INFO 7390
Advances in Data Sciences and Architecture
Exam 2

Student Name: _____Zixuan Yu_____
Professor: Nik Bear Brown

Due: August 16 Thursday, 2018

All the questions REQUIRE an explanation of the question. Yes or no responses get no credit. Any text or answers from the Internet MUST be cited. You MUST work alone on this; answers too similar to other students' answers will receive no credit.

Q1 (5 Points) What is "Deep Learning?" How does it different from traditional machine learning techniques such as SVMs, Naive Bayes, Decision Trees, etc? Why has it become so popular in the last decade?

1. Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised.
2. Comparing with Deep learning, traditional machine learning techniques can automatically learn representations from data such as images, video or text, without introducing hand-coded rules or human domain knowledge.
3. The reason why deep learning is popular in the last decade is that machine learning techniques can automatically learn representations from data such as images, video or text. It also can test accuracy, and change hyper-parameters to get better accuracy.

   Reference: https://en.wikipedia.org/wiki/Deep_learning

Q2 (5 Points) What are activation functions? What is their purpose? Must they be non-linear? Must they be continuously differentiable? Must they be monotonic? How does the derivative of the activation function effect the gradient?

1. Activation functions are an extremely important feature of the artificial neural networks. They basically decide whether a neuron should be activated or not.
2. Their main purpose is to convert an input signal of a node in an A-NN to an output signal. That output signal now is used as an input in the next layer in the stack.
3. No, the activation functions can be divided into 2 types: Linear Activation Function and Non-linear Activation Functions.
4. Yes, they must be continuously differentiable, because this property is desirable for enabling gradient-based optimization methods. The binary step activation function is not differentiable at 0, and it differentiates to 0 for all other values, so gradient-based methods can make no progress with it.
5. Yes, they must be monotonic, because when the activation function is monotonic, the error surface associated with a single-layer model is guaranteed to be convex.

Reference: https://en.wikipedia.org/wiki/Activation_function

Q3 (5 Points) What is backpropagation? Name three backpropagation algorithms and explain how they work. List the pros and cons of your three backpropagation algorithms.

1. Backpropagation is a method used in artificial neural networks to calculate a gradient that is needed in the calculation of the weights to be used in the network. It is commonly used to train deep neural networks, a term referring to neural networks with more than one hidden layer.
2. The ***trainlm*** is a network training function that updates weight and bias values according to Levenberg-Marquardt optimization.

   Pros: The ***trainlm*** is often the fastest backpropagation algorithm in the toolbox and is highly recommended as a first-choice supervised algorithm, although it does require more memory than other algorithms.

   Cons: This function uses the Jacobian for calculations, which assumes that performance is a mean or sum of squared errors. Therefore, networks trained with this function must use either the mse or sse performance function.

   The ***trainrp*** is a network training function that updates weight and bias values according to the resilient backpropagation algorithm (Rprop).

   Pros: The ***trainrp*** function is the fastest algorithm on pattern recognition problems. The memory requirements for this algorithm are relatively small in comparison to the other algorithms considered.
   Cons: it does not perform well on function approximation problems. Its performance also degrades as the error goal is reduced.

   The ***trainbfg*** is a network training function that updates weight and bias values according to the BFGS quasi-Newton method.

   Pros: The performance of ***trainbfg*** is like that of ***trainlm***. It does not require as much storage as ***trainlm***.

   Cons: the computation required does increase geometrically with the size of the network, because the equivalent of a matrix inverse must be computed at each iteration.

reference： https://en.wikipedia.org/wiki/Backpropagation

Q4 (5 Points) Name two forms of regularization used in neural networks. Explain how they work. Why does one use regularization with neural networks?

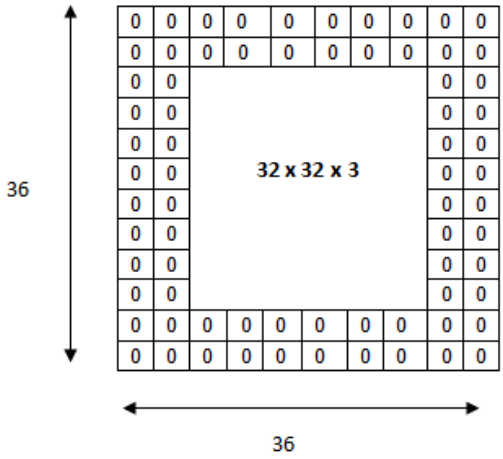
1. L1 regularization and L2 regularization.

