Module 3: Feedforward Neural Networks

Lesson 1: Feedforward Neural Networks

単語

- congruent: ≅, if one thing is congruent with another thing, they are similar or fit together well.

内容

- The mode of action of feedforward neural networks.
- The mathematical equations of the hidden layers, a building block that makes neural networks special among machine learning models.

Feedforward Neural Networks

- A Feedforward Neural Network defines a mapping from input x to output y as: $y = f(x;\theta)$.
  - θ: learned parameters.
  - The concept of learned parameters is important, as we do not start with the correct form of the function f, which maps the inputs to outputs directly.
  - This means that neural networks can be thought of as function approximators.
- An N layer FNN is represented as the function composition:

$$f(x;\theta) = f^{(N)}\left(f^{(N-1)}\left(\ldots f^{(2)}\left(f^{(1)}(x)\right)\right)\right).$$

  - N is a large number.
  - This layering led to the name deep learning for the field describing these sequences of functions.
- Feedforward: This is because information flows from the input x through some intermediate steps, all the way to the output y without any feedback connections.
  - The terms are used in the same way when describing control for self-driving cars.
- Functions to estimate
  - Object Classification: Image -> Label.
  - Object Detection: Image -> Label + Location.
  - Depth Estimation: Image -> Depth for every pixel.
  - Semantic Segmentation: Image -> Label for every pixel.
- This flexibility to represent hard-to-model processes is what makes neural networks so popular.

Mode of Action of Neural Networks

- Training: Given neural network examples of $f^*(x)$ for a wide variation of the input x.
  - Then optimize its parameters θ to force $f(x;\theta) \cong f^*(x)$.
- Paris of x and $f^*(x)$ are called training data.
- Only output is specified by training data!
  - Network is free to do anything with its hidden layers.

Hidden Units

$h_n = g(W^T h_{n-1} + b)$

- Activation function g.
  - element-wise non-linear function.
- Parameters θ are the weights and biases of all the layers of the network.
- Most of the time, g does not contain parameters to be learned by the network.

The Rectified Linear Unit: ReLU

- The ReLU hidden unit is currently the default choice of activation function for Feedforward Neural Networks: $g(z) = max(0, z)$.
  - Since they are very similar to linear units, they're quite easy to optimize.

他のActivation Functions
- Sigmoid non-linearity.
- the hyperbolic tangent non-linearity. (Tanh)
- the generalization of ReLU: the mahout non-linearity.

Lesson 2: Output Layers and Loss Functions

内容
- The general process of designing machine learning algorithm, and extend it to the design of neural networks.
- Different types of neural networks loss functions that can be used depending on the type of task at hand.

Machine Learning Algorithm Design
- Generally, supervised machine learning models including neural networks have two modes of operation, inference and training.
  - Inference is usually the operation we used when we deploy the machine learning algorithms in the real world.
- For self-driving, the training data often takes the form of human annotated images which take a long time to produce.
- The Optimization procedure takes in the output of the loss function and provides a new set of parameters θ that provide a lower value for that loss function.
- The major difference between the design of traditional machine learning algorithms and the design of artificial neural networks, is that the neural network only interacts with the loss function via the output layer.
  - Therefore, it is quite reasonable that the output layer and the loss function are designed together depending on the task at hand.

Tasks: Classification and Regression
- Classification: Given input x, map it to one of k classes or categories.
  - Image classification, semantic segmentation.
- Regression: Given input x, map it to a real number.
  - Depth prediction, bounding box estimation.

Classification: Softmax Output Layers（大事）
- Softmax output layers are most often used as the output of a classifier, to represent the probability distribution over K different classes.
- The Softmax output layer is comprised of:
  - A linear transformation: $z = W^T h + b$.
  - Followed by the Softmax function: $Softmax(z_i) = \frac{exp(z_i)}{\Sigma_j exp(z_j)}$.

Classification: Cross-Entropy Loss Function（大事）
- By considering the output of the softmax output layer as a probability distribution, the Cross Entropy Loss function is derived using maximum likelihood as:
  $$L(\theta) = -log\left(Softmax(z_i)\right) = -z_i + log\sum_j exp(z_j).$$
- The Cross-Entropy Loss has two terms to control how close the output of the network is to the true probability.
- z_i is usually called the class logit, which comes from the field of logistic regression.

