CSC412

Assignment 3

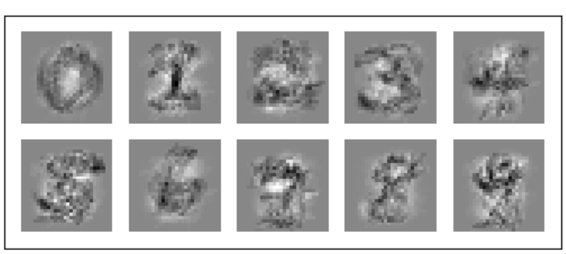
Vibhavi Peiris
Student #: 1000597687

April 6, 2018

1. Problem 1: L2-Regularized Logistic Regression
    a. *Use code from A2 with 300 training points*

```
#A3 Q1A From my-A2 Q3C code
def one_per_class(images, labels):
   out_images = np.zeros((10,images.shape[1]))
   out_labels = np.zeros((10,10))
   classes = np.where(labels == 1)[1] # get the class digit for each image by getting column idx of ones in labels
   #get first image in training set with each class label
   for i in range(0,10):
      img_num = np.where(classes == i)[0][0]
      out_images[i,:] = images[img_num,:]
      out_labels[i,:] = labels[img_num,:]
   return out_images,out_labels

def cost_function(w):
   sum_final = 0 #temporary create sum_final var
   dem = logsumexp(np.dot(np.transpose(w),grad_images))

   #mutliclass likelihood function is sum from 0 to k of label*predictive_log_likelihood
   for k in range(0,10):
      log_pc_x = np.dot(np.transpose(w[:,k]),grad_images) - dem
      if k == 0:
         sum_final = np.dot(grad_labels[k],log_pc_x)
      else:
         sum_final = sum_final + np.dot(grad_labels[k],log_pc_x)
   return sum_final

def logistic_gradient_desc(iterations,lr):
   #set globals so that cost function can access these values after usign autograd w.r.t. w
   global current_c
   global grad_images
   global grad_labels

   w = np.zeros((784,10)) #create the weights
   for i in range(0,iterations):
      for img_num in range(0,new_images.shape[0]):
          #get gradient of cost function/likelihood
         grad_images = new_images[img_num,:] #get current image
         grad_labels = new_labels[img_num,:] #get labels for current image
         current_c = img_num  #sinces we sampled 1 image for each class in order c = img_num
         cost_grad = elementwise_grad(cost_function)

         #update weights
         w = w + lr*cost_grad(w)
      print(i)
   return w

#run A2 Q3C code
new_images = train_images
new_labels = train_labels
grad_images=grad_labels = new_images #temporary just to create a gobal var for use with autograd
current_c = 0 #temporary just to create a gobal var for use with autograd
weights = logistic_gradient_desc(1000, 0.01) #5000 iterations with a common learning rate of 0.01
save_images(np.transpose(weights),'Q1a')
```

```
#A2 Q3d code
def avg_pred_log(w,images):
  log_pc_x = 0
  for i in range(0,images.shape[0]):
    current_log_pc_x = np.dot(np.transpose(w),images[i,:]) - logsumexp(np.dot(np.transpose(w),images[i,:]))
    log_pc_x = log_pc_x + current_log_pc_x

  return np.sum(log_pc_x)/float(images.shape[0])

def predict_regression(images, w):
  predictions = np.zeros((images.shape[0],w.shape[1])) #N by 10

  #find best class true class for each image
  for i in range(0,images.shape[0]):
    best_class = np.argmax(np.dot(np.transpose(w),images[i,:]))   #choose the class with highest
    predictions[i,best_class] = 1  #set index = digit to 1 as it is the best prediction for current image

  return predictions
def Q3D_report(train_images,train_labels,test_images,test_labels,w):
  #average for train
  avg_likelihood_train = avg_pred_log(w,train_images)
  print("Avg train log likelihood ",avg_likelihood_train)

  #average for test
  avg_likelihood_test = avg_pred_log(w,test_images)
  print("Avg test log likelihood ",avg_likelihood_test)

  #predictions for train
  predict_train = predict_regression(train_images, w)
  total_correct_train = np.sum(np.nonzero(predict_train)[1] == np.nonzero(train_labels)[1]) #get total number of correct predictions
  accuracy_train = total_correct_train/float(train_labels.shape[0])    #get accuracy
  print('Train Accuracy ',accuracy_train)

  #predictions for test
  predict_test = predict_regression(test_images, w)
  total_correct_test = np.sum(np.nonzero(predict_test)[1] == np.nonzero(test_labels)[1]) #get total number of correct predictions
  accuracy_test = total_correct_test/float(test_labels.shape[0])    #get accuracy
  print('Test Accuracy ',accuracy_test)


#run A2 Q3D code
Q3D_report(train_images,train_labels,test_images,test_labels,weights) #REPORT FOR A3 Q1a
```



