to plot something, please include the figure as well as the code used to plot it (and clearly explain in the README what the relevant files are). If multiple entities appear on a plot, make sure that they are clearly distinguishable (by color or style of lines and markers). When asked to provide a brief explanation or description, try to make your answers concise, but do not omit anything you believe is important.

When submitting code, please make sure it's reasonably documented, and describe succinctly what is done in the relevant bits of code in the notebook.

1 Generative models

Here we will consider the Gaussian generative model for $\mathbf{x} \in \mathbb{R}^d$ which assumes equal, *isotropic* covariance matrices for each class:

$$\mathbf{\Sigma}_c = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & \sigma_d^2 \end{bmatrix}. \tag{1}$$

The classes of course have separate means. That is, the conditional density of the j-th coordinate of $\mathbf{x}$ given class c is a Gaussian $\mathcal{N}(\mu_{c,j}, \sigma_j)$, with these densities independent (given class) for different values of j.

For simplicity, let's consider a two-class problem, with $y \in \{0, 1\}$. As we discussed in class, when training the generative model, we simply fit $p(\mathbf{x} \mid y)$ and $p(y)$ to the training data, and use the resulting discriminant analysis to produce a decision rule which predicts

$$\widehat{y}(\mathbf{x}) = \underset{c}{\operatorname{argmax}} \left\{ p(\mathbf{x} \mid y = c)\, p(y = c) \right\}. \tag{2}$$

However, this model does also produce an (implicit) estimate for the posterior $p(y = c \mid \mathbf{x})$.

Problem 1 [30 points]
Show that the posterior $p(y = c \mid \mathbf{x})$ resulting from the generative model above has the same form as the posterior in logistic regression model,

$$p(y = c \mid \mathbf{x}) = \frac{1}{1 + \exp(w_0 + \mathbf{w} \cdot \mathbf{x})}, \tag{3}$$

for appropriate values of $w_0 \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d$.

End of problem 1

Advice: Start with Bayes rule, and use the simplifying assumptions in (1) *to derive the posterior.*

Problem 2 **[10 points]**
The previous problem established that the two models – logistic regression and the linear discriminant analysis based on the isotropic Gaussian model (1) – have the same form of the posterior $p\,(y\,|\,\mathbf{x})$. Will the two models produce the same classifier when applied to a given training set? Why or why not?

End of problem 2

2 Multilayer neural networks

In this section we will return to the problem of handwritten digit recognition and attack it with a two-layer neural network. The general architecture of the network is as follows: let $\mathbf{x}$ be the input (pixels of the $d \times d$ digit image). Then, the first layer (with k hidden units) computes

$$\mathbf{f}_1(\mathbf{x}) \;=\; h\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right), \tag{4}$$

where $\mathbf{W}_1$ is a $k \times d^2$ matrix of weights, $\mathbf{b}_1$ is the k-dimensional vector of bias terms, and h is a nonlinearity (activation function for the hidden units); the syntax in (4) means that h is computed on each element of its k-dimensional argument vector (the output of $\mathbf{W}_1\mathbf{x}+\mathbf{b}_1$), yielding k-dimensional activation vector, called in neural network literature the *feature map* of the layer.

The second layer computes the network activation:

$$\mathbf{f}_2(\mathbf{x}) \;=\; \mathbf{W}_2\mathbf{f}_1(\mathbf{x}) + \mathbf{b}_2, \tag{5}$$

where $\mathbf{W}_2$ is a $10 \times k$ matrix and $\mathbf{b}_2$ a 10-dimensional vector. The values of the 10-dimensional vector $\mathbf{f}_2$ are interpreted as the scores (unnormalized log-probabilities) for the 10 classes. So, the posterior for class c computed by the network is

$$p(c|\mathbf{x}) \;=\; \frac{\exp(f_{2,c}(\mathbf{x}))}{\sum_j \exp(f_{2,j}(\mathbf{x}))}, \tag{6}$$

and the loss used to train the network is

$$-\frac{1}{n}\sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i) \tag{7}$$

where the summation is over the indices of training examples. You can of course add regularization (e.g., squared norm penalty over elements of $\mathbf{W}_1$, $\mathbf{W}_2$) to the loss.

Again, we will revisit and improve our results on a familiar data set, this time MNIST.

Problem 3 [35 points]
Implement a two-layer neural network for MNIST. We have set up a Kaggle competition for the *clean* version:

- `https://inclass.kaggle.com/c/mnist-clean-hw5-ttic31020`

We have provided skeleton code `PS5_neuralnet.ipynb` that features a complete forward and backward pass. However, you will have to work out what the gradients are.

We provide several different update algorithms in `utils.py`: SGD, SGD with momentum and SGD with nesterov momentum. Try these methods, and report the differences in their performances.

You should also decide what the activation function h is. The options are ReLU, Logistic and Tanh. For all activations you try, you need to implement their forward and backward functions.

This task will require you to understand the code that we have provided, so read the inline documentation carefully. We encourage you to re-use parts of your code and hyperparameters from Problem Set 2, e.g. `softmaxloss`, stopping criteria, stepsize reduction, batch_size, etc.

If you used loops in your solution in Problem Set 2, you will quickly find that it is not fast enough. Learn to write things in terms of matrix multiplications.

Optional:

More layers You can try more than two-layers, and it should be trivial given the code we provide.

Dropout You could also add Dropout after the activation layer.

End of problem 3