Answer Key for Exam $\boxed{\text{A}}$

Section 1. Multiple choice questions

Instructions: Each question may have any number of correct choices. Clearly mark (tick) all choices you believe to be correct (1 mark for each correct choice, −0.5 for each incorrect choice).

1. Which of the following might be valid reasons for preferring an SVM over a neural network?

   (a) An SVM can automatically learn to apply a non-linear transformation on the input space; a neural net cannot.

   [✓ (b)] An SVM can effectively map the data to an infinite-dimensional space; a neural net cannot.

   [✓ (c)] An SVM should not get stuck in local minima, unlike a neural net.

   (d) The transformed (basis function) representation constructed by an SVM is usually easier to visualise/interpret than for a neural net.

2. You are given a labeled binary classification data set with N data points and D features. Suppose that $N < D$. In training an SVM on this data set, which of the following kernels is likely to be most appropriate?

   [✓ (a)] Linear kernel

   (b) Quadratic kernel

   (c) Higher-order polynomial kernel

   (d) RBF kernel

3. You are training an RBF SVM with the following parameters: C (slack penalty) and $\gamma = 1/2\sigma^2$ (where σ^2 is the variance of the RBF kernel). How should you tweak the parameters to reduce overfitting?

   (a) Increase C and/or reduce γ

   (b) Reduce C and/or increase γ

   [✓ (c)] Reduce C and/or reduce γ

   (d) Reduce C only (γ has no predictable effect on overfitting)

   (e) Increase C only (γ has no predictable effect on overfitting)

4. For an RBF SVM with a particular pair of randomly chosen values of the hyperparameters C and $\gamma = 1/2\sigma^2$ (where σ^2 is the variance of the RBF kernel), which of the following tests can be taken to indicate that the chosen values likely correspond to overfitting?

   (a) When I increase C the training and validation accuracies both increase.

   [✓ (b)] When I decrease C the training accuracy reduces, but validation accuracy increases.

   (c) When I decrease C the training and validation accuracies are both reduced.

   (d) When I decrease γ the number of support vectors reduces.

   [✓ (e)] When I decrease γ the training accuracy reduces, but validation accuracy increases.

   [✓ (f)] When I increase γ the training accuracy increases, but validation accuracy reduces.

5. Consider the following possible choices of error function in training a neural network for classification: cross-entropy error (I), classification error (II), and sum-of-squares error (III). Which of the following are true?

   (a) (II) is problematic because it's non-differentiable, but either of (I) or (III) should give the same result.

   (b) Any of the three could be easily used for backpropagation, but (I) is preferred because it corresponds to maximising the likelihood of the data.

   $\boxed{\text{(c)}}$ (II) is problematic because it's non-differentiable; (I) is preferred to (III) because the latter corresponds to an inappropriate noise model.

   (d) (II) is problematic because it's non-differentiable; (III) is preferred to (I) because the former corresponds to maximising the likelihood of the data.

   (e) Any of the three could be easily used for backpropagation, but (III) is preferred because it corresponds to maximising the likelihood of the data.

Section 2. Numerical/Short-answer questions

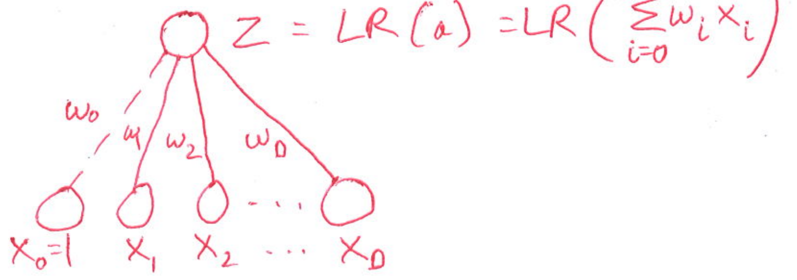
Instructions: Please write *only the final answers* on this question paper, in the space provided for each item. The provided answer script should be used for all working, but will not be graded. However, in case of any doubt regarding your answers, we may refer to the answer script to check your working. So please try to write out your working as clearly as possible.