2. (L1) Ridge regression adds “squared magnitude” of coefficient as penalty term to the loss function. And (L2) Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds “absolute value of magnitude” of coefficient as penalty term to the loss function.
3. Regularization is a form of regression, that constrains the coefficient estimates towards zero. This technique discourages learning a more complex or flexible model, to avoid the risk of overfitting.

REFERENCE: https://www.quora.com/Should-we-always-use-neural-networks-with-regularization

Q5 (5 Points) What is a convolution? Give an example of a convolution on image data with and without padding.

1. In mathematics (and functional analysis) convolution is a mathematical operation on two functions to produce a third function, that is typically viewed as a modified version of one of the original functions, giving the integral of the pointwise multiplication of the two functions as a function of the amount that one of the original functions is translated. Convolution is like cross-correlation. For discrete real valued signals, they differ only in a time reversal in one of the signals. For continuous signals, the cross-correlation operator is the adjoin operator of the convolution operator.
2. When you apply three 5 x 5 x 3 filters to a 32 x 32 x 3 input volume. The output volume would be 28 x 28 x 3. Notice that the spatial dimensions decrease. As we keep applying conv layers, the size of the volume will decrease faster than we would like. In the early layers of our network, we want to preserve as much information about the original input volume so that we can extract those low-level features. Let’s say we want to apply the same conv layer but we want the output volume to remain 32 x 32 x 3. To do this, we can apply a zero padding of size 2 to that layer. Zero padding pads the input volume with zeros around the border. If we think about a zero padding of two, then this would result in a 36 x 36 x 3 input volume.
3. Like the image below, if this image is without padding, after the conv layer the output volume would be 28 x 28 x 3, so the size of the volume will decrease faster than we would like. However, if this image is with padding like below image, the output volume would be kept as 32 x 32 x 3.



The input volume is 32 x 32 x 3. If we imagine two borders of zeros around the volume, this gives us a 36 x 36 x 3 volume. Then, when we apply our conv layer with our three 5 x 5 x 3 filters and a stride of 1, then we will also get a 32 x 32 x3 output volume.

Reference:
https://github.com/nikbearbrown/INFO_7390/blob/master/Week_10/NBB_Convolutions.ipynb
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

Q6 (5 Points) What is max-pooling? Why is it used? How does it compare to mean-pooling and min-pooling? Which would you use between max-pooling, mean-pooling and min-pooling?

1. Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.
2. This is done to in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.
3. max-pool layer compressed by taking the maximum activation in a block. If you have a block with mostly small activation, but a small bit of large activation, you will lose the information on the low activations. I think of this as saying, "this type of feature was detected in this general area", which is different from the mean-pooling and min-pooling.
4. I always use the max-pooling. Because Input is feed from one end and each layer produces activations which is fed to next layer. Finally, all of this is passed through a softmax which renormalized everything otllto sum to 1 and the, the index which has largest value is declared as the class of input. Back prop training also reinforces the same notion of correct class getting highest activations in last layer. Largest activations come from sum of larger activations of previous layer. The entire learning is all about getting the larger no in the correct place. So, when a decision had to be made as to which features to be allowed to pass through, the obvious choice is the maximum number. Hence max-pooling.

REFERENCE: https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks

Q7 (5 Points) What is the feature learning pipeline in a CNN? How does it relate to the classification in a CNN? Can the feature learning pipeline be used with techniques like SVMs, Naive Bayes, Decision Trees, or GBMs?

