CS 1678/2078: Deep Learning

First Exam: Review Guide

February 23, 2024

General info: The exam will include all lectures up until March 4 (inclusive). The format will include multiple choice and true/false questions, short answers, and applications of algorithms on small toy problems. The exam will be on March 6, at our usual class time, in our classroom. You do not need to (and shouldn’t) bring calculators or other aids. You don’t need to memorize equations, except a few key ones (e.g. linear classifier, weight update, convolution, RNN recurrence), but should understand what they do at a high level (i.e. what AdaGrad does differently than gradient descent).

Other notes

Stride is how much the kernel moves in spatial pooling

Concepts and algorithms to review:

1. Intro
   1. How would you describe deep learning to a non-expert?

Teaching a computer to find patterns and predict based on those patters

* 1. What are some example deep learning (or machine learning) tasks and problems?

Predicting liklihood of a crash. Image classification

* 1. Why are these tasks challenging?

Sometimes there are patterns that show up in the data via coincidence but arent actually important

An example is speech where a lot of words have multiple meanings so it can be hard to know exactly which meaning we want

* 1. What does the overall DL/ML framework look like?

Y = wx + b

Overall it looks like

Input -> hidden layer(s) -> output

* 1. What do X and Y typically denote in a DL/ML framework?

X is predictors y is whats predicted

* 1. What is one example approach of predicting the label of a text or image (e.g. spam/not, object category)?

One example approach is linear classification each feature is added up and if the total is in range of one of our classifcations thats the prediction

* 1. How do we measure similarity of features?

We measure them based on distance from each other

* 1. What are some examples of vector norms?

examples are absolute value and adding up the vectors

Squaring the vectors adding them then square rooting the total

* 1. What is a linear classifier?

A linear function separating the classes

* 1. Why do support vector machines maximize a margin?

It reduces overfitting and avoids the effect of noise

* 1. What margin do they maximize?

The margin between 2 classes

* 1. Why might it be beneficial to allow some training examples to be misclassified, in soft-margin SVMs?

This prevents overfitting sometimes all features can point to one thing but the truth could be different

* 1. How do we validate ML/DL systems? What does the evaluation pipeline look like?

Split it into 3 sets train(which is the largest like 70-80%) validation and test The weights are moved during train while validation fine tunes

* 1. What are the train, validation, and test sets? How do we use them?

Train set is used put into the model in order to move the weights and improve prediction.

Validation is used to find the best hyperparameters it has both features and labels and is tested with multiple hyperparameters.

Test set is used to tell how good the algorithm is at predicting unused data.

* 1. What is a loss function?

A loss function is overall a way of finding how far the machines guess was from the truth

Examples include summing of squares or the mean of squares

* 1. What is optimization?

It is solving for the weights of a model starting randomly and slowly moving them to where they should be based on feedback of training

* 1. Why is generalization important?

Generalization is important because it helps with overfitting/underfitting if it's too general a model can be underfit while to generalized makes it overfit. Like generally apples are red but what about golden delicious apples there yellow

* 1. What is underfitting? What is overfitting?

Underfitting is when the modedl is to simple to represent all characteristics its high bias but low variance and has high training and test error.

Overfitting is when the model is to complex and is low bias high variance with low training error but high test error.

* 1. How do we measure complexity of a model, e.g. for polynomial curve fitting?

Complexity is the trade off between bias and variance. In terms of looks a graph that constantly switches from increasing to decreasing and vice versa is more complex than a straight one. In other words a higher polynomial is more complex

* 1. What is the effect of the amount of training data available on overfitting, for a model of fixed complexity?

More training data reduces overfitting since each data point has less influence on the prediction

* 1. What is regularization? What is its effect on overfitting?

Regularization is a way of penalizing large coefficient values by reducing the change in high value weights. It reduces overfitting.

1. Neural network basics
   1. How do we compute activations in a neural network?

Activations = sum of weights \* input + bias

* 1. What are some common non-linear activation functions?

