug in the prior for f_i and for $P(x_i|f, y_i)$. As a reminder these are

$$P(f_i) \propto \begin{cases} \frac{1}{\beta} & \text{if } f_i = 1 \\ 1 & \text{if } f_i = 0 \end{cases}, \quad P(x_j^{(i)}|f_j, y_i) = \begin{cases} P(x_j^{(i)}|y_i) & \text{if } f_j = 1 \\ P(x_j^{(i)}) & \text{if } f_j = 0 \end{cases}$$

   Since f_j can only be either 0 or 1, this is now reduced into a form that can be computed in time linear to the number of features.

4. (12pts) Implement [model] = AvgNaiveBayes(XTrain, yTrain) and [predictions] = AvgNaiveBayesClassify(model,XTest). *model* should contain any precomputed values necessary for classification, which is passed directly to the classifier function. *Predictions* is a $m \times 1$ vector of predicted labels for the datapoints in *XTest*.

3 What breaks the convexity of deep learning? [Fish; 40 pts]

In gradient descent algorithm, convexity of the target function plays an important role in determining whether or not the algorithm will converge, the convergence rate and so on. We didn't cover too much about convex function in the class. So here we will go through some useful techniques for examining convexity of a function.

3.1 Basic definition of convex function(28pts)

- Definition of convex set:
  A set $C \subseteq \mathbb{R}^n$ is called convex set, if $\forall x, y \in C$, $tx + (1-t)y \in C$ for $0 \leq t \leq 1$.
  That means for every pairs of points in the convex set C, every point falls on the straight line segment that joins the pair of points will also be in the set. A set that has such properties is super powerful, because you can get to any point in the set from a given point through a straight line without hitting the boundary. Hitting the boundary often means you might get stuck at a local minimum and never have the chance to find the global optimal solution.

- Definition of convex function:
  A function $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function, if $\text{dom}(f)$ is a convex set, and $f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$ for $0 \leq t \leq 1$.
  The inequalities say that the line segment between two points on the function will always lies above the function. Function that has such property is very powerful when you are trying to find the minimum value on the function. Because for any randomly chosen two points, x, y, you can always find a point z that lies between x and y such that $f(z) = \min(f(x), f(y))$. So you will surely be able find the optimal value.

Sometimes it is not easy to examine the convexity though the basic definition of convex. Here are some properties of convex function. You need to prove them in both directions. Half of the points in each questions are given if you prove the argument based on the definition, half of the points are given for proof from argument to definition. (For simplicity, let's assume $dom(f) = \mathbb{R}$)

1. (4pts) If f is continuous, then $f(\frac{x+y}{2}) \leq \frac{f(x)+f(y)}{2}$, $\forall x, y \in \mathbb{R}$
2. (8pts) If f is continuously differentiable, $f(y) \geq f(x) + f'(x)(y-x)$, $\forall x, y \in \mathbb{R}$.
3. (8pts) If f is twice differentiable, $f''(x) \geq 0$, $\forall x \in \mathbb{R}$.

Solution:

3.1.1

From definition to statement 1:
Take $t = \frac{1}{2}$ in the definition and you will get statement 1.
From statement 1 to definition:
It will be easier to prove this by contradiction. Suppose there is a function that satisfies statement 1 but it is not convex. So there must exists a pair of x, y such that

$$f(tx + (1-t)y) > tf(x) + (1-t)f(y), \tag{1}$$

for some $t \in [0, 1]$. Define a function

$$g(t) = f(tx + (1-t)y) - tf(x) - (1-t)f(y), \tag{2}$$

for $t \in [0, 1]$. Note that $g(0) = 0$ and $g(1) = 0$. According to our assumption, there must be a point t such that $g(t) > 0$. Define $m = \sup_{t \in (0,1)} g(t)$ to be the maximum value on g and $t^* = \inf t \in (0, 1), g(t) = m$ to be the smallest t that achieves such value. We first show that g also follows the properties as in statement 3 then prove the contradiction by using this rule.