- When minimizing this loss function, the negative of the class logit z_i（つまり$-z_i$）encourages the network to output a large value for the probability of the correct class.
- The second term on the other hand, encourages the output of the affine transformation（つまりz）to be small.
- Note how the loss function heavily penalizes erroneous predictions even when the difference in output is only one.
    - これは$-z_i + log \sum_j exp(z_j)$からすぐ分かれるでしょう。分母はlog、分子はlinearだから。正しさも間違いも全部exaggerateされてる。
    - This difference accelerates the learning process and rapidly steers network outputs to the true values during training.

Regression: Linear Output Layers
- Linear Output Units are based only on an affine transformation with no non-linearity: $z = W^T h + b$.
- Linear Output Units are usually used with the Mean Squared Error loss function to model the mean of a probability distribution: $L(\theta) = \sum_i \left(z_i - f^*(x_i)\right)^2$.

Lesson 3: Neural Network Training with Gradient Descent

内容
- How to train a neural network using the iterative optimization algorithm: gradient descent.
- How to initialize parameters at the start of the optimization process.

Neural Network Loss Functions
- Thousands of training example pairs $[x, f^*(x)]$.
- The Loss function computed over all N training examples is termed the Training Loss and can be written as: $J(\theta) = \frac{1}{N}\sum_{i=1}^{N} L[f(x_i, \theta), f^*(x_i)]$.
- The gradient of the training loss with respect to the parameters θ can be written as:
$$\nabla_\theta J(\theta) = \nabla_\theta \left[\frac{1}{N}\sum_{i=1}^{N} L[f(x_i, \theta), f^*(x_i)]\right] = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta L[f(x_i, \theta), f^*(x_i)].$$
    - using the fact that the gradient and the sum are linear operators.

Batch Gradient Descent
- Batch Gradient Descent is an iterative first order optimization procedure.
- Batch Gradient Descent Algorithm:
    - Initialize parameters θ.
    - While Stopping Condition is Not Met
        - Compute gradient of loss function over all training examples:
        $$\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta L[f(x_i, \theta), f^*(x_i)].$$
        - Update parameters according to: $\theta \leftarrow \theta - \epsilon \nabla_\theta J(\theta)$.
- How do we initialize the parameters θ and how do we decide when to actually stop the algorithm?
    - The answer to both of these questions is still highly based on heuristics that work well in practice.

Parameter Initialization and Stopping Conditions（大事）

- Parameter Initialization
  - Weights: initialized by randomly sampling from a standard normal distribution.
  - Biases: initialized to 0.
  - Other heuristics exist.
- Stopping Conditions
  - Number of iterations.
  - Change in θ values: Stop if $\theta_{new} - \theta_{old} <$ Threshold.
  - Change in $J(\theta)$ value: Stop if $J(\theta_{new}) - J(\theta_{old}) <$ Threshold.
    - JはN training samplesのLの平均値。

Batch Gradient Descent

- Backpropagation used to compute $\nabla_\theta J(\theta)$ is very expensive to compute over the whole training dataset.
  - 原因：Backpropagation involves computing the output of the network for the example on which we would like to evaluate the gradient.
- $\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta L[f(x_i, \theta), f^*(x_i)]$ is a mean!
- Standard error of the mean estimated from N samples is $\frac{\sigma}{\sqrt{N}}$, where σ is the standard deviation of the value of the samples.
- Using all samples to estimate the gradient results is less than linear return in accuracy of this estimate.
- Use a small subsample (Minibatch) of the training data to estimate the gradient!

Stochastic (Minibatch) Gradient Descent

- Stochastic (Minibatch) Gradient Descent Algorithm
  - Initialize parameters θ.
  - While Stopping Condition is False
    - Sample a preset number N' of examples (minibatch) from the training data.
      - new parameter N'.
    - Compute gradient of Loss Function over these N' training examples:

$$\nabla_\theta J(\theta) = \frac{1}{N'}\sum_{i=1}^{N'} \nabla_\theta L[f(x_i, \theta), f^*(x_i)].$$

    - Update parameters according to: $\theta \leftarrow \theta - \epsilon \nabla_\theta J(\theta)$.

What Minibatch Size to Use? これは覚えてる

「Deep Learning」p270~275: 8.1.3 Batch and Minibatch Algorithmsに全部書いてある!

- GPUs work better with powers of 2 batch sizes.
- Large batch sizes > 256:
  - Hardware underutilized with very small batch sizes.
  - More accurate estimate of the gradient, but with less than linear returns.
- Small batch size < 64:
  - Small batches can offer a regularizing effect. The best generalization error is often achieved with batch size of 1.
    - 「Deep Learning」p272によると、Small batches can offer a regularizing effect, perhaps due to the noise they add to the learning process. Generalization error is often best for a batch size of 1. Training with such a small batch size might require a small learning rate to maintain stability because of the high variance in the estimate of the gradient. The total runtime can be very high as a result of the need to make more steps, both because of the reduced learning rate and because it takes more steps to observe the entire training set.