1. The feature learning pipeline is a series of steps: convolution, max-pooling, and finally a fully-connected network.
2. When we use CNN, our goal is to classify the different features in the image and identify their boundaries. And the feature learning pipeline in CNN is to classify and identify the feature, so CNN could do the classification based on the feature map produced in the pipeline.
3. Yes, sometimes, on the final layer of the CNN adds a Support Vector Machines (SVMs) that simply classifies whether this is an object, and if so what object, so we could also used with techniques like Naïve Bayes, Decision Trees of GBMs.

Reference: https://towardsdatascience.com/pipelines-mind-maps-and-convolutional-neural-networks-34bfc94db10c https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

Q8 (5 Points) What are loss functions? When would one choose cross-entropy vs mean-square error?

1. Loss functions provide more than just a static representation of how your model is performing – they're how your algorithms fit data in the first place. Most Machine Learning algorithms use some sort of loss function in the process of optimization, or finding the best parameters (weights) for your data.
2. (Rohan Varma) – "*Loss functions are a key part of any machine learning model: they define an objective against which the performance of your model is measured, and the setting of weight parameters learned by the model is determined by minimizing a chosen loss function. There are several different common loss functions to choose from: the cross-entropy loss, the mean-squared error, the Huber loss, and the hinge loss – just to name a few.*"

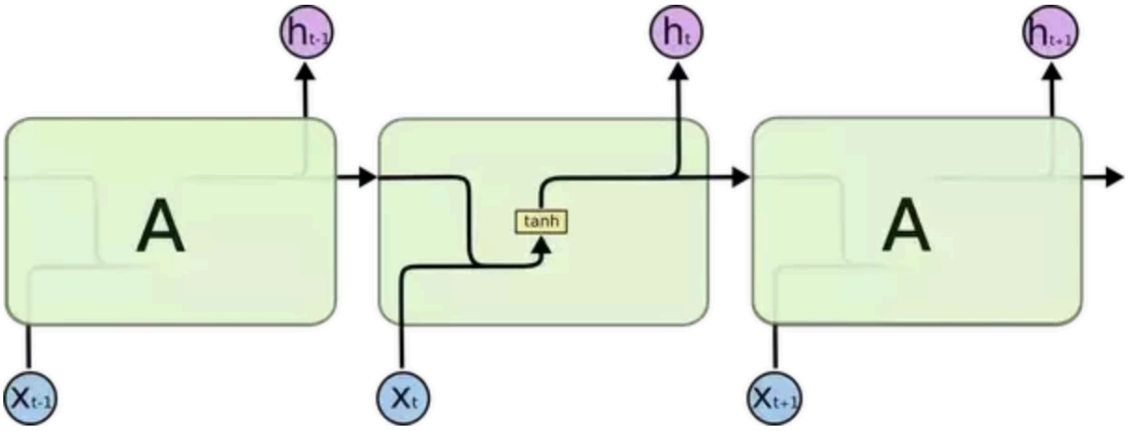
   Reference: https://blog.algorithmia.com/introduction-to-loss-functions/

Q9 (5 Points) What is an RNN? What kind of data is an RNN used for? How does a Vanilla RNN differ from an LSTM?

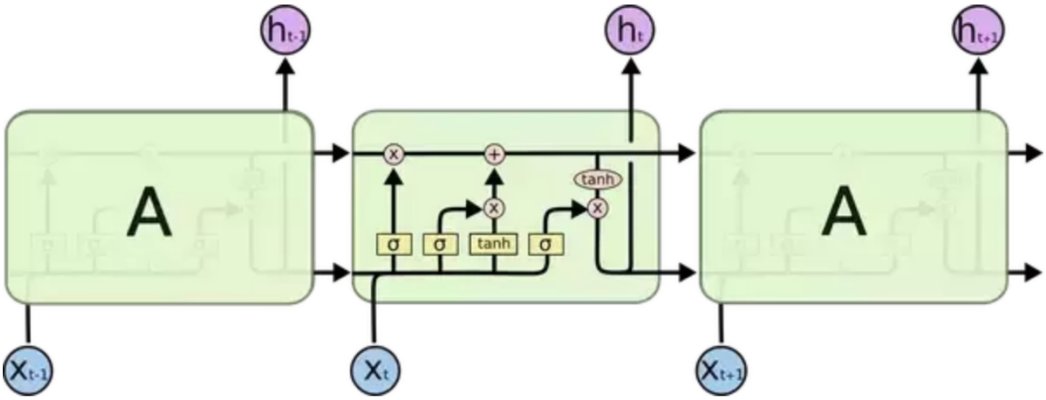
1. A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition[1] or speech recognition

Now, in the vanilla RNN, the internal of the cell is this:

2.