$$g(\frac{a+b}{2}) = f(\frac{a+b}{2}x + (1 - \frac{a+b}{2})y) - \frac{a+b}{2}f(x) - \left(1 - \frac{a+b}{2}\right) f(y) \tag{3}$$

$$\leq \frac{1}{2}f(ax + (1-a)y) + \frac{1}{2}f(bx + (1-b)y) - \frac{1}{2}af(x) - \frac{1}{2}bf(x) - \frac{1}{2}(1-a)f(x) - \frac{1}{2}(1-b)f(y) \tag{4}$$

$$= \frac{1}{2}g(a) + \frac{1}{2}g(b). \tag{5}$$

So we proved that function g also have the mid-point convex properties. Define ϵ to be a very small number such that $(t^* + epsilon)$ and $(t^* - epsilon)$ is still in $(0, 1)$. Lastly, using this properties and we get

$$g(t^*) = g(\frac{(t^* + \epsilon) + (t^* - \epsilon)}{2}) \tag{6}$$

$$\leq \frac{1}{2}g(t^* + \epsilon) + \frac{1}{2}g(t^* - \epsilon) < \frac{m + m}{2} = m. \tag{7}$$

The last inequality follows the fact that $g(t^* - \epsilon) < m$ and $g(t^* + \epsilon) \leq m$. This contradicts our assumption that $g(t^*) = m$. Thus we prove that function following the rule in statement 1 must be a convex function.

3.1.2

From definition to statement 2:
We said that for every x, y there is always a $t \in [0, 1]$, such that

$$f(tx + (1 - t)y) \leq tf(x) + (1 - t)f(y), \tag{8}$$

which can be rearranged to

$$\frac{f(y + t(x - y)) - f(y)}{t} \leq f(x) - f(y). \tag{9}$$

By dividing and multiplying the right side with $(x - y)$, we have

$$\frac{f(y + t(x - y)) - f(y)}{t(x - y)}(x - y) \leq f(x) - f(y). \tag{10}$$

Let $t \to 0$, we have

$$f'(y)(x - y) \leq f(x) - f(y), \tag{11}$$

which is equivalent to

$$f(y) + f'(y)(y - x) \leq f(x). \tag{12}$$

Thus we proved the statement.
From statement 2 to definition:

By statement 2, we have

$$f(x) \geq f(z) + f'(z)(x - z) \tag{13}$$

$$f(y) \geq f(z) + f'(z)(y - z). \tag{14}$$

By multiplying (13) with t and (14) with $(1 - t)$, we have

$$tf(x) + (1 - t)f(y) \geq f(z) + f'(z)(t(x - z) + (1 - t)(y - z)) = f(z) + f'(z)(tx + (1 - t)y - z) \tag{15}$$

Let $z = tx + (1 - t)y$, then the second term disappears. So we have

$$tf(x) + (1 - t)f(y) \geq f(z) = f(tx + (1 - t)y) \tag{16}$$

3.1.3

From definition to statement 3: By definition, for any x and y,

$$f(y) \geq f(x) + f'(x)(y - x). \tag{17}$$

Using Taylor expansion, we can replace the left hand side with

$$f(x) + f'(x)(y-x) + f''(x)(y-x)^2 + o((y-x)^2). \tag{18}$$

Thus we have

$$f(x) + f'(x)(y-x) + f''(x)(y-x)^2 + o((y-x)^2) \geq f(x) + f'(x)(y-x) \tag{19}$$

$$f''(x)(y-x)^2 + o((y-x)^2)0. \tag{20}$$

Taking $y \to x$, we have

$$f''(x) \leq 0. \tag{21}$$

From statement 3 to definition:
Since function $f(x)$ is twice differentiable, by Taylor expansion, we have

$$f(y) = f(x) + f'(x)(y-x) + f''(x)(y-x)^2 + o((y-x)^2). \tag{22}$$

The statement said that the $f''(x) \geq 0$ at every point. So the last two term is greater or equal to zero. Thus we have

$$f(y) = f(x) + f'(x)(y-x) + f''(x)(y-x)^2 + o((y-x)^2) \geq f(x) + f'(x)(y-x) \tag{23}$$

