Homework 3

Young Won Kim (yk41) and Minh Pham (mnp7)

Spring 2017

1 MAP and MLE parameter estimation

Part 1. Using MLE

$D = \{x^{(i)} \mid 1 \leq i \leq m; x \in \{0, 1\}\}$

Probability mass function of each $x^{(i)}$:

$$f(x^{(i)}; \theta) = \theta^{x^{(i)}}(1-\theta)^{(1-x^{(i)}}$$

Likelihood function:

$$L(\theta) = \prod_{i=1}^{m} \theta^{x^{(i)}}(1-\theta)^{(1-x^{(i)})}$$

Maximum likelihood function:

$$L(\theta) = \theta^{\sum_{i=1}^{m} x^{(i)}}(1-\theta)^{(m-\sum_{i=1}^{m} x^{(i)})}$$

$$\theta = argmax_{\theta} L(\theta) = argmax_{\theta} log(L(\theta))$$

$$\theta = argmax_{\theta} \sum_{i=1}^{m} x^{(i)} log(\theta) + (m - \sum_{i=1}^{m} x^{(i)}) log(1-\theta)$$

$$\frac{\partial log(L(\theta))}{\partial \theta} = \frac{\sum_{i=1}^{m} x^{(i)}}{\theta} - \frac{(m-\sum_{i=1}^{m} x^{(i)})}{1-\theta} \equiv 0$$

$$\sum_{i=1}^{m} x^{(i)}(1-\theta) - (m - \sum_{i=1}^{m} x^{(i)})\theta = 0$$

$$\sum_{i=1}^{m} x^{(i)} - m\theta = 0$$

$$\widehat{\theta} = \frac{\sum_{i=1}^{m} x^{(i)}}{m}$$

Part 2. Using MAP

$P(\theta) = \theta^{(a-1)}(1-\theta)^{(b-1)}$

$\widehat{\theta} = argmax_{\theta} P(D \mid \theta)P(\theta) = argmax_{\theta} \prod_{i=1}^{m} \theta^{x^{(i)}}(1-\theta)^{(1-x^{(i)})}\theta^{(a-1)}(1-\theta)^{(b-1)}$

$\widehat{\theta} = argmax_{\theta} \sum_{i=1}^{m} x^{(i)} log(\theta) + (m - \sum_{i=1}^{m} x^{(i)}) log(1-\theta) + log(\theta)(a-1) + (b-1)log(1-\theta)$

$\frac{\partial L(\theta)}{\partial \theta} = \frac{\sum_{i=1}^{m} x^{(i)}}{\theta} - \frac{(m - \sum_{i=1}^{m} x^{(i)})}{1-\theta} + \frac{a-1}{\theta} - \frac{b-1}{1-\theta} \equiv 0$

$0 = \sum_{i=1}^{m} x^{(i)}(1-\theta) - \theta(m - \sum_{i=1}^{m} x^{(i)}) + (a-1)(1-\theta) - \theta(b-1)$

$0 = \sum_{i=1}^{m} x^{(i)} + \theta(-m-a-b+2) - 1 + a$

With a = b = 1, we have:

$\theta = \frac{\sum_{i=1}^{m} x^{(i)}}{m}$

2 Logistic regression and Gaussian Naive Bayes

Part 1 Logistic regression

$P(y = 1 \mid x; \theta) = g(\theta^T x); \theta \in \mathbb{R}^{d+1}$

$P(y = 0 \mid x; \theta) = 1 - g(\theta^T x)$

where $g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$

Part 2 Gaussian Naive Bayes

$P(y = 1 \mid x) = \frac{P(x|y=1)P(y=1)}{P(x)} = \frac{P(x|y=1)P(y=1)}{P(x|y=1)P(y=1)+P(x|y=0)P(y=0)}$

$P(y = 1 \mid x) = \frac{1}{1+\frac{P(y=0)P(x|y=0)}{P(y=1)P(x|y=1)}}$

$P(y = 1 \mid x) = \frac{1}{1+exp(ln\frac{P(y=0)P(x|y=0)}{P(y=1)P(x|y=1)})}$

$P(y = 1 \mid x) = \frac{1}{1+exp(ln\frac{P(y=0}{P(y=1)}+ln\frac{P(x|y=0)}{P(x|y=1)})}$

We have:

$\frac{P(y=0)}{P(y=1)} = \frac{1-\phi}{\phi}$

And:

$ln\frac{P(x|y=0)}{P(x|y=1)} = \sum_{j=1}^{d}(ln\frac{P(x_j|y=0)}{P(x_j|y=1)}$

$ln\frac{P(x|y=0)}{P(x|y=1)} = \sum_{j=1}^{d} ln\frac{\frac{1}{\sqrt{2\pi\sigma^2}}exp(\frac{-(x_j+\mu_{j0})^2}{2\sigma_j^2})}{\frac{1}{\sqrt{2\pi\sigma^2}}exp(\frac{-(x_j+\mu_{j1})^2}{2\sigma_j^2})}$

