Deep Learning

Ronald Parr

CompSci 570

Estimating the Gradient

- Recall: Backpropagation is gradient descent
- Computing exact gradient of the loss function requires summing over all training samples
- Why not update after each training sample?
  - Called online or stochastic gradient
  - Possibility of more efficient learning
    - Suppose you need only a small number of samples to estimate the gradient correctly?
    - Why do lots of unnecessary computation?
  - But, theoretically, ban be unstable unless you use a small step size

Batch/Minibatch Methods

- Find a sweet spot by estimating the gradient using a subset of the samples
- Randomly sample subsets of the training data and sum gradient computations over all samples in the subset
- Take advantage of parallel architectures (multicore/GPU)
- Still requires careful selection of step size and step size adjustment schedule – art vs. science

Tricks for Speeding Things Up

- Second order methods, e.g., Newton's method – may be computationally intensive in high dimensions
- Conjugate gradient is more computationally efficient, though not yet widely used
- Momentum: Use a combination of previous gradients to smooth out oscillations

Tricks For Breaking Down Problems

- Built up deep networks by training shallow networks, then feeding their output into new layers (may help with vanishing gradient and other problems) – a form of "pretraining"
- Train the network to solve "easier" problems first, then train on harder problems – curriculum learning, a form of "shaping"

Convolutional Neural Networks (CNNs)

- Championed by LeCun (1998)
- Originally used for handwriting recognition
- Now used in state of the art systems in many computer vision applications
- Well-suited to data with a grid-like structure

Convolutions

- What is a convolution?
- Way to combine two functions, e.g., x and w:

$$s(t) = \int x(a)w(t-a)da$$

Entire Domain

- Discrete version

$$s(t) = \sum x(a)w(t-a)$$

Convolutions on Grids

- For image I
- Convolution "kernel" K:

$$S(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) = \sum_m \sum_n I(i-m,j-n)K(m,n)$$

Convolution on Grid Example

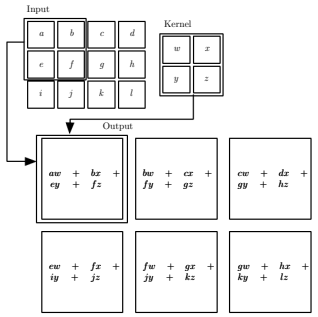


Figure 9.1 from **Deep Learning**, Ian Goodfellow and Yoshua Bengio and Aaron Courville

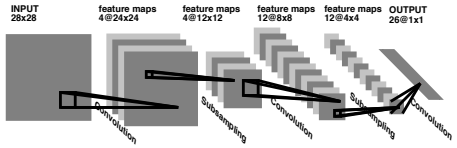
Application to Images & Nets

- Images have huge input space: 1000x1000=1M
- Fully connected layers = huge number of weights, slow training
- Convolutional layers reduce connectivity by connecting only an mxn window around each pixel
- Use *weight sharing* to learn a common set of weights so that same convolution is applied everywhere

Additional Stages

- Convolutional stages feed to *detector* stages
- Detectors are nonlinear
- Detectors feed to pool stages
- Pooling stages summarizing upstream nodes, e.g., average, 2-norm, max (should we be worried that max isn't differentiable?)

Example Convolutional Network



From, **Convolutional Networks for Images**, Speech, and Time-Series, LeCun & Bengio

N.B.: Subsampling = averaging

Why This Works

- ConvNets use weight sharing to reduce the number of parameters learned – mitigates problems with big networks
- Can be structured to learn scale and position invariant feature detectors
- Final layers then combine feature to learn the target function
- Can be viewed as doing **simultaneous feature selection and classification**

ConvNets in Practice

- Most successful applications still require some thought about the stucture – not yet a turnkey solution
- Number of convolutional layers, form of pooling and detecting units may be application specific