LSTM tries to prevent that.



3.

Reference: https://en.wikipedia.org/wiki/Recurrent_neural_network

Q10 (5 Points) What is a Markov model? What kind of data is a Markov model used for?

1. In probability theory, a Markov model is a stochastic model used to model randomly changing systems.[1] It is assumed that future states depend only on the current state, not on the events that occurred before it (that is, it assumes the Markov property). Generally, this assumption enables reasoning and computation with the model that would otherwise be intractable. For this reason, in the fields of predictive modelling and probabilistic forecasting, it is desirable for a given model to exhibit the Markov property.
2. HMM is suitable to be used in application that dealing with recognizing something based on sequence of feature. The decision making is done based on sequence of information. Same as DTW or edit distance. However, for KNN, ANN or SVM, there are under same category which is one-off decision making. Meaning that the decision is made after one static features was given. Markov models are often used to model the probabilities of different states and the rates of transitions among them.

Reference: https://en.wikipedia.org/wiki/Markov_model

Q11 (5 Points) What is network initialization? How can it effect a neural network? Name three common approaches for network initialization and why one would choose one over another.

1. The initialization of weights will affect the performance of neural networks.
2. During forward propagation and back propagation, we can get the optimum values of weights to produce accurate outputs.
3. Three common approaches are zero initialization, random initialization and he-et-al initialization.

In zero initialization, we set all the weights to zero, all neurons of layers have same calculation. The whole deep net would be the same as a single neuron and the predictions would be nothing better than random.

w=np.zeros((layer_size[l],layer_size[l-1]))

Random initialization helps in breaking symmetry, which makes every neuron performs different computation.

w=np.random.randn(layer_size[l],layer_size[l-1])*0.01

In case of he-et-al initialization, the weights are initialized based on the size of the previous layer, which helps in attaining a global minimum of the cost function faster and more efficiently.

w=np.random.randn(layer_size[l],layer_size[l-1])*np.sqrt(2/layer_size[l-1])

Reference:

https://towardsdatascience.com/random-initialization-for-neural-networks-a-thing-of-the-past-bfcdd806bf9e

Q12 (5 Points) What is transfer learning? When would one use it? Give an example of transfer learning using neural networks.

1. Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.[1]
2. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks. This area of research bears some relation to the long history of psychological literature on transfer of learning, although formal ties between the two fields are limited.

   Reference: https://en.wikipedia.org/wiki/Transfer_learning

Q13 (5 Points) What is an autoencoder? What are they used for? Explain how an autoencoder works.

1. An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner.[1][2] The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. Recently, the autoencoder concept has become more widely used for learning generative models of data.[3][4] Some of the most powerful AI in the 2010s has involved sparse autoencoders stacked inside of deep neural networks
2. An autoencoder learns to compress data from the input layer into a short code, and then uncompressed that code into something that closely matches the original data. This forces the autoencoder to engage in dimensionality reduction, for example by learning how to ignore noise. Some architectures use stacked sparse autoencoder layers for image recognition. The first autoencoder might learn to encode easy features like corners, the second to analyze the first layer's output and then encode less local features like the tip of a nose, the third might encode a whole nose, etc., until the final autoencoder encodes the whole image into a code that matches (for example) the concept of "cat".[5] An alternative use is as a generative model: for example, if a system is manually fed the codes it has learned for "cat" and "flying", it may attempt to generate an image of a flying cat, even if it has never seen a flying cat before

   Reference: https://en.wikipedia.org/wiki/Autoencoder

Q14 (5 Points) What is a variational autoencoder? What are they used for? How do they differ from autoencoders?

