APS360. Term Test Study Topics

This is a list of lecture/lab topics compiled by your instructor to help you study for the term test. Note that the list may not be exhaustive.

In particular, the list of topics doesn't cover the skills that we develop throughout the course through the labs and project. For example, interpreting the training curve is a skill that we practiced throughout the course. As another example, you should be able to predict the output shape of PyTorch layers we discussed in class, like `nn.Conv2d` and `nn.RNN`.

Week 1

Machine Learning Overview

- artificial intelligence, machine learning, deep learning
- type of problems where machine learning is successful
- supervised learning (regression, classification), unsupervised learning, reinforcement learning
- issues with deep learning: interpretability, fairness

Biological motivations

- biological neuron: dendrite, axon, synapse, cell body, the direction that information travel
- grandmother cell
- distributed encoding

Artificial Neuron

- artificial neuron: activation, weight, bias
- input layer, hidden layer(s), output layer
- fully-connected layer
- feed-forward neural network
- multi-layer perceptron (MLP)
- steps to training a neural network
- loss function (binary cross-entropy)
- optimizer (stochastic gradient descent)
- overfitting, training set, test set, ground-truth, prediction
- ethics and interpretability

Week 2

Artificial Neural Networks

- ReLU, sigmoid, tanh (know the behaviour)
- LeakyRelu (from Lab1)
- neural network architecture
- counting number of layers in a MLP
- parameters, weights, biases
- counting number of parameters / weights / biases
- credit assignment problem
- standard training/test set splits

Optimization and Hyperparameter

- batch size (effect of large / small batch size)
- iteration, epoch
- the use of PyTorch's `DataLoader` to batch data for us
- learning rate (effect of large / small learning rate)
- training curve (what's on each axis)
- training loss / accuracy / error
- interpreting training curves
- validation set
- validation loss / accuracy / error
- hyperparameter search
- examples of hyperparameters beyond learning rate and batch size
- model selection (based on validation)
- Checkpointing (from Lab2)

Week 3

Multi-Class Classification

- the use of PyTorch fully-connected `nn.Linear` layer
  - the meaning of `in_features` and `out_features`
  - changing the number of layers in a neural network
- the use of PyTorch activation functions `F.relu` and `F.softmax`
- the difference between PyTorch `CrossEntropyLoss` and `BCEWithLogitsLoss`
- the general training loop code:

```
out = model(imgs)              # forward pass
loss = criterion(out, labels) # compute the total loss
loss.backward()                # backward pass (compute parameter updates)
optimizer.step()               # make the updates for each parameter
optimizer.zero_grad()          # a clean up step for PyTorch
```

- interpreting training curves to identify overfitting
- being able to understand relevant PyTorch layers we discussed when the API documentation is given
- forward pass, backwards pass
- predict the output shape of a PyTorch layer

Troubleshooting

- the idea of overfitting on a small data set

Week 4

Convolutions

- reasons why fully-connected layers can be problematic
- locally-connected layers, weight-sharing
- convolutional kernels / filters and their sizes
- input and output channels / feature maps and their sizes
- convolutional arithmetics (computing forward pass of a convolution layer given the input and kenrel)
- PyTorch "NCHW" format
- PyTorch convolution layers `nn.Conv2D`, its parameters, and computing the output shape of a convolution
- zero-padding (the reason for their use, selecting padding size)
- max-pooling, and the `nn.MaxPool2D` layer

- average-pooling
- strided convolutions
- examples of the types of features computable using convolutions
- forward pass arithmetics of pooling and padded/strided convolutions

Architectures

- What is new in each architectures:
  - LeNet
  - AlexNet
  - GoogLeNet: repeated modules, multiple convolution kernels
  - ResNet: residual connections
- Global average pooling
- Fully-convolutional architectures

Week 5

Regularization

- why we don't talk about underfitting as much
- how to detect overfitting
- weight sharing
- data normalization
- data augmentation
- weight decay
- early stopping
- dropout
- marking whether a PyTorch model is in training mode or evaluation mode

Convolution Transpose

- upsampling
- convolution transpose arithmetics
  - effect of padding
  - effect of strides
  - effect of output_padding
- we did not cover counting the number of parameters of a transpose convolution layer

