COMP 540 Assignment #3

Yunda Jia yj32
Yu Wu yw92

February 21, 2020

1 MAP and MLE parameter estimation(10 points)

- Use the method of maximum likelihood estimation to derive an estimate for θ from the coin flip results in D.

$$\begin{aligned} P(x=1) &= \theta \\ P(x=0) &= 1-\theta \\ f(x_i;\theta) &= \theta^{x_i}(1-\theta)^{1-x_i} \end{aligned}$$

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{m} f(x_i;\theta) \\ &= \theta^{x_1}(1-\theta)^{1-x_1} \times \theta^{x_2}(1-\theta)^{1-x_2} \times ... \times \theta^{x_m}(1-\theta)^{1-x_m} \\ &= \theta^{\sum_i^m x_i}(1-\theta)^{m-\sum_i^m x_i} \end{aligned}$$

$$\begin{aligned} \log(L(\theta)) &= \sum_i^m x_i \log(\theta) + (m - \sum_i^m x_i)(1-\theta) \\ \frac{\partial \log(L(\theta))}{\partial \theta} &= \frac{\sum_i^m x_i}{\theta} - \frac{m - \sum_i^m x_i}{1-\theta} \end{aligned}$$

Let $\frac{\partial \log(L(\theta))}{\partial \theta} = 0$.

$$\begin{aligned} (1-\theta)\sum_i^m x_i - \theta(m - \sum_i^m x_i) = 0 \\ \sum_i^m x_i - \theta \sum_i^m x_i - \theta m + \theta \sum_i^m x_i = 0 \\ \widehat{\theta} = \frac{\sum_i^m x_i}{m} \end{aligned}$$

So $\widehat{\theta} = \frac{\sum_i^m x_i}{m}$.

- Derive the MAP estimate for θ using D and this prior distribution. Show that under a uniform prior (Beta distribution with a = b = 1), the MAP and MLE estimates of θ are equal.

$$Beta(\theta|a,b) \propto \theta^{a-1}(1-\theta)^{b-1}$$

$$
\begin{aligned}
f(\theta) &= L(\theta) Beta(\theta|a,b) \\
&= \theta^{\sum_i^m x_i}(1-\theta)^{m-\sum_i^m x_i}\theta^{a-1}(1-\theta)^{b-1} \\
&= \theta^{\sum_i^m x_i + a - 1}(1-\theta)^{m-\sum_i^m x_i+b-1}
\end{aligned}
$$

$$
\begin{aligned}
\log f(\theta) &= (\sum_i^m x_i + a - 1)\log\theta + (m - \sum_i^m x_i + b - 1)\log(1-\theta) \\
\frac{\partial \log f(\theta)}{\partial \theta} &= \frac{\sum_i^m x_i + a - 1}{\theta} - \frac{m - \sum_i^m x_i + b - 1}{1-\theta}
\end{aligned}
$$

Let $\frac{\partial \log f(\theta)}{\partial \theta} = 0$.

$$
\begin{aligned}
(1-\theta)(\sum_i^m x_i + a - 1) - \theta(m - \sum_i^m x_i + b - 1) &= 0 \\
\sum_i^m x_i + a - 1 - (\sum_i^m x_i + a - 1)\theta - m\theta - (-\sum_i^m x_i + b - 1)\theta &= 0 \\
\sum_i^m x_i + a - 1 + (2 - a - b - m)\theta &= 0 \\
\theta &= \frac{\sum_i^m x_i + a - 1}{m + a + b - 2}
\end{aligned}
$$

So $\theta = \frac{\sum_i^m x_i + a - 1}{m+a+b-2}$.