Variational autoencoder models inherit autoencoder architecture, but make strong assumptions concerning the distribution of latent variables. They use variational approach for latent representation learning, which results in an additional loss component and specific training algorithm called *Stochastic Gradient Variational Bayes (SGVB)*.[3] It assumes that the data is generated by a directed graphical model $p(\mathbf{x}|\mathbf{z})$ and that the encoder is learning an approximation $q_\phi(\mathbf{z}|\mathbf{x})$ to the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ where ϕ and θ denote the parameters of the encoder (recognition model) and decoder (generative model) respectively. The objective of the variational autoencoder in this case has the following form:

$$\mathcal{L}(\phi, \theta, \mathbf{x}) = D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left(\log p_\theta(\mathbf{x}|\mathbf{z})\right)$$

Here, D_{KL} stands for the Kullback–Leibler divergence. The prior over the latent variables is usually set to be the centred isotropic multivariate Gaussian $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$; however, alternative configurations have also been recently considered, e.g.[11]

Reference: https://en.wikipedia.org/wiki/Autoencoder

Q15 (5 Points) What are deep generative models? What are they used for?

1. A Generative Model is a powerful way of learning any kind of data distribution using unsupervised learning and it has achieved tremendous success in just few years. All types of generative models aim at learning the true data distribution of the training set to generate new data points with some variations. But it is not always possible to learn the exact distribution of our data either implicitly or explicitly and so we try to model a distribution which is as similar as possible to the true data distribution. For this, we can leverage the power of neural networks to learn a function which can approximate the model distribution to the true distribution.

Reference: https://towardsdatascience.com/deep-generative-models-25ab2821afd3

Q16 (5 Points) What are multilayer perceptron? What are they used for?

1. A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training.[1][2] Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable
2. MLPs are useful in research for their ability to solve problems stochastically, which often allows approximate solutions for extremely complexproblems like fitness approximation.MLPs are universal function approximates as showed by Cybenko's theorem,[3] so they can be used to create mathematical models by regression analysis. As classification is a particular case of regression when the response variable is categorical, MLPs make good classifier algorithms.MLPs were a popular machine learning solution in the 1980s, finding applications in diverse fields such as speech recognition, image recognition, and machine translation software,[6] but thereafter faced strong competition from much simpler (and related[7]) support vector machines. Interest in backpropagation networks returned due to the successes of deep learning.

Reference: https://en.wikipedia.org/wiki/Multilayer_perceptron

Q17 (5 Points) In deep generative models what is meant by density estimation? What is meant by latent variable models?

1. Density estimation walks the line between unsupervised learning, feature engineering, and data modeling. Some popular density estimation techniques includes mixture models like Gaussian Mixtures, and neighbor-based approaches such as the kernel density estimate.
2. A latent variable model is a statistical model that relates a set of observable variables to a set of latent variables. It is assumed that the responses on the indicators or manifest variables are the result of an individual's position on the latent variables, and that the manifest variables have nothing in common after controlling for the latent variable (local independence).

Reference: http://scikit-learn.org/stable/modules/density.html

Q18 (5 Points) What are soft-max functions? What are they used for?

1. The softmax function squashes the outputs of each unit to 0 ~ 1, same as a sigmoid function. However, it divides each output to make the total sum of the outputs to be 1.
2. The softmax can be used for any number of classes. For example, in object recognition problems, there are hundreds of different possible objects.

Reference: https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions

Q19 (5 Points) What is a session in TensorFlow? Why aren't sessions required for scikit-learn?

1. A TensorFlow session allows to execute graphs or part of graphs. It allocates resources (could be on one or more machines) for them and holds the actual values of intermediate results and variables.
2. Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Pythonprogramming language. While Session can run following the graph.

Reference: https://danijar.com/what-is-a-tensorflow-session/

Q20 (5 Points) TensorFlow uses a dataflow graph. What is it? Why is it used?

1. TensorFlow uses a dataflow graph to represent the computation in terms of the dependencies between individual operations.
2. It leads to a low-level programming model in which we first define the dataflow graph, then create a TensorFlow session to run parts of the graph across a set of local and remote devices.

Reference: https://www.tensorflow.org/guide/graphs#why_dataflow_graphs