$ln\frac{P(x|y=0)}{P(x|y=1)} = \sum_{j=1}^{d} lnexp(\frac{(x_j-\mu_{j1})^2-(x_j-\mu_{j0})^2}{2\sigma_j^2}) = \sum_{j=1}^{d}(\frac{\mu_{j0}-\mu_{j1}}{\sigma_j^2}x_j + \frac{\mu_{j1}^2-\mu_{j0}^2}{\sigma_j^2})$

Therefore,

$P(y = 1 \mid x) = \frac{1}{1+exp(w_0+\sum_{j=1}^{d} w_j x_j)}$,

in which,

$w_0 = ln\frac{1-\phi}{\phi} + \sum_{j=1}^{d} \frac{\mu_{j1}^2-\mu_{j0}^2}{\sigma_j^2}$

$w_j = \frac{\mu_{j0}-\mu_{j1}}{\sigma_j^2}$

$P(y = 0 \mid x) = 1 - P(y = 1 \mid x) = \frac{exp(w_0+\sum_{j=1}^{d} w_j x_j)}{1+exp(w_0+\sum_{j=1}^{d} w_j x_j)}$

Part 3 Uniform Class Priors

Since class 1 and class 0 are equally likely, we have:

$\frac{P(y=0)}{P(y=1)} = \frac{1-\phi}{\phi} = 1$

Therefore,

$P(y = 1 \mid x) = \frac{1}{1+exp\sum_{j=1}^{d}(\frac{\mu_{j1}^2-\mu_{j0}^2}{2\sigma_j^2}+\frac{\mu_{j0}-\mu_{j1}}{\sigma_j^2}x_j)}$

We call:

$\theta_0 = \frac{\mu_{j1}^2-\mu_{j0}^2}{2\sigma_j^2}$

$\theta_j = \frac{\mu_{j0}-\mu_{j1}}{\sigma_j^2}$

Therefore,

$P(y = 1 \mid x) = \frac{1}{1+exp(\theta_0+\sum_{j=1}^{d}(\theta_j x_j))} = \frac{1}{1+exp(\theta^T x)} \equiv P(y = 1 \mid x)$ for logistic regression

3 Reject Option In Classifiers

Part 1 Minimum Risk

Risk for choosing an action i is:

$$R(\alpha_i \mid x) = \sum_{j=1}^{C} L(\alpha = i, y = j)P(y = 1 \mid x) \text{ for } i = 1, ..., C$$

$$R(\alpha_i \mid x) = \sum_{j=1, j\neq i}^{C} \lambda_s P(y = j \mid x) = \lambda_s(1 - P(y = i \mid x))$$

Risk for a rejection is:

$$R(\alpha_{C+1} \mid x) = \lambda_r$$

Minimal risk that is obtained when we decide $y = j$ is:

$$\lambda_r \leq \lambda_s(1 - P(y = i \mid x)) \text{ with } i = 1, ..., C$$

Therefore,

$$P(y = i \mid x) \leq 1 - \frac{\lambda_r}{\lambda_s} \text{ with } i = 1, ..., C$$

In addition, in an intuitive sense, we also have to choose the most probable class, which means $P(y = j \mid x) \leq P(y = k \mid x)$ for all k

Part 2 The loss ratio between reject and misclassification increases from 0 to 1

As $\frac{\lambda_1}{\lambda_2}$ increases from 0 to 1, the loss of rejection increases. When $\frac{\lambda_1}{\lambda_2} = 1$, the loss of rejection is larger than the loss of misclassification, and we shouldn't choose the rejection option.

4 One_vs_all logistic regression and softmax regression

- As θ is initially generated randomly, there is no preference for the class y belongs to, and therefore probability is equal for each classes, $\frac{1}{10}$. If we sum up accross m examples and divide by m, the loss results to $-log(0.1)$.

- Naive loss: 2.365103 computed in 26.325576s
- Vectorized loss: 2.365103 computed in 0.607100s
- Loss difference: 0.000000
- Gradient difference: 0.000000

```
one_vs_all on raw pixels final test set accuracy: 0.362000
[[465  59  21  24  19  35  26  60 201  90]
[ 67 465  18  35  23  31  44  50  94 173]
[123  65 193  77  96  89 151  89  69  48]
[ 66  86  78 161  49 193 171  51  62  83]
[ 65  38 103  64 234  90 194 128  36  48]
[ 48  63  81 126  81 272 114  88  72  55]
[ 31  53  67 102  85  78 457  52  29  46]
[ 53  62  51  46  69  84  66 405  49 115]
[143  78   8  25   9  34  22  20 547 114]
[ 59 208  14  22  23  29  60  56 108 421]]
```

Figure 1: Confusion matrix of OVA

```
softmax on raw pixels final test set accuracy: 0.408100
[[484  51  50  23  14  24  25  45 203  81]
 [ 61 514  17  30  15  38  36  48  83 158]
 [ 96  59 258  68 119  86 164  65  58  27]
 [ 46  84  89 247  47 186 124  53  54  70]
 [ 63  38 110  67 293  76 179 110  31  33]
 [ 47  53  94 137  72 338  93  67  69  30]
 [ 18  53  51  98 102  80 496  38  25  39]
 [ 56  49  53  48  95  84  53 431  48  83]
 [161  74   8  16   5  51  10  14 550 111]
 [ 73 205  10  20  15  21  45  51  90 470]]
```

Figure 2: Confusion matrix of SOFTMAX

- The best value for learning rate and regularization strength are: lr=1.000000e-06, reg=1.000000e+05. This resulted in validation accuracy of 41.6%.