6. We have seen that the hidden units in neural nets can implement any suitable non-linear activation function. One of the most widely used activation functions nowadays is the *Leaky ReLU* (Rectified Linear Unit), which we will define as follows:

$$LR(a) = \begin{cases} a & \text{if } a > 0 \\ 0.01a & \text{otherwise} \end{cases}$$

Consider a single hidden unit called z, which implements the above activation function. Take z to be connected to D input features $x_1, x_2, ..., x_D$, with corresponding weights $w_1, w_2, ..., w_D$.

(a) Draw a picture of the unit z with all its incoming connections and the corresponding input units. Also include the bias term w_0 in your picture via a dummy input variable. All units and weights should be labeled clearly. You picture should include the expression for the value of z as a function of the input units. [2]



(b) For backpropagation, we wish to calculate the error gradient with respect to each of these weights just defined, i.e., $\frac{\partial E(\mathbf{w})}{\partial w_i}, i \in \{0, 1, 2, ...D\}$. Apply the chain rule using z as the intermediate variable, to write this partial derivative as a product of two partial derivatives. Where will the value of the first one come from? [1]

$$\frac{\partial E(\underline{w})}{\partial w_i} = \boxed{\frac{\partial E(\underline{w})}{\partial z}} \cdot \frac{\partial z}{\partial w_i}$$

δ at z, comes from backprop. above z

(c) Here we wish to show the calculation for the second partial derivative obtained via the above chain rule. To do this, we will first need to obtain the derivative of the *Leaky ReLU* function with respect to its argument. Write down (in an appropriate form) the value of this derivative, i.e., $\frac{\partial LR(a)}{\partial a}$. Is there any value of a where it is undefined? [2]

$$\frac{\partial LR(a)}{\partial a} = \begin{cases} 1 & \text{if } a > 0 \\ 0.01 & \text{if } a < 0 \end{cases}$$

Undefined at $\underline{a = 0}$

(d) Now write down (again, in appropriate form) the entire expression for the value of the second partial derivative from the above chain rule. **[1.5]**

$$\frac{\partial z}{\partial w_i} = \begin{cases} x_i & \text{if } \sum_{i=0}^{D} w_i x_i > 0 \\ 0.01 x_i & \text{if } \sum_{i=0}^{D} w_i x_i < 0 \end{cases}$$

(e) Is there any specific situation in which backpropagation through a Leaky ReLU hidden unit would be theoretically ill-defined? If so, what is it, and why do you think Leaky ReLUs are still usable in practice? **[1.5]**

Yes, at $\underline{\sum_{i=0}^{D} w_i x_i = 0}$.

In practice, exact value of 0 is only infinitesimally likely.

7. Consider the Google/Stanford unsupervised deep learning example which was discussed in yesterday's class.

(a) At first glance, there is something self-contradictory about a modelling approach which uses both pooling and sparse autoencoding: as discussed, the former reduces the dimensionality of the representation, whilst the latter typically increases it. Why do you think they chose to combine both of these ideas? **[3]**

Removing irrelevant features/info

Pooling: Should find lower-dim. structure (manifold) in original space, which is very high-dim.

Grouping using only relevant info

Sparse autoencoder: To obtain fine-grained clustering of frequent patterns, on top of identified low-dim. repn.

(b) They demonstrated the usefulness of their representation by showing that it improves performance on an image classification data set (ImageNet). But, since the ImageNet labeled data is available, they could simply have trained a supervised CNN using it to learn the representations. Why do you think they chose to attempt ImageNet classification using features obtained from an unsupervised sparse autoencoder instead? **[1]**

They could use even more (+ more diverse) data which was unlabeled, e.g., millions of Youtube images. So hopefully could get a very generic image repn.