    - Small batch sizes allow for faster convergence, as the algorithm can compute the parameter updates rapidly.
- Always make sure dataset is shuffled before sampling minibatch.
- As a result of these trade-offs, typical power of two minibatch sizes range from 32 to 256, with smaller sizes sometimes being attempted for large models or to improve generalization.

SGD Variations
- Many variations of SGD exist
    - Momentum SGD, Nestrove Momentum SGD.
    - Ada-Grad, RMS-Prop.
    - ADAM (Adaptive Moment Estimation).
- Which one to use?
    - ADAM: Implemented in most deep neural network libraries, fairly robust to the choice of the learning rate and other hyperparameters.

Lesson 4: Data Splits and Neural Network Performance Evaluation

内容
- How to split a dataset for an unbiased estimate of performance. 意味はまだ分かっていない。そうだ！ training set, validation set, test setの話でしょう！
    - unbiased estimate,つまりtest setがtraining setとindependent, identically distributedという意味でしょう。
- How to improve the performance of neural network by observing the difference in performance on the various data splits.

Data Splits
- Given a large enough neural network, we are almost guaranteed to get a very low training loss.
    - This is due to the very large number of parameters in a typical deep neural network allowing it to memorize the training data to a large extent given a large enough number of training iterations.
- Training Split: used to minimize the Loss Function.
- Validation Split: used to choose best hyperparameters, such as the learning rate, number of layers, the number of units per layer, and the activation function type.
    - Hyperparameters are those parameters that either modify the network structure or affect the training process. つまりhyperparametersこそネットワークを決めるでしょう！
- Test Split: the neural network never observes this set. The developer never uses this set in the design process.
    - used to get an unbiased estimate of performance.
- データのサイズによって各splitのpercentageは変わる。（大事）
    - ~10,000
        - Training, 60%
        - Validation, 20%
        - Testing, 20%
    - ~1,000,000
        - Having 20% of the data in the validation and test sets is unnecessary as the validation and test would contain far more samples than are needed for the purpose.
        - Training, 98%
        - Validation, 1%
        - Testing, 1%

Behavior of Split Specific Loss Functions

	Training	**Validation**	**Testing**	
	$J(\theta)_{train}$	$J(\theta)_{val}$	$J(\theta)_{test}$	$J(\theta)_{Minimum}$

	Training	Validation	Testing	
Good Estimator	0.21	0.25	0.30	0.18
Underfitting	1.9	1.9	2.1	
Overfitting	0.21	2.05	2.1	

- Good Estimator: the loss on the three sets are fairly consistent and the loss is close to the minimum achievable loss on the entire task.
- Underfitting: the neural network fails to bring the training loss down.
- Overfitting: caused by the neural network optimizing its parameters to precisely reproduce the training data output.
- The gap between training and validation loss is called the generalization gap.

Reducing the Effect of Underfitting/Overfitting
- Underfitting: (Training loss is high)
  - Train longer
    - If the architecture is suitable for the task at hand.
  - More layers or more parameters per layer.
  - Change architecture.
- Overfitting: (Generalization gap is large)（大事）
  - More training data.
    - Unfortunately, for self-driving cars, collecting training data is very expensive as it requires engineering time for data collection and a tremendous amount of annotator time to properly define the true output.
  - Regularization.
  - Change architecture.

- A dataset should be split to a training, a validation and a test split.
- Observing the performance on each of these splits helps in determining why a neural network is not performing well in the real world.
- Underfitting: Train longer or use a larger neural network.
- Overfitting: Regularization.
- A much more commonly faced scenario in self-driving car perception is overfitting.

Lesson 5: Neural Network Regularization

内容
- Remedy overfitting through various regularization strategies including:
  - Parameter norm penalties.
  - Dropout.
  - Early Stopping.

Toy Example
- New Design
  - 6 layer NN, 6 Hidden Units/Layer.
  - Train set Loss: 0.1.
  - Val set Loss: 0.45.
  - Minimum Loss achievable: 0.1.
- Overfitting is caused by the network learning the noise in the training data.
  - Because the neural network has so many parameters, it is able to curve out small regions in the space that correspond to the noisy training examples.

Parameter Norm Penalties

$$J(\theta)_{reg} = J(\theta) + \alpha\Omega(\theta)$$

- α is a hyperparameter that weights the relative contribution of the norm penalty to the value of the loss function.
- $\Omega(\theta)$ is a measure of how large θ's value is, usually an L_p Norm.