Tanh Relu and sigmoid are common

* 1. What does it mean to train a neural network? What are the inputs and outputs? What are the steps?

Training a neural network means adjusting the weights until it can accurately predict something.

It is done by first making up random weights. Running data through the it aka the forward step. Then calculating the loss which is based on how different the prediction is from the true answer. Then Using the loss to adjust the weights in backprop

* 1. Why do deep networks require lots of data to train?

It takes a lot of data for several reasons first each step only moves weights slightly so having little data will cause it to underfit. You cant just increase the change in weights as that may cause overfitting. You also need enough data to test the accuracy of the model.

* 1. What are some common losses? How are they similar/different?

Hinge loss and cross entropy and triplet loss are common. Hinge loss tries to maximize the margin between the boundary and the data points while cross entropy doesn’t care as long as its correctly classified

* 1. How do we use a loss to train a deep network?

The loss is used to calculate the new weights for future iterations

* 1. What is a gradient, and how do we use it?

The gradient is a mapping of weights and loss from a 2d perspective it would be the vertex of a parabola but in n-dimensions based where n is the number of weights

* 1. What is gradient descent?

Gradient descent is moving the weights towards where loss would be lowest using loss to calculate

* 1. What is learning rate and why is it important?

Learning rate is a multiplier that decreases how much we move the weights and is generally set to around .001. Its important because having it to high would move it to the best weight for an individual data point vastly overfitting the model.

* 1. How do we learn the weights in a multi-layer neural network?

We generally start with random weights or for pretrained models premade weights. We learn what they are through training and either calculating what they should be or letting the gpu do it and just printing it.

* 1. How do we use the chain rule to compute gradients?

We use the chain rule to calculate the derivative of loss the derivative of the output layer and the derivative of weights seperately.

* 1. What does the full algorithm for training a neural network look like, given a fixed architecture, activation functions, and loss choice?

Activation(x, weights1) = z

Activation(z, weight2) = y

Mean((Y – y\_True)^2) = mse loss

Errors1 = y \* (1-y) \* (y-ytrue)

Erros2 = z \* (1-z)\*(weights2 \* errors1)

New weights = old\_weight – learning\_rate\*(error1 \*z) for 2nd layer or

old\_weight – learning\_rate\*(error2 \*x) for first layer

1. Training part 2
   1. What are some ways of preprocessing data for use in a neural network?

Centering the data around 0 and normalizing it are the most common ways. Other ways include sqrt or any root trying to make it linear

* 1. What are some possible ways to initialize weights in a neural network?

First is randomly

Second is randomly over the sqrt of number of rows aka xavier

* 1. What are the advantages/disadvantages of some common non-linear activation functions?

Sigmoid which squashes it in range 0-1 but it kills the gradients its not zero centered and it take a lot of computing power.

Tanh is zero centered squashing between –1 and 1 but has the same problem with gradients

Relu does not saturate and is computationally efficient and converges quickly but is not zero centered and has problems with negative losses

Leaky relu is the same as relu but can have negatives they are just multiplied by .01 same advantages but doesnt have the negative loss problem and wont die.

Elu which has all the benefits of relu but with closer to zero mean outputs and negative saturation compared to leaky making it better with noise. But its more computationally expensive

* 1. What is batch normalization, and how do we give the network flexibility to decide how much normalization to apply?

Its normalization but only using data from a single batch ie zero centering it and squashing how far points are from each other

* 1. How do we use learning curves (on the training and validation set) to gauge how well training is proceeding?

We use the learning curve to tell us if we are only improving the models ability to predict new data. Or if we are just overfitting it on the data thats already there.

* 1. How can we prevent overfitting in a neural network?

Using a validation set

Regularization ie smoothing out the curve the higher the reularization variable the smoother

Data augmentation

Transfer learning

* 1. What is dropout and why is it useful?