- The best softmax classier on the test set resulted in overall accuracy of 40.8%.

- Comparing OVA vs SOFTMAX on CIFAR-10: OVA yielded lower recall, precision, and accuracy compared to Softmax. Sofmax works better on datasets where the features aren't obviously related to specific classes, which makes it a better classifier for CIFAR-10. Empirical data also supports this, as Softmax performed better than OVA in terms of recall, precision, and accuracy.

- The visualized coefficients corresponds to the importance of each pixel in determining the class. Bluer color represent higher correlation between the class and the pixel.

	OVA		Softmax	
Label	Recall	Precision	Recall	Precision
0	0.465	0.4151786	0.484	0.438009
1	0.465	0.3950722	0.514	0.435593
2	0.193	0.3044164	0.258	0.348649
3	0.161	0.2360704	0.247	0.327586
4	0.234	0.3401163	0.293	0.377091
5	0.272	0.2909091	0.338	0.343496
6	0.457	0.3501916	0.496	0.404898
7	0.405	0.4054054	0.431	0.467462
8	0.547	0.4317285	0.55	0.45417
9	0.421	0.3528919	0.47	0.426497
Accuracy	0.362		0.4081	

Figure 3: Comparison between OVA and SOFTMAX - precision, recall, accuracy

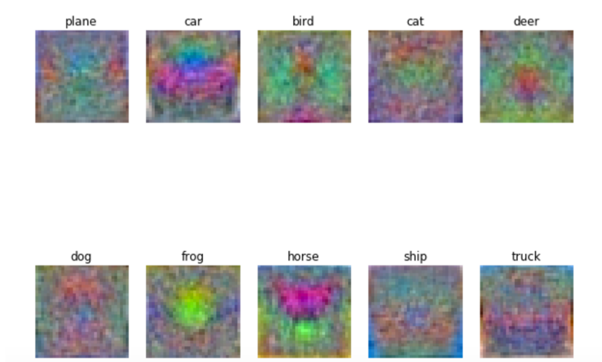

Figure 4: Visualization of the learned parameter matrix

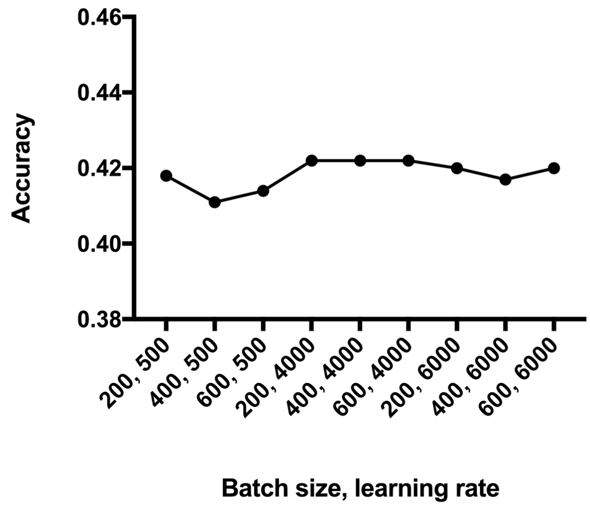

Figure 5: Other hyperparameters and their accuracies

Extra Credit
Based on runs with different hyperparameters, we found that higher batch size and number of iterations tend to result in higher accuracy. However, increasing the batch size and number of iterations too much also leads to overfitting. The number of iterations that yielded the best accuracy was 4000, regardless of the batch size, highest accuracy being 0.422 (Fig.5).

Selected runs and their hyperparameters:
200 batch size, 500 iterations, lr 5.0e-07, reg 1.0e+05: train accuracy of 0.395, val accuracy of 0.418
400 batch size, 500 iterations, lr 1.0e-06, reg 1.0e+05: train accuracy of 0.403, val accuracy of 0.411
600 batch size, 500 iterations, lr 1.0e-06, reg 5.0e+05: train accuracy of 0.405, val accuracy of 0.414
200 batch size, 4000 iterations, lr 5.0e-07, reg 1.0e+05: train accuracy of 0.403, val accuracy of 0.422
400 batch size, 4000 iterations, lr 1.0e-06, reg 5.0e+04: train accuracy of 0.412, val accuracy of 0.422
600 batch size, 4000 iterations, lr 1.0e-06, reg 1.0e+05: train accuracy of 0.409, val accuracy of 0.422
200 batch size, 6000 iterations, lr 5.0e-07, reg 1.0e+05: train accuracy of 0.406, val accuracy of 0.420
400 batch size, 6000 iterations, lr 5.0e-07, reg 5.0e+04: train accuracy of 0.413, val accuracy of 0.417
600 batch size, 6000 iterations, lr 5.0e-07, reg 5.0e+04: train accuracy of 0.412, val accuracy of 0.420