2 Logistic regression and Gaussian Naive Bayes (15 points)

- For logistic regression, what is the posterior probability for each class?

$$
\begin{aligned}
P(y=1|x) &= g(\theta^T x) \\
&= \frac{1}{1+e^{-\theta^T x}} \\
P(y=0|x) &= 1 - g(\theta^T x) \\
&= \frac{1}{1+e^{\theta^T x}}
\end{aligned}
$$

https://www.overleaf.com/project/5e4094873cca90000118857c

- Derive the posterior probabilities for each class.

$$
\begin{aligned}
y &\sim Bernoulli(\gamma) \\
x_j|y=1 &\sim N(\mu_j^1, \sigma_j^2) \\
x_j|y=0 &\sim N(\mu_j^0, \sigma_j^2)
\end{aligned}
$$

$$
\begin{aligned}
P(y=1|x) &= \frac{P(x|y=1)P(y=1)}{P(x)} \\
&= \frac{P(x|y=1)P(y=1)}{P(x|y=0)P(y=0) + P(x|y=1)P(y=1)} \\
&= \frac{\gamma \prod_{j=1}^d \frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{-(x_j-\mu_j^1)^2}{\sigma_j^2}}}{(1-\gamma)\prod_{j=1}^d \frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{-(x_j-\mu_j^0)^2}{\sigma_j^2}} + \gamma \prod_{j=1}^d \frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{-(x_j-\mu_j^1)^2}{\sigma_j^2}}}
\end{aligned}
$$

Similar for $P(y=0|x)$.

$$
\begin{aligned}
P(y=0|x) &= \frac{P(x|y=0)P(y=0)}{P(x)} \\
&= \frac{P(x|y=0)P(y=0)}{P(x|y=0)P(y=0)+P(x|y=1)P(y=1)} \\
&= \frac{\gamma \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{-(x_j-\mu_j^0)^2}{\sigma_j^2}}}{(1-\gamma)\prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{-(x_j-\mu_j^0)^2}{\sigma_j^2}} + \gamma \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{-(x_j-\mu_j^1)^2}{\sigma_j^2}}}
\end{aligned}
$$

- Show that with appropriate parameterization, $P(y=1|x)$ for Gaussian Naive Bayes with uniform priors is equivalent to $P(y=1|x)$ for logistic regression.
  Since class 1 and class 0 are equally likely. $\gamma = 0.5$.

$$
\begin{aligned}
P(y=1|x) &= \frac{\gamma \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{-(x_j-\mu_j^1)^2}{\sigma_j^2}}}{(1-\gamma)\prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{-(x_j-\mu_j^0)^2}{\sigma_j^2}} + \gamma \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{-(x_j-\mu_j^1)^2}{\sigma_j^2}}} \\
&= \frac{\sqrt{2\pi}^{-d}\sigma_j^{-d} e^{-\sum_{j=1}^{d}\frac{(x_j-\mu_j^1)^2}{\sigma_j^2}}}{\sqrt{2\pi}^{-d}\sigma_j^{-d} e^{-\sum_{j=1}^{d}\frac{(x_j-\mu_j^0)^2}{\sigma_j^2}} + \sqrt{2\pi}^{-d}\sigma_j^{-d} e^{-\sum_{j=1}^{d}\frac{(x_j-\mu_j^1)^2}{\sigma_j^2}}} \\
&= \frac{1}{1+e^{\sum_{j=1}^{d}\frac{(x_j-\mu_j^1)^2}{\sigma_j^2} - \sum_{j=1}^{d}\frac{(x_j-\mu_j^0)^2}{\sigma_j^2}}}
\end{aligned}
$$

$$
\begin{aligned}
e^{\sum_{j=1}^{d}\frac{(x_j-\mu_j^1)^2}{\sigma_j^2} - \sum_{j=1}^{d}\frac{(x_j-\mu_j^0)^2}{\sigma_j^2}} &= e^{\sum_{j=1}^{d}\frac{(\mu_j^{12}-\mu_j^{02})+2(\mu_j^0-\mu_j^1)x_j}{\sigma_j^2}} \\
&= e^{\sum_{j=1}^{d}\frac{(\mu_j^{12}-\mu_j^{02})}{\sigma_j^2} + \sum_{j=1}^{d}\frac{2(\mu_j^0-\mu_j^0)}{\sigma_j^2}x_j}
\end{aligned}
$$

For Logistic regression.

$$
\begin{aligned}
P(y=1|x) &= \frac{1}{1-e^{-\theta^T x}} \\
&= \frac{1}{1+e^{-\theta_0-\sum_{j=1}^{d}\theta_{jj}}}
\end{aligned}
$$

Let

$$
\begin{aligned}
\theta_0 &= \sum_{j=1}^{d}\frac{(\mu_j^{02}-\mu_j^{12})}{\sigma_j^2} \\
\theta_j &= \sum_{j=1}^{d}\frac{2(\mu_j^0-\mu_j^0)}{\sigma_j^2}
\end{aligned}
$$

Then, with above parameterization, $P(y=1|x)$ for Gaussian Naive Bayes with uniform priors is equivalent to $P(y=1|x)$ for logistic regression.