Dropout is randomly turning off some neurons. It prevents overfitting and allows individual neurons to bave more influence training them better

* 1. What is data augmentation and why is it useful?

Data augmentation is changing the data like flipping an image cropping resizing etc. It helps generalize the data thus reducing overfitting. And improving its ability to make predictions based on smaller things

* 1. What is transfer learning and why is it useful?

Transfer learning is using a model changing the classification layer while keeping the weights for other layers. Then either training just the classification layer or multiple layers with a small learning rate. Its useful when you only have a small dataset but also have a similar enough dataset.

* 1. What are some of the ways to do transfer learning?

Some ways of doing it are training the whole thing with new data after changing the classification layer. Only training the classification layer. Or only training the classification layer and a few other layers for fine tuning.

* 1. What is the hardware needed for deep learning?

CPU GPU or TPU

* 1. What is convergence vs divergence?

Convergence is when results move towards the same place while convergence is when they move towards different places.

* 1. Why are local minima problematic/not? Why are saddle points problematic?

Local minima are problematic because at them a small change to any weight is worse than at the minima so its stuck but that minima is not the actual optimal place. Saddle points are similar and where the slope is zero ie any small change in the weight changes nothing. This is a problem because it doesnt know where to go so it gets stuck.

* 1. What are some desirable and undesirable properties of the loss surface?

Desirable properties are convex if it continuously curves up then then there are no local minima. It desires a single global optimal point not local ones

* 1. What is a condition number and why is it important?

It is the change in output relative to change in input

Condition number effects the speed of convergence a smaller condition number the slower it is.

* 1. What is mini-batch gradient descent?

Its gradient descent performed on very small batches like 5-10

* 1. What are some problems with gradient descent?

It gets stuck at local minima or saddle points

* 1. What variants of gradient descent are there, and how are their weight updates different?

There is momentum which as the weight moves one way it accelerates in that direction ie moves more

There is adagrad which squares it and makes it faster in “flat” directions and slower on steep ones.

rmsprop which is loke adagrad but with decay rate

Adam which decays the number from ada grad slowing it down also has momentum

* 1. What is learning rate decay?

Learning rate decay is essentially as the program runs reducing learning rate, so it slows down

* 1. What are some tips for choosing the values of important hyperparameters? How do we decide what is a good setting of these hyperparameters?

Start with no hyperparameters

Overfit a small sample

Find an lr that reduces loss

Change hyperparemeters

Refine the and keep training

Look at loss curves

* 1. What is a computation graph, and why is it useful?

They are a way of visualising a neural net while keeping it simple to follow

* 1. What role do different operators (e.g. plus, max) play when backpropagating gradients? What does each gate do in terms of propagating the gradient?

In backprop they do for plus it acts as copy in forward and vice versa for multiplier it multiplies the backprop by the number of the other path. For max the backprop follows the max while the other path gets a 0

* 1. What are Jacobians and how are they related to gradients?

Jacobians are a matrix of partial derivatives of the gradients its gradient split into multiple parts.

1. Convolutional neural networks (CNNs)
   1. What are scanning networks? How are they different than vanilla neural networks? What are their advantages over vanilla networks?

Scanning networks are better at finding an off center image. They look at different parts of an image and try to find the object in it. In them each neuron is connected to a subset of the neurons on the previous layer. There advantage is location is less important as long as the pattern is found in a subset.

* 1. What is the role of filters? What kind of patterns can they capture? How do we decide what the filters should capture?

Filters compute in the local neighborhood of a pixel in an image. They enhance the image by removing noise and making the important part bigger. Help extract information like texture. And detect patterns

We do it by averaging the weight of each pixel. We decide based on what we want

* 1. What operations do we perform (i.e. what layers do we have) in a CNN?

We perform convolution non-linearity and spatial pooling

Correlation weighs the nearby value equaly while convolution increases the weight of closer values.