Also, use the definition to prove that

1. (2pts) If f, g are convex functions, show that $h(x) = max\{f(x), g(x)\}$ is also a convex function.
2. (2pts) If f, g are convex functions, show that $h(x) = f(x) + g(x)$ is also a convex function.
3. (4pts) If function f and g are convex functions, is $f(g(x))$ necessarily a convex function? If not, what kind of conditions do we need to add to make it convex? (Please provide reasons. I know you can always find answer on Wiki.)

Solution:

3.1.4

You can prove it by the definition of convex function. For any x and y,

$$\begin{aligned}
h(tx + (1-t)y) &= \max\{f(tx + (1-t)y), g(tx + (1-t)y)\} && (24)\\
&\leq \max\{tf(x) + (1-t)f(y), tg(x) + (1-t)g(y))\} && (25)\\
&\leq \max\{tf(x), tg(x)\} + \max\{(1-t)f(y), (1-t)g(y)\} && (26)\\
&= t\max\{f(x), g(x)\} + (1-t)\max\{f(y), g(y)\} && (27)\\
&= th(x) + (1-t)h(y). && (28)
\end{aligned}$$

So function h is convex.

3.1.5

You can also prove this one using the basic definition of convex functions. For any x and y,

$$\begin{aligned}
h(tx + (1-t)y) &= f(tx + (1-t)y), g(tx + (1-t)y) && (29)\\
&\leq tf(x) + (1-t)f(y), tg(x) + (1-t)g(y)) && (30)\\
&= t\left(f(x) + g(x)\right) + (1-t)\left(f(y), g(y)\right) && (31)\\
&= th(x) + (1-t)h(y) && (32)
\end{aligned}$$

So function h is convex.

3.1.6

It is not necessarily true. Take $g(x) = x^2$ and $f(x) = -x$, you get $f(g(x)) = -x^2$, which is a concave function. By 3.1.3, if f and g are both twice differentiable, we hope that the second derivative of $f(g(x))$ is always positive. The second to derivative of the function is

$$[f(g(x))]'' = f''(g(x))\left(g'(x)\right)^2 + f'(g(x))g''(x). \tag{33}$$

Since functions f and g are convex and twice differentiable, we know that $f''(x) \geq 0$ and $g''(x) \geq 0$. So the first term in (33) is always greater or equal to zero. To make the second term also greater and equal to zero, we need $f'(x) \geq 0$, which mean f is a non-decreasing function. So if f and g are both convex and twice differentiable and f is non-decreasing function, then $f(g(x))$ is a also a convex function.

3.2 Functions used in deep learning(12pts)

Here are several commonly used objective functions in deep learning. Determine whether they are convex or not and provide formal proof. (Of course you can use the properties we have proved so far) Provide only yes/no answer will get only partial scores. We define some commonly used notations as follows: $w \in \mathbb{R}^d$ is the weight; $x_i \in \mathbb{R}^d$ is the feature vector for the i-th sample. (note that the first element in x_i is set to 1, so we can get rid of the bias term in the lecture note.); $r_i = < w, x_i > \in \mathbb{R}$; $y \in \mathbb{R}$ is the label for the i-th sample.