3 Reject option in classifiers (10 points)

- Show that the minimum risk is obtained if we decide $y = j$ if $p(y = j|X) \geq p(y = k|x)$ for all k (i.e., j is the most probable class) and if $p(y = j|x) \geq 1 - \frac{\lambda_r}{\lambda_s}$, otherwise we decide to reject.

$$L_r = \sum_{k=1}^{C} L(\alpha_{C+1}|y = k)P(y = k|x)$$
$$= \lambda_r \sum_{k=1}^{C} P(y = k|x)$$
$$= \lambda_r$$

If we decide $y = j$.

$$L_j = \sum_{k=1}^{C} L(\alpha_i|y = k)P(y = k|x)$$
$$= 0 * P(y = j|x) \sum_{k=1}^{C} \lambda_s P(y = k|x)$$
$$= \lambda_s[1 - P(y = j|x)]$$

If $P(y = j|x) \geq 1 - \frac{\lambda_r}{\lambda_s}$

$$L_j \leq \lambda_s[1 - (1 - \frac{\lambda_r}{\lambda_s})]$$
$$= \lambda_r$$
$$= L_r$$

- Describe qualitatively what happens as $\frac{\lambda_r}{\lambda_s}$ is increased from 0 to 1.
If $\frac{\lambda_r}{\lambda_s} = 0$, then $1 - \frac{\lambda_r}{\lambda_s} = 1$, so we always reject. If $\frac{\lambda_r}{\lambda_s} = 1$, then $1 - \frac{\lambda_r}{\lambda_s} = 0$, we never reject.
Thus, when $\frac{\lambda_r}{\lambda_s}$ is increased from 0 to 1, we less likely to reject.

4 Kernelizing k-nearest neighbors (5 points)

We compute distance.

$$d = \sqrt{\sum_{j=1}^{d}(x_j - x_j^{(i)})^2}$$
$$d^2 = \sum_{j=1}^{d}(x_j - x_j^{(i)})^2$$
$$= x^T x - 2x^T x^{(i)} + (x^{(i)})^T x^{(i)}$$
$$= k(x, x) - 2k(x, x^{(i)}) + k(x^{(i)}, x^{(i)})$$

In order to meet the Mercer condition.
Let $k(x, x^{'})^{=exp(-\frac{||x-x^{'}||^2}{2\sigma^2})}$
Then

$$d^2 = 2 - exp(-\frac{||x - x^{'}||^2}{2\sigma^2})$$

5 Constructing kernels (10 points)

- $k(x, x^{'}) = ck_1(x, x^{'})$
  The Gram matrix of k_1 is K_1, which is positive semidefinite.
  So $\forall u$, $u^T K_1 u > 0$.
  Gram matrix of k is K, $K = cK_1$.
  Thus, $\forall u$

$$u^T K u = cu^T K_1 u > 0$$

  So $k(x, x^{'}) = ck_1(x, x^{'})$ is a valid kernel.

- $k(x, x^{'}) = f(x)k_1(x, x^{'})f(x^{'})$

$$\begin{aligned} k(x, x^{'}) &= f(x)k_1(x, x^{'})f(x^{'}) \\ k_1(x, x^{'}) &= \phi_1(x)^T \phi_1(x^{'}) \\ k(x, x^{'}) &= f(x)\phi_1(x)^T \phi_1(x^{'})f(x^{'}) \end{aligned}$$

  Let $\phi(x)^T = f(x)\phi_1(x)^T$, $\phi(x^{'}) = \phi_1(x^{'})f(x^{'})$.

$$k(x, x^{'}) = \phi(x)^T \phi(x^{'})$$

  So $k(x, x^{'}) = f(x)k_1(x, x^{'})f(x^{'})$ is a valid kernel.