Average train log likelihood -117.989022988
Average test log likelihood -97.3003729208
Train Accuracy 1.0
Test Accuracy 0.7727

b. *MAP Estimator*

$$N\left(w_{cd} \mid 0, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-w^2}{2\sigma^2}\right)$$

$$\log\left(N\left(w_{cd} \mid 0, \sigma^2\right)\right) = \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{w^2}{2\sigma^2} = -\log\left(\sqrt{2\pi\sigma^2}\right) - \frac{w^2}{2\sigma^2}$$

$$\log\left(p\left(t \mid \mathrm{X}, w\right) p\left(w \mid \sigma^2\right)\right) = \log\left(\prod_{i=0}^{300}\left(\frac{\exp\left(w_t^T x_i\right)}{\sum_{c=0}^{9}\exp\left(w_c^T x_i\right)}\right)\prod_{c=0}^{9}\prod_{d=0}^{784} N\left(w_{cd} \mid 0, \sigma^2\right)\right)$$

$$= \sum_{i=0}^{300}\log\left(\frac{\exp\left(w_t^T x_i\right)}{\sum_{c=0}^{9}\exp\left(w_c^T x_i\right)}\right) + \sum_{c=0}^{9}\sum_{d=0}^{784}\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-w^2}{2\sigma^2}\right)\right)$$

$$= \sum_{i=0}^{300}\left(w_{\mathbf{t}}^T x_i - \log\sum_{c=0}^{9}\exp\left(w_c^T x_i\right)\right) + \sum_{c=0}^{9}\sum_{d=0}^{784} -\log\left(\sqrt{2\pi\sigma^2}\right) - \frac{w^2}{2\sigma^2}$$

$$\nabla_w \log\left(p\left(t \mid \mathrm{X}, w\right) p\left(w \mid \sigma^2\right)\right) = \nabla_w \sum_{i=0}^{300}\left(w_{\mathbf{t}}^T x_i - \log\sum_{c=0}^{9}\exp\left(w_c^T x_i\right)\right) + \nabla_w \sum_{c=0}^{9}\sum_{d=0}^{784} -\log\left(\sqrt{2\pi\sigma^2}\right) - \frac{w^2}{2\sigma^2}$$

$$= \sum_{i=0}^{300}\left(x_i - \frac{\exp\left(w_c^T x_i\right) x_i}{\sum_{c=0}^{9}\exp\left(w_c^T x_i\right)}\right) - \frac{w}{\sigma^2}$$

c. Fit map

```
#A3 Q1C

#logitsic regression with gradient descent using map
def grad_desc(iterations,lr,sigma):
    #set globals so that cost function can access these values after usign autograd w.r.t. w
    global grad_images
    global grad_labels

    w = np.zeros((784,10)) #create the weights
    for i in range(0,iterations):
        for img_num in range(0,new_images.shape[0]):

            #get gradient of cost function/likelihood
            grad_images = new_images[img_num,:] #get current image
            grad_labels = new_labels[img_num,:] #get labels for current image
            cost_grad = elementwise_grad(cost_function)

            #update weights
            w = w + lr*cost_grad(w)

        #NEW ADDITION FOR A3
        w = w - w/sigma**2

        print(i)
    return w


#run cod for A3 q1c
print("Map logitsic regression")

##Testing for best sigma value was 36
#for i in range(1,10):
    #sigma = i**2  # from 5 to 100
    #print(sigma)
    #map_weights = grad_desc(100, 0.01,sigma) #5000 iterations with a common learning rate of 0.01
    #save_images(np.transpose(map_weights),'Q1c')
    #Q3D_report(train_images,train_labels,test_images,test_labels,map_weights)


    #sigma = 1/i**2   #from 1 to 1/100
    #print(sigma)
    #map_weights = grad_desc(100, 0.01,sigma) #5000 iterations with a common learning rate of 0.01
    #save_images(np.transpose(map_weights),'Q1c')
    #Q3D_report(train_images,train_labels,test_images,test_labels,map_weights)
sigma = 36
map_weights = grad_desc(1000, 0.01,sigma) #5000 iterations with a common learning rate of 0.01
save_images(np.transpose(map_weights),'Q1c')
Q3D_report(train_images,train_labels,test_images,test_labels,map_weights)
```

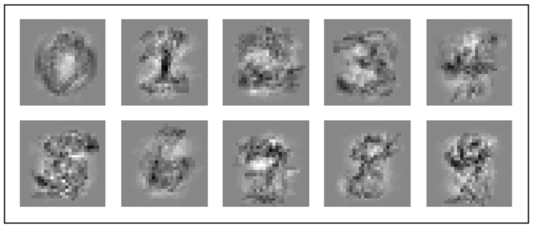
Average train log likelihood -93.0039030291

Average test log likelihood -77.2325259507

Train Accuracy 1.0

Test Accuracy 0.7711

Best sigma value was 36. With 4,9,16,25 close behind.