* 1. How do these operations affect the size (width, height, depth) of each volume that is output by one layer and input into the next layer?

Convolution applys filter weights with one map per filter separating an image into different part

Non linearity is just the activation function per element it helps reduce saturation which is zero gradient points.

And spatial pooling is summing getting the mean of or maxing regions which makes feature maps smaller by merging them.

* 1. How do CNNs integrate feature learning and classification?

CNN’s seperate features into levels and tries to learn deteect each feature seperately before bringing everything back together in order to classify it.

* 1. What is AlexNet? What layers does it have?

It has a convolution layer max layer pool layer normalization layer, repeat aboce then 3 convolution layers a max latyer a pool layer then 3 final classifaction layers

* 1. What are some other popular CNN architectures? How are they similar? How are they different?

Vggnet which has 16-19 layers as opposed to alexnets 8 and vggnet stacks 3 3x3 stride 1 conv layers on top of each other

Also googlenet which has 22 layers and no final classification layer it essential stacks tiny networks on top of each other to create a larger network

Resnet which stacks residual blocks contain 2 3x3 conv layers but have input send through the resnet block and also send a copy that skips the block to think about it imagine x is passed through f(x) instead of the next block taking f(x) as input it takes x+f(x)

* 1. What are the advantages of using only 3x3 filters in VGG?

Using multiple 3x3 as opposed to one bigger layer means that its deeper and has more non linearities. Also fewer parameters in each layer

* 1. How does GoogLeNet deal with vanishing gradients?

It uses auxiliary classifiers which pushes gradients to lower layers

* 1. Why do we need to add residual connections in ResNets?

It solves a problem with depth where instead of looking deep layers only try to fit a residual mapping with no understanding of its origin

* 1. What are three ways to visualize/understand what a CNN has learned?

First is a heatmap of where an object is.

Second is an occlusionmap which outlines the shape of what your looking for.

Finally you can try seeing which images maximize the class score to find what its looking for,

* 1. Why do we say that CNNs are easy to fool? What are some examples to illustrate this?

We say this because distortions to the image can change what it sees. Also since its looking for small patterns and not at the totality of an image if you include the pattern and texture of an objecct but not the object itself in a picture it gets fooled.

* 1. What are some flavors of object recognition/detection, i.e. some sub-tasks and problems?

Semantic segmentation ie seperating all the objects in an image. Classification and localization finding where and what something is. Detection of an object. And segmentation seperating objects

* 1. How can we approach object localization as a regression task?

From the network output we get the class scores and the coordinates of the object including weights and height

* 1. What are region proposals and why are they useful?

They are finding blobby shapes in an image and assume that these blobs likely contain objects which helps limit whats being checked reducing noise

* 1. What is R-CNN and how does it work?

Rcnn uses region proposals and passes those regions into their own copy of the network.

* 1. Why is R-CNN slow and how does Fast R-CNN speed up the detection process?

It needs to run the network multiple times and since it transforms the image into the right size each run isnt any faster. Fast R-cnn fixes this by creating a feature map first then separating into regions of interest before pooling and entering final classification

1. Recurrent neural networks (RNNs)

An RNN is a model that loops the output back into itself be it x number of times or as much as possible within a time frame

* 1. How is classifying sequences different than classifying images?

Classifying an image takes one input (the image) Classifying a sequence of has multiple inputs that need to be combined in some way

* 1. What is a language model?

A language model is a system that given a sequence of words predicts what comes next

* 1. What are the advantages of a fixed-window neural network model over an n-gram model?

Ngram model predicts the next word based on the preceding n-1 words

A fixed window removes the sparsity problem which comes in 2 forms. First form is if lets say a sentance like “I want to eat \_\_\_” occurs if I want to predict the blank is apples it will never happen if “I want to eat apples” never occurs in the training data. The second form is when the “I want to eat” never occurs we cant calculate any prediction.