- $k(x, x^{'}) = k_1(x, x^{'}) + k_2(x, x^{'})$
  For $\forall u$,

$$\begin{aligned} u^T K u &= u^T (K_1 + K_2) u \\ &= u^T K_1 u + u^T K_2 u \\ &> 0 \end{aligned}$$

  So $k(x, x^{'}) = k_1(x, x^{'}) + k_2(x, x^{'})$ is a valid kernel.

- $k(x, x^{'}) = x^T rev(x^{'})$
  Gram matrix of $k(x, x^{'})$, let $c^T = (1, 0, 0, 0..., 0)^T$

$$\begin{aligned} c^T K c &= x_1^T rev(x_1^{'}) \\ &= x_1^T (-x_1) \\ &= -1 \end{aligned}$$

  So K is not positive definite, $k(x, x^{'}) = x^T rev(x^{'})$ is not a valid kernel.

6 One_vs_all logistic regression (15 points)

6.1 Implementing a one-vs-all classifier for the CIFAR-10 dataset (10 points)

See one_vs_all.py for my implementation.

6.2 Predicting with a one-vs-all classifier (5 points)

My result is shown below.

```
one_vs_all on raw pixels final test set accuracy: 0.371300
[[483  52  23  24  20  34  28  55 196  85]
 [ 68 464  16  26  22  31  45  54  92 182]
 [130  65 204  67  94  90 150  85  68  47]
 [ 71  80  79 155  46 199 168  49  66  87]
 [ 71  43 113  49 240  91 191 123  31  48]
 [ 56  65  79 110  79 300 109  84  67  51]
 [ 34  55  70  89  88  85 456  45  26  52]
 [ 58  64  51  38  73  84  51 421  44 116]
 [141  78   8  16  11  37  19  19 549 122]
 [ 68 198  12  15  24  31  55  55 101 441]]
```

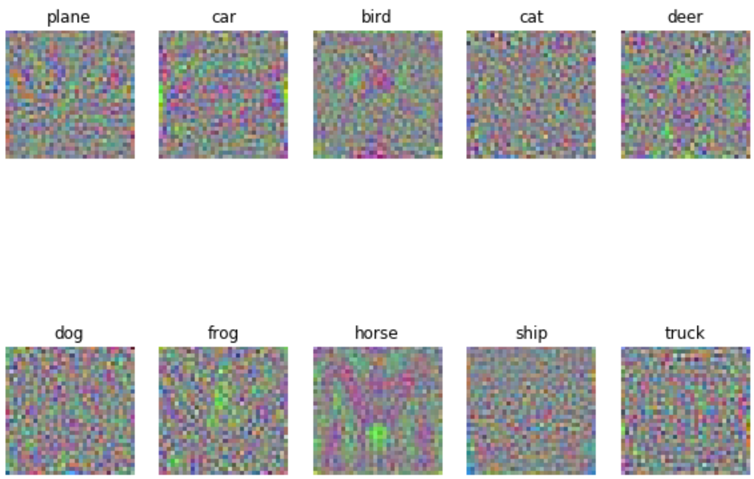

Figure 1: Visualization

6.3 Comparing your OVA classifier with sklearn's classifier

The result from kelearn library is shown below.

```
one_vs_all on raw pixels final test set accuracy (sklearn): 0.371300
[[483  52  23  24  20  34  28  55 196  85]
 [ 68 464  16  26  22  31  45  54  92 182]
 [130  65 204  67  94  90 150  85  68  47]
 [ 71  80  79 155  46 199 168  49  66  87]
 [ 71  43 113  49 240  91 191 123  31  48]
 [ 56  65  79 110  79 300 109  84  67  51]
 [ 34  55  70  89  88  85 456  45  26  52]
 [ 58  64  51  38  73  84  51 421  44 116]
 [141  78   8  16  11  37  19  19 549 122]
 [ 68 198  12  15  24  31  55  55 101 441]]
```