2. Problem 2: Bayesian Logistic Regression using Stochastic Variational Inference
   a. *Number of parameters?*
      For w it is 784 * 10 = 7840 parameters.
      For ϕ its (mean + standard deviation) =7840+7840 = 14960 parameters.
   b. *Code SVI.*

```
def elbo_estimate(var_params, logprob, num_samples, rs):
    """Provides a stochastic estimate of the variational lower bound.
    var_params is (mean, log_std) of a Gaussian."""
    mean, log_std = var_params
    samples = sample_diag_gaussian(mean,log_std,num_samples,rs)
    log_ps = logprob(samples)
    log_qs = diag_gaussian_log_density(samples,mean,log_std)
    E_q = np.sum(log_ps-log_qs)/num_samples   # E_q(z|x)[log p(x,z) - log q(z|x)]
    return E_q

def logprob_given_data(params):
    data_logprob = logistic_logprob(params,train_images,train_labels)
    prior_logprob =  np.sum(np.sum(-np.log(np.sqrt(2*np.pi*prior_std))-(params**2)/(2*prior_std),axis=2),axis=1)
    return data_logprob + prior_logprob
```

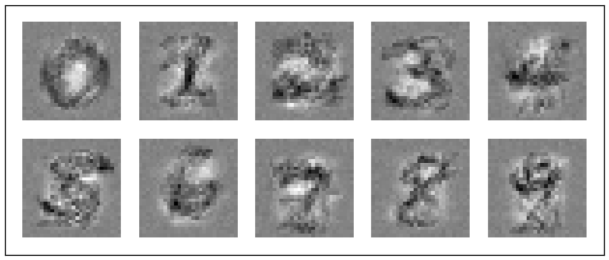
   c. Compute accuracy for test

```
predict_test = predict_regression(test_images, np.transpose(optimized_params[0])) #A3 q1/A2 code
total_correct_test = np.sum(np.nonzero(predict_test)[1] == np.nonzero(test_labels)[1]) #get total number of correct predictions
accuracy_test = total_correct_test/float(test_labels.shape[0])    #get accuracy
print('\nTest Accuracy ',accuracy_test)

##testing a bunch of std values, std = 1 is best with 77.68% and std =9 is second best 77.09%
#for i in range(1,10):
   #prior_std = i**2  # from 5 to 100
   #print(prior_std)
   #optimized_params = adam(objective_grad, init_params, step_size=0.05, num_iters=100, callback=print_perf)
   #predict_test = predict_regression(test_images, np.transpose(optimized_params[0])) #A3 q1/A2 code
   #total_correct_test = np.sum(np.nonzero(predict_test)[1] == np.nonzero(test_labels)[1]) #get total number of correct predictions
   #accuracy_test = total_correct_test/float(test_labels.shape[0])    #get accuracy
   #print('\nTest Accuracy ',accuracy_test)

   #prior_std = 1/i**2   #from 1 to 1/100
   #print(prior_std)
   #optimized_params = adam(objective_grad, init_params, step_size=0.05, num_iters=100, callback=print_perf)
   #predict_test = predict_regression(test_images, np.transpose(optimized_params[0])) #A3 q1/A2 code
   #total_correct_test = np.sum(np.nonzero(predict_test)[1] == np.nonzero(test_labels)[1]) #get total number of correct predictions
   #accuracy_test = total_correct_test/float(test_labels.shape[0])    #get accuracy
   #print('\nTest Accuracy ',accuracy_test)
```

Standard deviation = 1 is the best with 77.68% and Standard deviation = 9 is second best 77.09%. These number are only slightly higher than MAP inferences accuracy.