Autoencoders

- the idea of an embedding space, and different features/representation of the input
- mean-squared error loss (MSELoss)
- how to train an autoencoder
- how to use the decoder to generate new images
- the effect of interpolation in the embedding space vs interpolating in the pixel space
- possible uses of an autoencoder
- effect of size of latent space
- the general idea of an "embedding space"

Word Embeddings

- one-hot encodings
- word2vec idea: meaning of a word depends on its context

- general architecture of a word2vec model (input/output of the encoder/decoder)
- distances measures in embedding spaces (euclidean distance, cosine similarity)
  - intuition about how “close” points are in these measures
- effect of word embedding size
- how to use word embeddings as input
- limitations to using word embeddings (e.g. bias)

Bias

- learning the bias in the data set
- the type of bias in GloVe embeddings
- the type of bias can exist in any classifier

Week 6

Recurrent Neural Networks

- classification for sequences (e.g. sentiment analysis)
- advantages and disadvantages of modeling text as a seqence of words vs characters
- how to set the `input_size`, `hidden_size` and `output_size` of a recurrent network
- the difference between `nn.RNN`, `nn.LSTM` and `nn.GRU`
- Note: You are not expected to know about the computations that happen when a hidden state is updated. You are expected to know how to use a Recurrent Neural Network in PyTorch
- RNN batching, the use of padding

Week 8

Recurrent Neural Networks for Text Generation

- architecture; how to set the `input_size`, `hidden_size` and `output_size` of the RNN
- training text generation model
  - by learning to predict the next character
  - teacher forcing
  - how the loss is computed
  - why `<BOS>` and `<EOS>` tokens are useful
  - You don’t need to memorize the code, but should be able to explain, step-by-step, how to train a generative RNN
- generating text using the model
  - how to generate one token at a time
  - why we need to *sample* the next character
  - the temperature setting: effect of a high vs low temperature
  - You don’t need to memorize the code, but should be able to explain, step-by-step, how to generate text using an RNN

Week 9

Generative Adversarial Networks

- generative model (unsupervised learning) vs discriminative model (supervised learning)
- why autoencoders produce blurry images (MSELoss)
- GAN architecture
  - the generator, its purpose, and input/output
  - the discriminator, its purpose, and input/output
- loss functions (objectives) for the generator and discriminator

- why plotting the loss over time will not be as helpful as in supervised learning
  - e.g. discriminator loss will increase when the generator gets better, and vice versa
  - difficult to tell when to stop training
- why plotting the discriminator score for real & fake images are helpful
- effect of a discriminator that is too powerful
- mode collapse
- LeakyReLU activation
- Batch Normalization

Adversarial Examples

- the goal of an adversarial attack
- targeted vs non-targeted adversarial attack
- white-box vs black-box adversarial attack
- optimize a noise tensor in PyTorch
  - what loss function to use

Week 10

Reinforcement Learning

- examples of reinforcement learning problems
- environment, agent, state, action, reward
- spare reward vs dense reward
- how to set up an RL problem (i.e. choose the appropriate state, action, and rewards)
- the reward hypothesis: you can train a model to solve any RL tasks (achieve any goal you want), by having the model learn to maximize some total reward.
- episode
- return (discounted return)
- components of an agent: policy, value function, model
- deterministic vs stocastic policy
- value function: maps state to expected return
- Q-function: maps (state, action) to expected return
- taxonomy of RL agents:
  - policy based, value based, and actor-critic agents
  - model free vs model based agents
- neural network input/output for a value-based and policy-based agent, how to interpret the output of the neural network
- why we train based on returns rather than rewards
- epsilon-greedy policy
- exploration vs expoitation

Week 11

Ethics and Fairness

- major issues in AI ethics, including:
  - misuses
  - data collection and privacy
  - adversarial examples
  - reproducibility
  - fairness
- equality vs equity
- disparate treatment vs disparate impact

- ways of measuring fairness
  - demographic parity
  - equalized odds (accuracy parity)
  - individual fairness
- why different measures of fairness are inconsistent and sometimes contradictory
- ways to make models more fair:
  - pre-processing
  - add a fairness regularizer
  - post-processing