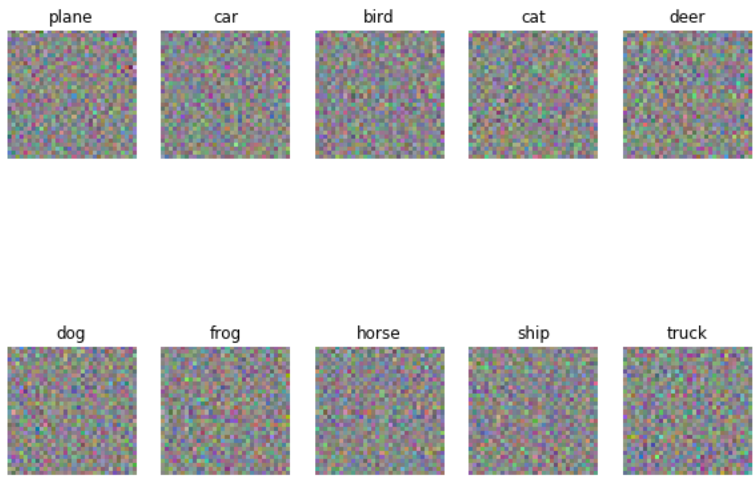

Figure 2: Library visualization

7 Softmax regression (45 points)

7.1 Implementing the loss function for softmax regression (naive version) (5 points)

Loss is 2.32, which is close to $-ln(0.1)$, because:

$$
\begin{aligned}
J(\theta) &= \frac{-1}{m}\sum_{i=1}^{m}\log\frac{exp(\theta^{(k)T})x^{(i)}}{\sum_{j=1}^{K}(\theta(k)Tx^{(j)}} \\
&\approx \frac{-1}{m}\sum_{i=1}^{m}\log(0.1) \\
&= -\log(0.1)
\end{aligned}
$$

7.2 Implementing the gradient of loss function for softmax regression (naive version) (5 points)

My gradient implementation is the same as numerical gradients.

7.3 Implementing the loss function for softmax regression (vectorized version) (10 points)

7.4 Implementing the gradient of loss for softmax regression (vectorized version) (5 points)

7.5 Implementing mini-batch gradient descent (5 points)

7.6 Using a validation set to select regularization lambda and learning rate for gradient descent (5 points)

Best validation accuracy achieved during cross-validation: 0.415000. When learning rate is $5e-7$ and regularization term is $5e+5$.

7.7 Training a softmax classifier with the best hyperparameters (5 points)

The softmax on raw pixels final test set accuracy: 0.400400.
The visualization is shown below.