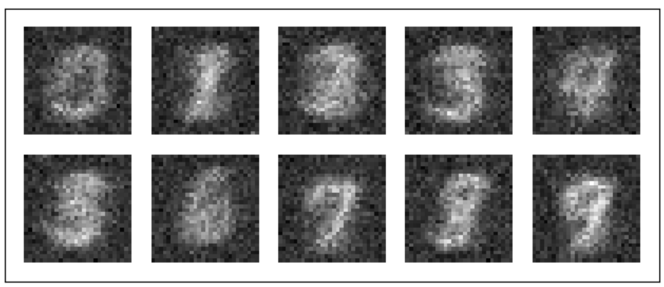
d. Plot 10 images

i. Mean



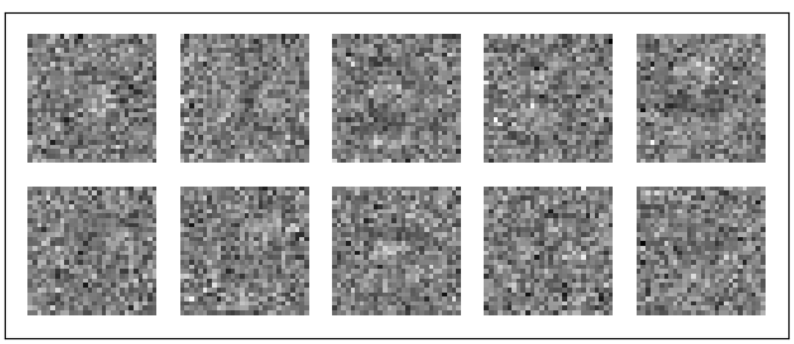
The mean is as I expected because the darkest regions would be the most common pixels of a digit (middle for 1).

ii. Variation



The variation would have the opposite effect of mean because the further away (up to a certain distance) from the digits mean locations the more variation on pixel values. I.E. for digit 1 the close pixels around the one would have high variation (due to all possible ways one is drawn)

iii. Samples



The sample is not what I expected, there is more noise. I expected it to be a combination of mean and variance.

e. *Single sample q*

When $x_d \epsilon B$ that means that $x_d = 0$. So $w_{cd}^T x_d$ will be 0. Which means it does not affect the optimal because p(t|w,x) is 0 (substitute $w_{cd}^T x_d=0$ into the p(t|w,x equation).

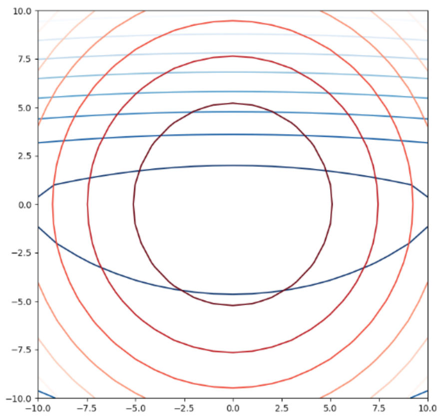
f. Posterior questions

i. Does training affect posterior and variational?

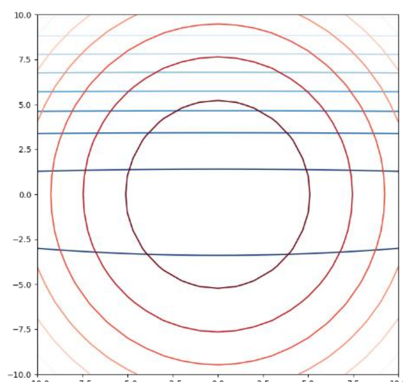
No, true posterior seems to stay the same through the training. Yes, Variational posterior changes.

ii. How does standard deviation affect posterior?

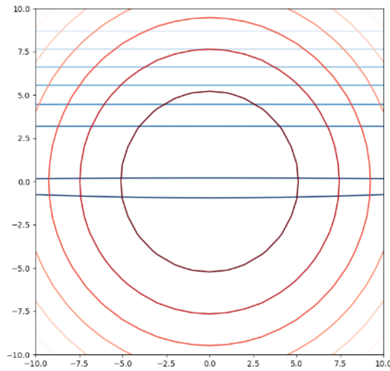
Increasing standard deviation makes the true posterior wider across the horizontal.



Standard deviation = 1



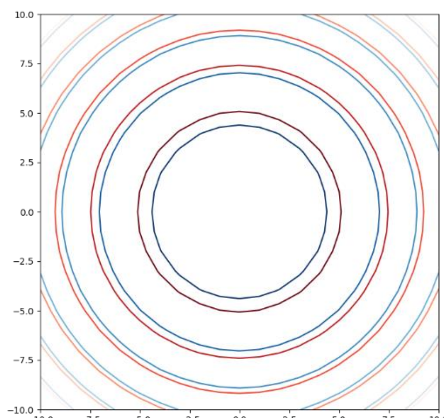
standard deviation = 10



standard deviation = 100

iii. *Px2's affect?*
It makes the true posterior oval shaped & different from variational posterior.

iv. *Changing px2's affect?*
It makes true posterior more circular similar to variational.



v. *Using improper flat model?*
No