- Usually only constrain the size of weights and not biases: $J(\theta)_{reg} = J(\theta) + \alpha\Omega(W)$. 覚えてる！
  - This is motivated by the fact that the number of weights is much larger than the number of biases in the neural network.
  - しかしDeep Learning 7.1 Parameter Norm Penalties page 223によると、理由はこれじゃない。理由は2つある
    - First, “Each weight specifies how two variables interact. Fitting the weight well requires observing both variables in a variety of conditions. Each bias controls only a single variable. This means that we do not induce too much variance by leaving the biases unregularized.”
    - Second, “Regularizing the bias parameters can introduce a significant amount of underfitting.”

L2-Norm Parameter Penalty

$$\Omega(W) = \frac{1}{2}W^TW = \frac{1}{2}\|W\|_2^2.$$

- Toy exampleの結果
  - train set loss: 0.1 -> 0.176.
  - val set loss: 0.45 -> 0.182.
  - In this case, the decrease in the generalization gap is higher than the increase in training set loss.

Dropout

1. $P_{keep} = 0.5$（例えば）.
   - At every training iteration, this probability is used to choose a subset of the network nodes to keep in the network.
2. Evaluate the output y after cutting all the connections coming out of these units.
3. Multiply the final weights by P_{keep} at the end of training.

- Dropout can be intuitively explained as forcing the model to learn with missing input and hidden units.
  - Or in other words, with different version of itself.
- Computationally inexpensive but powerful regularization method.
- Does not significantly limit the type of model or training procedure that can be used.
- Dropout layers are practically implemented in all neural network libraries!
- We recommend using dropout whenever you have dense feedforward neural network layers.

Early Stopping

- Early stopping ends training when the validation loss keeps increasing for a preset number of iterations or epochs.
- After stopping the training algorithm, the set of parameters with the lowest validation loss is returned. stopした時のparametersではない。
- Early stopping should not be used as a first choice for regularization.
  - As it also limits the training time, which may interfere with the overall network performance.

Lesson 6: Convolutional Neural Networks

内容

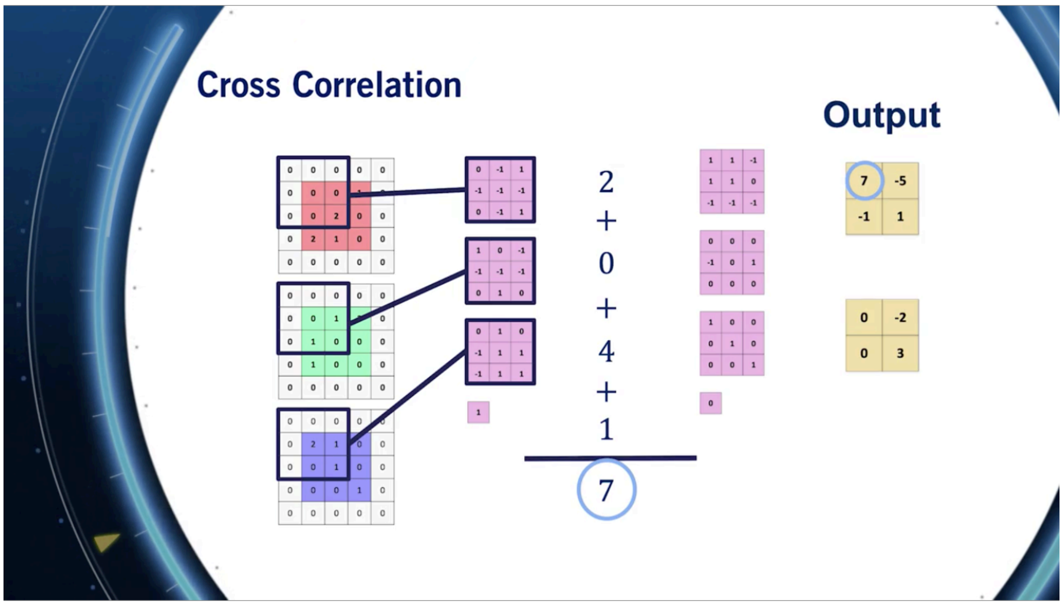
- How a neural network can use cross-correlation in its hidden layers instead of general matrix multiplication, to form ConvNets.
  - to tailor neural networks for image input data.
- The advantages of using ConvNets over traditional neural networks for processing images.