Fixed window takes each word individually so the sequence doesn't matter, but it's limited in words and can never be big enough. Also the fixed length is inherently bad.

* 1. What are the advantages of a recurrent network over a fixed-window network?

RNN processes any length without increasing model size. Same weights are applied every timestep so there is symmetry in how its processed. In theory can use info from earlier steps though in practice thats hard. Its weakness is its very slow run time. Also recent sequences have more sway then ones that are several steps back

* 1. What are some applications of recurrent networks? What are inputs/outputs in those cases?

Predictive typing which takes what your typing as input and outputs new text. Speech recognition which takes someone talking and tells who it was. Image captioning which takes an image as input and outputs text explaining it. Translation which is what it sounds like

* 1. Why do we say that an RNN is recurrent?

Because it loops on itself passing input back into part of itself multiple times.

* 1. Why is training RNNs challenging?

Vanishing gradients are really bad in Rnn’s which cause near effects to have large influence while old ones have little sway. Other problem is the opposite where gradient explodes and step size becomes to big that it over does it.

* 1. How are GRUs/LSTMs different than standard RNNs?

Instead of computing the next time step directly it computes one or more gates from the output first

* 1. What are the roles of the gates in a GRU? What about the gates in an LSTM?

In GRU The update gate controls what parts of the hidden state are updated and what parts are preserved. The reset gate controls what part of the previous state is used to compute the next one

LSTM has forget gate which says what is kept from previous state and what is forgotten. The input gate which contains parts of what will be written and output which controls what parts are output to hidden.

GRU is faster and there is no conclusive evidence saying one is better than the other

* 1. How does beam search help achieve a more optimal solution?

Beam search decoding is when on each step of a decoder it keeps track of the k most probable partial translations. Its faster than exaustive which gets everything instead it gets the top k next words and scores them The deepest word with the highest score get extended until the sequence is complete.

* 1. How do we use attention for sequence-to-sequence tasks, e.g. machine translation? What does attention capture?

Attention is used to keep track of a core idea and focus primarily on that essentially breaking down the problem. It uses a encoder hidden state and passes it to a decoder hidden state

* 1. What are some ways to use an RNN for image captioning? What are some ways to use an RNN for video captioning?

It can add attention to image captioning making it more detailed also it isnt bound to a set output size. Same for video captioning.

* 1. How does attention work for image captioning?

Attention helps by allowing the network to breakdown the problem. It can focus on the texture to find a material and then the shape to find the object like a wooden house. Then it could move attention to location and so on.

Practical skills:

1. Write down some possible features (X) that could be used in the non-DL ML setting (i.e. where a network is not computing features)
2. Write down some possible (hypothetical) weights for a linear model, given X and Y
3. Compute activations for some node in a neural network, given inputs, architecture/connections and choice of non-linear activation functions
4. Use the equations for some loss function to calculate loss values, from scores for different samples, and the ground-truth labels on the samples
5. Compute loss when regularization (with L1, L2) is included
6. Compute gradient numerically (using a “black box” to output loss given weight, as in the example shown in class)
7. Compute a weight update (using gradient descent)
8. Match pseudocode for gradient descent, gradient descent + momentum, AdaGrad, and RMSProp to the right optimizer name (e.g. AdaGrad)
9. Compute a few iterations of gradient descent by hand, as in HW1 but with a simpler architecture
10. Show the terms required to compute dE/dw for some fixed weight, and show how each term can be calculated analytically (i.e. with calculus) for simple examples
11. Show how to construct a computation graph for a simple problem
12. Using a given computation graph, compute the values of the gradients for the weights
13. Match “gates” in a neural network to what they do (e.g. “swap multiplier”)
14. Compute convolution outputs or suggest possible filters that can be used to look for a pattern (e.g. vertical edge)
15. Match the RNN algorithm (e.g. GRU) to its equations, and understand the role of each variable (e.g. each gate)
16. Write the RNN recurrence formula