1. (2pts) Hinge Loss: $H([r_1, r_2, ...r_N]) = \sum_{i=1}^{N} \max(0, 1 - yr_i)$
2. (2pts) Relu: $R(r_i) = \max(0, r_i)$
3. (2pts) Logistic Function: $L(r_i) = -\log(1 + e^{-r_i})$
4. (3pts) Fully connected layer with Soft-max loss: Define $\mathbf{W} \in \mathbb{R}^{d\times k}$ and w_i is the i-th column of $\mathbf{W}$. $\mathbf{W}^T x_i = [< w_1, x_i >, \cdots, < w_k, x_i >] := [r_i(1), r_i(2), \cdots, r_i(k)]$. $S_s(\mathbf{W}) = \log\left(\sum_{j=1}^{k} e^{r_i(j)}\right) - r_i(s)$.
5. (3pts) From the above questions, do you know what breaks the convexity of deep learning? Briefly describe why most of the deep learning functions are non-convex.

Solution:

3.2.1

Yes, it is. Because 0 and $1-yr_i$ are linear linear function, so they are convex. By 3.1.4, we know $\max(0, 1-yr_i)$ is convex. By 3.1.5, sum of several convex functions is still a convex function. So function H is convex.

3.2.2

Yes, it is. 0 and r_i are linear function so they are convex functions. By 3.1.4, maximum of two convex functions is still convex. So the function R is convex.

3.2.3

No, it is not. And actually it is concave. Take the second derivative of the function and you will get $L''(r_i) = -\frac{e^{-r_i}}{(1+e^{-r_i})^2}$, which is always smaller than zero. So it is not convex. (And the original logistic function $\frac{1}{1+e^{-x}}$ is actually neither convex or concave.)

3.2.4

Yes, it is. The second term is a linear combination of W_{ij}, so it is convex. We first show that log sum of exponential is convex. The Hessian for the function

$$L(\theta) = \log(\sum_{j=1}^{k} e^{\theta_i}) \tag{34}$$

is

$$\nabla^2 L(\theta) = \frac{1}{\sum_{i=1}^{k} z_i}\text{diag}(\mathbf{z}) - \frac{1}{(\sum_{i=1}^{k} z_i)^2}\mathbf{z}\mathbf{z}^T, \tag{35}$$

where $z_k = exp(\theta_k)$. To show that this function is positive semi-definite, we need to show that $v^T\nabla^2 L(\theta)v \geq 0$ for all v:

$$v^T\nabla^2 L(\theta)v = \frac{\left(\sum_{i=1}^{k} z_k v_k^2\right)\left(\sum_{i=1}^{k} z_i\right) - \left(\sum_{i=1}^{k} v_i z_i\right)^2}{(\sum_{i=1}^{k} z_i)^2} \geq 0. \tag{36}$$

The last inequality follows Cauchy-Schwarz inequality. Then we show that $L(A \cdot \text{vec}(W) + b)$ is convex if $L(\theta)$ is convex. By taking the second derivative of $L(A \cdot \text{vec}(W) + b)$, we have

$$\nabla^2 L(A \cdot \text{vec}(W) + b) = A^T \nabla^2_{A\cdot\text{vec}(W)+b} L(A \cdot \text{vec}(W) + b)A \succeq 0 \tag{37}$$

The last inequality follows the fact that $\nabla^2 L(x) \succeq 0$. So $L(A \cdot \text{vec}(W) + b)$ is also convex. Thus we conclude that $\log\left(\sum_{j=1}^{k} e^{r_i(j)}\right)$ is convex. Since both terms in $S_s(W)$ are convex in W, by 3.1.5, they are still convex.

3.2.5

There are several reasons that makes most neural net non-convex. According to this question, you can see people use non-linear activation function like logistic function, and it is not convex. Besides, when we have several layers, the chain rule for convexity no longer works since function g in $f(g(x))$ is no longer an affine function in your parameters. But the non-convexity actually gives the model more flexibility and works pretty well on several tasks.

References

[1] Peng, H.; Fulmi Long; Ding, C., "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," in Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.27, no.8, pp.1226-1238, Aug. 2005 doi: 10.1109/TPAMI.2005.159

[2] Hoeting, Jennifer A., et al. "Bayesian model averaging." In Proceedings of the AAAI Workshop on Integrating Multiple Learned Models. 1998.