We will now focus on reconstructing images from patterns of human brain activity. You do not need to use Chainer for this assignment.

Read the paper

Schoenmakers, S, Barth, M, Heskes, T, van Gerven, MAJ. Linear reconstruction of perceived images from human brain activity. Neuroimage. 2013; 83:951-961

Now download the 69 dataset from: http://www.ccnlab.net/data/. The dataset contains fMRI data acquired from the early visual cortex of one subject as the subject was presented with 100 grayscale images of handwritten sixes and nines (50 sixes and 50 nines). The fMRI data has been realigned and slice time corrected. Furthermore, stimulus specific response amplitudes have been estimated with a general linear model.

Let's first familiarize ourselves with the dataset. It contains a number of variables:

> I: This is a 100 x 784 matrix. The ith row contains the pixel values of the stimulus that was presented in the ith trial of the experiment. Note that the stimuli are 28 pixel x 28 pixel images, which were reshaped to 1 x 784 vectors.
>
> Y: This is a 100 x 3092 matrix. The ith row contains the voxel values of the responses that were measured in the ith trial of the experiment.
>
> X_prior: This is a 2000 x 784 matrix. Each row contains the pixel values of a different stimulus, which was not used in the experiment.

Load the dataset. Normalize X and Y to have zero mean and unit variance.[1] Split X and Y in two parts called X_train and X_test, and Y_train and Y_test. The training set should contain 80 stimulus-response pairs (40 pairs for sixes and 40 pairs for nines). The test set should contain 20 stimulus-response pairs (10 pairs for sixes and 10 pairs for nines). In the remainder of this assignment, we will use **x** for referring to a 784 x 1 stimulus vector and **y** for referring to a 3092 x 1 response vector.

Our goal is to solve the problem of reconstructing **x** from **y**. One possible approach to solve this problem is to use a discriminative model. Discriminative models predict **x** as a function of **y**. That is:

$$\boldsymbol{x} = f(\boldsymbol{y})$$

We will assume that f is a linear function. That is:

$$\boldsymbol{x} = B^T \boldsymbol{y}$$

This linear regression model can be seen as a very simple neural network consisting of one layer of weights (i.e., B). We can estimate B with ridge regression. That is:

$$B = ({Y_{train}}^T Y_{train} + \lambda I)^{-1} {Y_{train}}^T X_{train}$$

[1] Recall that normalization means subtracting the mean of each pixel/voxel from itself and dividing it by its standard deviation. Note that you need to undo this operation when you want to visualize the stimuli.

where λ is the regularization coefficient, I is the q x q identity matrix, and q is the number of voxels. Note that we can safely ignore the intercept since we normalized our data to have zero mean and unit variance.

1) Estimate B on the training set. Normally, you should use cross validation to estimate lambda. For simplicity, you can assume that $\lambda = 10^{-6}$. Reconstruct **x** from **y** in the test set. Visualize the original test stimuli and their reconstructions.

Another possible approach to solve the problem of reconstructing **x** from **y** is to use a generative model and invert it by applying Bayes' rule. We reformulate the problem as finding the most probable **x** that could have caused **y**. That is:

$$arg\ max_{x}\, p(\boldsymbol{x} \mid \boldsymbol{y})$$

where p(**x** | **y**) is called the posterior (probability of the stimulus being **x** if the observation is **y**). In other words, we have to define the posterior, estimate its parameters and find the argument that maximizes it, which will be the reconstruction of **x** from **y**. While, this may seem daunting, it actually has a simple solution. The posterior assigns a probability to an event by combining our observations and beliefs about it, and can be decomposed with Bayes' theorem as the product of how likely our observations are given the event (probability of observing **y** if the stimulus is **x**) and how likely the event is independent of our observations (probability of the stimulus being **s**). That is:

$$p(\boldsymbol{x} \mid \boldsymbol{y}) \ \propto\ p(\boldsymbol{y} \mid \boldsymbol{x})\, p(\boldsymbol{x})$$

where p(**y** | **x**) is called the likelihood and p(**x**) is called the prior.

We will assume that the likelihood and the prior are multivariate Gaussian distributions. A Gaussian is characterized by two parameters: a mean vector and a covariance matrix. In the case of the likelihood, the mean of the Gaussian is given by:

$$\boldsymbol{\mu} \ = \ B^{T}\, \boldsymbol{x}$$

As before, we can estimate B with ridge regression:

$$B \ = \ ({X_{train}}^{T}\, X_{train} \ + \ \lambda\, I)^{-1}\, {X_{train}}^{T}\, Y_{train}$$

where λ is the regularization coefficient, I is the p x p identity matrix, and p is the number of pixels.

The covariance matrix of the likelihood is given by:

$$\Sigma \ = \ diag(E[||\boldsymbol{y} \ - \ B^{T}\, \boldsymbol{x}||^{2}]).$$

In the case of the prior, the mean of the Gaussian is given by:

$$\boldsymbol{\mu}_{prior} = \boldsymbol{0}$$

The covariance matrix of the prior is given by:

$$\Sigma_{prior} \ = \ {X_{prior}}^{T}\, X_{prior}\ /\ (n \ - \ 1)$$

where n is the number of items in X_{prior}.

2) Estimate B on the training set. Normally, you should use cross-validation to estimate lambda and Σ. For simplicity, you can assume that $\lambda = 10^{-6}$ and $\Sigma = 10^{-3}$I. Estimate Σ_{prior}. Tip: Add 10^{-6} to the diagonal of Σ_{prior} for regularization. Visualize Sigma_prior using the imshow function. Can you explain what it shows?

Having defined the likelihood and the prior as Gaussians, we can derive the posterior by multiplying them. It turns out that the product of two Gaussians is another Gaussian, whose mean vector is given by:

$$\boldsymbol{\mu}_{post} = \left({\Sigma_{prior}}^{-1} + B\,\Sigma^{-1}\,B^{T}\right)^{-1} B\ \Sigma^{-1}\,\boldsymbol{y}$$

We are almost done. Recall that the reconstruction of x from y is the argument that maximizes the posterior, which we derived to be a Gaussian. We will be completely done once we answer the following question: What is the argument that maximizes a Gaussian? The answer is its mean vector, which is the solution of our initial problem. That is:

$$arg\ max_{\boldsymbol{x}}\, p(\boldsymbol{x} \mid \boldsymbol{y}) = \boldsymbol{\mu}_{post}$$

Now, we can plug any **y** in the above equation and reconstruct the most probable **x** that could have caused it.

2) Reconstruct **x** from **y** in the test set using the generative approach. Visualize the reconstructions. Compare the reconstructions with the earlier reconstructions. Which one is better? Why? Can you think of ways to improve the results using neural networks?