Training Neural Networks

THE UNIVERSITY OF CHICAGO

Crescat scientia vita excolatur

1890

1. Backpropagation
2. Hyperparameters
3. Problems With Training NN
4. Ways To Improve Training

Backpropagation

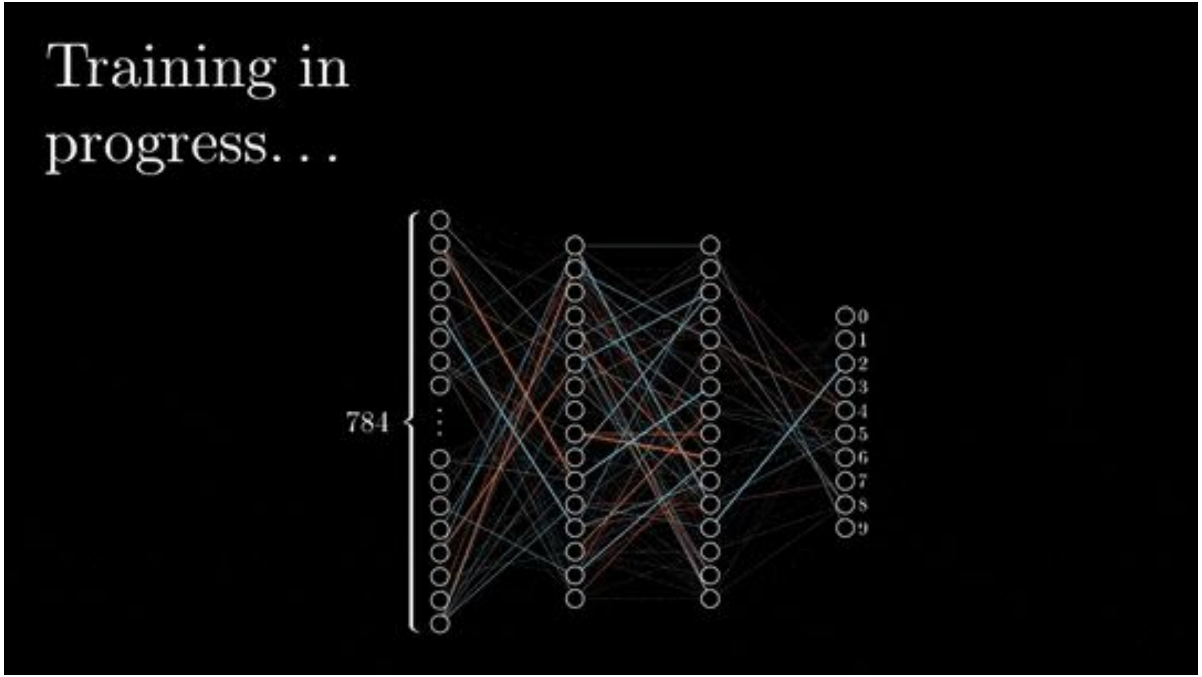
Weight Updates

- We want the neural network to learn how to correctly predict the output given the input
- To do this, we need to update the weights of each neuron in each layer

Backpropagation (BP) Algorithm

1. We feed a training instance into the NN
2. Measure the output error of our prediction with respect to the label
3. Compute how much each neuron in the previous hidden layer contributed to each output neuron’s error
4. Repeat step 3 until we reach the input layer
5. Perform gradient descent on all the weights using the errors

Backpropagation Algorithm


Training in progress...

