1. Q: **What exactly is**  \( \text{wts, b, wts\_grad, wts\_inc, b\_grad, b\_inc, and input\_grad} \)** supposed to be?  

   **A:** Please see the following: 
   
   - \( \text{wts} \): weights for each layer.
   - \( \text{b} \): bias for each layer.
   - \( \text{wts\_grad} \): gradient for weights you calculated from \text{back\_prop} for each layer.
   - \( \text{wts\_inc} \): actual update you will do for \( \text{wts} \) in a SGD step for each layer.
   - \( \text{b\_grad} \): gradient for bias you calculated from \text{back\_prop} for each layer.
   - \( \text{b\_inc} \): actual update you will do for \( \text{b} \) in a SGD step for each layer.
   - \( \text{act\_grad} \text{ in @layer/back\_prop} \): gradient wrt activation function of this layer.
   - \( \text{input\_grad} \text{ in @layer/back\_prop} \): gradients wrt the input of this layer.

2. Q: **What do you expect us to implement?**

   **A:** In general your task is to:  
   
   1. implement the following functions:
      
      - \( \text{@nn/fwd\_prop} \): perform a feedforward pass over all layers, and return a list of (cell in \text{matlab}) activations for each layer. You may want to call \( \text{@layer/fwd\_prop} \) for each layer.
      - \( \text{@nn/back\_prop} \): perform a backpropagation, return \text{self} with updated gradient of weights and biases for all layers. You may want to call \( \text{@layer/back\_prop} \) for each layer.
      - \( \text{@nn/apply\_gradients} \): perform stochastic gradient descent step. You may want to call \( \text{@layer/apply\_gradients} \) for each layer.
@layer/fwd_prop: perform a forward pass.
@layer/back_prop: back propagate activation gradients and compute gradients for one layer. The output is a struct consisting of three parts, wts_grad, b_grad, input_grad. Please refer to Q1 for their meaning.
@layer/compute_act_gradients_from_targets: compute the gradients wrt activations of sigmoid layer, the input are the current activations of this layer and the gradients wrt outputs of the sigmoid. This function is needed for sigmoid layer.
@layer/compute_act_grad_from_output_grad: compute the gradients wrt activations of the softmax layer, given the targets and the outputs of the softmax, the inputs are the current activations of this layer and the target.
@layer/apply_gradients: update wts_inc(b_inc) and use wts_inc(b_inc) to update the weight(bias). You may want to use the gradient wts_grad(b_grad) as well as momentum and learning rate.

2. choose proper hyper-parameters in train_nnet:
   - eps, learning rate for SGD
   - l2, coefficient for $\ell_2$ regularizer
   - momentum, momentum for SGD
   - batch_size, batch size for SGD

3. implement stopping criterion in train_nnet. Consider how to control overfit.

4. apply pre-processing in creat_pred and train_nnet. You may want to use functions implemented in speech_data.

3. Q: Do we have to use both svd and eig in order to calculate PCA?
   
   A: No. You may use either one of them. Try to think about the difference between two approaches.