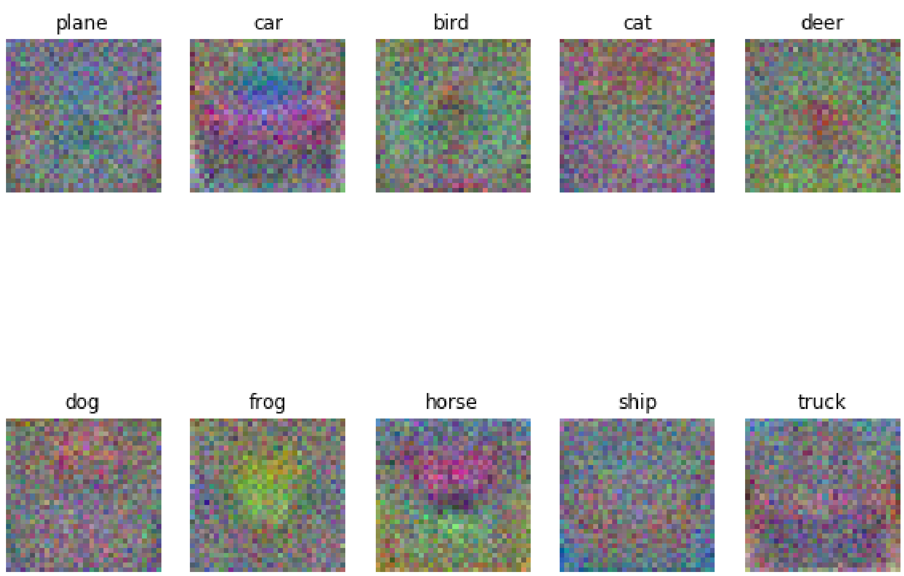

Figure 3: Visualization of coefficients

7.8 Experimenting with other hyper parameters and optimization method (10 points)

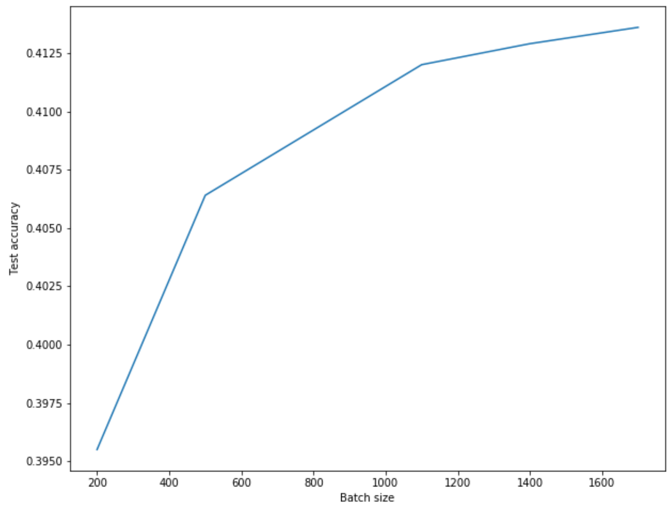

Figure 4: Test accuracy with batch sizes

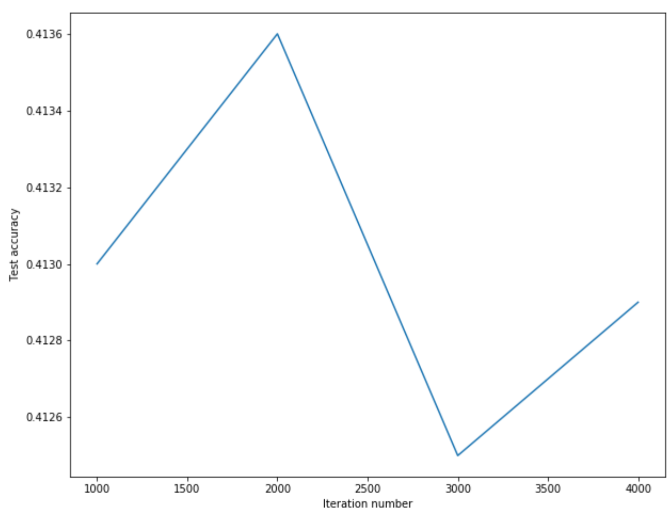

Figure 5: Test accuracy with iteration number

The best test accuracy is 0.331600 when learning rate 5e-7, regularization 5e5, iteration number 1700, batch size 2000.

7.9 Comparing OVA binary logistic regression with softmax regression (5 points)

	Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
OVA	483	464	204	155	240	300	456	421	549	441
Softmax	514	484	204	246	279	349	519	405	521	483

Overall, the softmax classifier performs better than the OVA classifier. Although for each of the classes, softmax performs slightly worse than the OVA, softmax is in general better than OVA since the 10 classifiers trained in OVA are trained independently, while in softmax, the loss function considers the error of all classes. From the overall accuracy, I recommend softmax.

7.10 Building GDA classifiers for CIFAR10 (10 points)

For Linear Discriminant Analysis, test accuracy is 0.3708

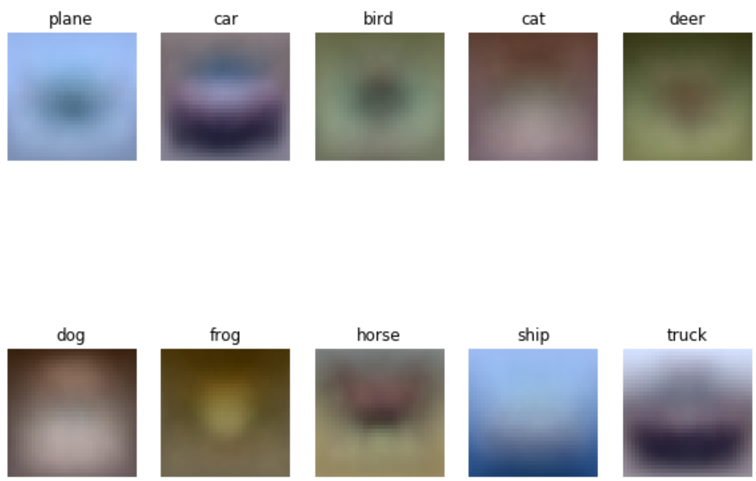

Figure 6: Visualization of coefficients for LDA

Visualization of covariance matrix can be seen in notebook. GDA has similar test accuracy comparing to OVA and Softmax, but the weight is more blur and not pixel-level.