Hyperparameters

Number of Hidden Layers

- Deeper networks more efficiently
  - model complex functions than shallow networks
  - generalize to new datasets
- It’s a good idea to gradually increase hidden layers until the model starts overfitting

Number of Neurons Per Hidden Layer

- One common practice is to form a funnel: there are fewer and fewer neurons at each layer
- Or you can just have the same number of neurons at each layer
- You get more bang out of your buck by increasing number of layers than you do by increasing number of neurons

Problems With Training NNs

Vanishing / Exploding Gradients

- Vanishing gradient problem: during BP, the gradients of updates can get smaller and smaller. Update weights may stop changing and training will not converge.
- Exploding gradient problem: during BP, gradients may grow bigger and bigger and the algorithm diverges

Solutions

- Xavier and He initialization
  - Initialize the weights according to a normal or uniform distribution that depends on the input and output sizes
- Use a non-saturating activation function

Batch Normalization

- Just before the activation function, we can perform batch normalization
- This involves zero-centering the mean, normalizing the input, then scaling and shifting the results using two new parameters per layer
- The mean and standard deviation is evaluated on the current mini-batch

Ways To Improve Training

Transfer Learning

- Instead of training a large NN from scratch, you can find an existing NN that accomplishes a similar task and then reuse the lower layers of that NN
- It is generally a good idea to freeze the transferred weights because it makes the NN easier to train

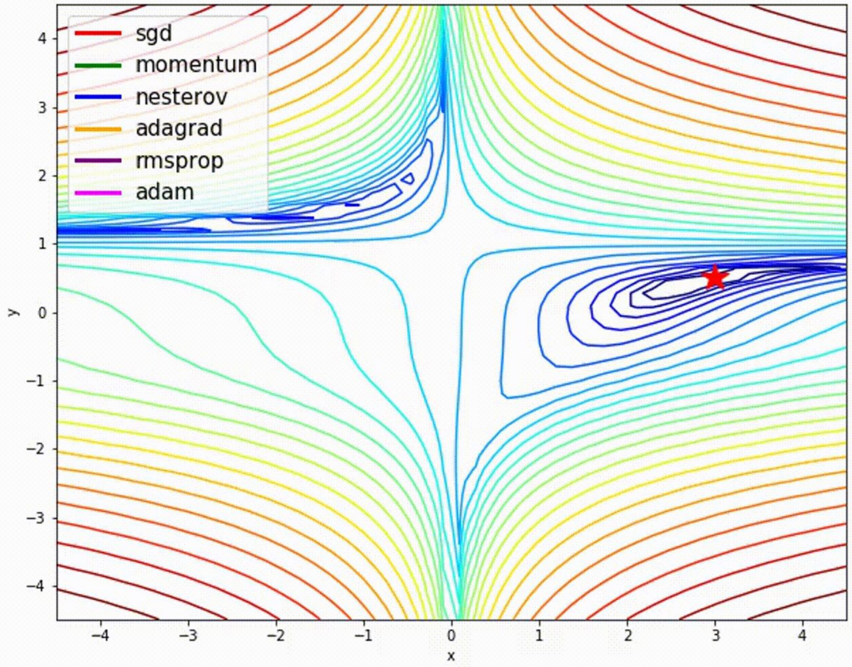
Pre-training

- When you don’t have a lot of data, you can pre-train your model on unlabeled data using an unsupervised NN, and then transfer those weights
- You can also pre-train your model on a different task where data is easily available, and then transfer some weights

Faster Optimization

- Instead of using simple gradient descent, we can use different optimizers to speed up training
- The Adam optimizer generally works better than most other optimizers, although you may want to experiment with the other optimizers
- You can also trying learning rate scheduling, where the learning rate changes over time

Faster Optimizers



Regularization

- Early stopping
- L1, L2 regularization
- Dropout
  - At every training step, each neuron has a probability of being unused
- Data augmentation
  - Create more data from the data we already have

Questions to Answer

1. How many neurons do you need in the output layer to classify whether an image is a dog or a cat? What about trying to what digit an image is (digits 0 to 9)?
2. Is it okay to initialize all the weights to the same value as long as that value is selected randomly using He initialization?
3. If your neural network is overfitting, how would you tweak hyperparameters to reduce overfitting?
4. Why would you use a logistic activation function when performing classification tasks?

Questions to Answer

1. When transfer learning, what layers do you NOT want to transfer over?
2. When transfer learning, why do you think freezing the transferred layers make it easier for the neural network to train?
3. Does transfer learning work when you are using layers trained on a task that is very different from the task that you are trying to solve?
4. Does dropout slow down training? Does it slow down inference?