ConvNets

- Used for processing data defined on grid.
- 1D time series data (sampled at regular intervals), 2D images, 3D videos.
- Two major type of layers

- Convolution Layers.
  - Pooling Layers.
- Example: VGG 16.

Fully Connected VS Convolutional Layers
- Fully connected layers connect each node output to every node input in the next layer.
  - This is implemented in software through dense matrix multiplication.
  - $h_n = g(W^T h_{n-1} + b)$.
- Although counter-intuitive, convolutional layers use cross-correlation not convolutions for their linear operator instead of general matrix multiplication.（大事）
  - $h_n = g(W * h_{n-1} + b)$.
  - The logic behind using cross-correlation is that if the parameters are learned, it does not matter if we flip the output or not.
  - Since we are learning the weights of the convolutional layer, the flipping does not affect our results at all. よく分かっていない。
  - This results in sparse connectivity.
  - Each input element to the convolutional layer only affects a few output elements, thanks to the use of a limited size kernel for the convolutional operation.



Cross-Correlation（大事）
- Width: horizontal dimension of input volume.
- Height: vertical dimension of input volume.
- Depth: number of channels of input volume.
- Padding size: essential to retain shape.
- Each filter is comprised of a set of weights and a single bias.
  - The number of channels of the kernel needs to correspond to the number of channels of the input volume.
  - In this case, we have three weight channels per filter corresponding to red, green, and blue channels of the input image.
  - input imageの3つchannelsとfilterの3つchannelsとfilterのbiasが一緒に1つvalueを計算する。つまりinput imageのchannel数とoutputの形は関係ない。
- Usually we have multiple filters per convolutional layer.
- Notice that we get one output channel per filter.
- Stride: The number of pixels the filter is moved in the vertical and horizontal direction.

Output Volume Shape
- Filters are size $m \times m$.
- Number of filters $= K$.
- Stride $= S$, Padding $= P$.
- $W_{out} = \dfrac{W_{in} - m + 2 \times P}{S} + 1$.
- $H_{out} = \dfrac{H_{in} - m + 2 \times P}{S} + 1$.
- $D_{out} = K$.
- When designing ConvNets, it is very important to know what size output layers you'll end up with.
  - As an example, you don't want to reduce the size of your output volume too much if you are trying to detect small traffic signs and traffic lights on road scenes.

Pooling Layers: Max Pooling
- Pooling helps make the representations become invariant to small translations of the input.

Output Volume Shape
- Pool size $n \times n$.
- Stride $= S$.
- $W_{out} = \dfrac{W_{in} - n}{S} + 1$.
- $H_{out} = \dfrac{H_{in} - n}{S} + 1$.
- $D_{out} = D_{in}$.

Advantages of ConvNets
- Convolutional neural networks are by design, a natural choice to process images.
- Convolutional layers have less parameters than fully connected layers, reducing the chances of overfitting.
  - through parameters sharing and allows ConvNets to operate on larger images.
  - parameterが多すぎると、training dataを覚えちゃう。
- Convolutional layers use the same parameters to process every block of the image. Along with pooling layers, this leads to translation invariance which is particularly important for image understanding.
  - つまりtranslation invarianceの結果はpoolingからだけではなく、cross-correlationからも。
- ConvNets were one of the first neural network models to perform well at a time where other feedforward architectures failed.
  - In many ways, ConvNets carry the torch for the rest of deep learning, and pave the way to the relatively new acceptance of neural networks in general.
- ConvNets were one of the first neural network models to solve important commercial applications, such as handwritten digit recognition in the early 1990s. [LeCun et. al.]

学んだこと

1. （https://en.wikipedia.org/wiki/Deep_learning）によると
- For a feedforward neural network, the depth of the CAPs is that of the network and is the number of hidden layers plus one (as the output layer is also parameterized).
- For recurrent neural networks, in which a signal may propagate through a layer more than once, the CAP depth is potentially unlimited.
- CAP: credit assignment path, the chain of transformations from input to output.

2. np.maximum
- element-wise maximum of array elements.

- Compare two arrays and returns a new array containing the element-wise maxima. If one of the elements being compared is a NaN, then that element is returned. （つまり常にNaNをreturn）
  If both elements are NaNs then the first is returned.
- The maximum is equivalent to np.where(x1>=x2, x1, x2) when neither x1 nor x2 are nans, but it is faster and does proper broadcasting.
- Broadcastingについて、このページ（https://docs.scipy.org/doc/numpy/user/basics.broadcasting.html）まだ細かく読んでいないが。また、このページ（https://numpy.org/devdocs/user/theory.broadcasting.html）も: Array Broadcasting in Numpy.（大事）