Statistical Machine Learning   Lecture 21 – Concentration of Measure   11/11/2014

Lecturer: Prof. Homrighausen   Scribe: Chuck Vollmer

1 Chuck Vollmer: Intro to Deep Learning

Check out the IPython notebook that charlie showed the class. It gives an overview from Autoencoders to Denoising Autoencoders, then on to the famous Restricted Boltzmann Machines (rBMs) which are the classical form of a generative network, used for pre-training neural networks. It also has working code that implements all of the above on GPU's. You will need to install some things to make it run, however. If you need any help, email charlie (charlesv@rams.colostate.edu) and he will help you set it all up.

IPython plug:

IPython can be downloaded from www.ipython.org or from a number of python distributions: charlie recommends anaconda (http://continuum.io/downloads), as anaconda comes with the packages (libraries) that you need for most statistical modeling, like SciPy, Pandas, and NumPy. NumPy, by the way, solves a lot of issues with multidimensional arrays that R has (sparce structures is one of them). If you are using Windows, charlie has no idea what you should do, other than recommend your switching to Unix.

Another note about IPython notebooks is that you can interactively code in R, C, and Bash all from within the notebook! Great!

2 Deep Learning: An Overview

(Darren's Disclaimer: "These notes are largely from a conversation with Rob Tibshirani. The ideas contained herein are partially his, and will appear in a future book 'L1 methods and the Lasso' ")

Neural networks are models for supervised learning, where linear combinations of features are fed through nonlinear functions repeatedly. At the top layer, the resulting latent factor is fed into a linear logistic regression.

The central idea is referred to as *greedy layerwise unsupervised pre-training* (Terminology appeared in Bengio et al. (2007)).

Here, we wish to learn a hierarchy of features one level at a time, using

1. unsupervised feature learning to learn a new transformation at each level
2. which gets composed with the previously learned transformations

Essentially, each iteration of unsupervised feature learning adds one layer of weights to a deep neural network.

The top layer is used to initialize a (supervised) neural network.

2.1 The Traditional Approach:

Traditionally, a neural net is fit to all *labelled* data in one operation, with weights randomly chosen near zero.

Due to the nonconvexity of the objective function, the final solution can get 'caught' in poor local minima.

2.2 The Aim of Deep Learning:

Deep learning seeks to find a good starting value, while allowing for:

- ...modeling the joint distribution of the covariates separately
- ...use of unlabeled data (included the test covariates)

3 AutoEncoders

In neural networks the idea of an *AutoEncoder* generalizes the ideas of PCA and sparse coding by

- Using multiple hidden layers, leading to a hierarchy of dictionaries

  (As PCA is linear, composing multiple layers adds no generality. Sparse coding provides only 1 layer between covariates and the representation.)
- applying the encoding models to local patches of an image, commonly with weight sharing where constraint weights are enforced to be equal across an image

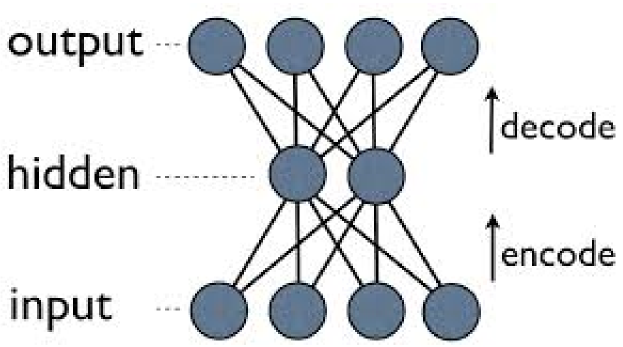
  (This is the so-called convolutional neural network framework.)

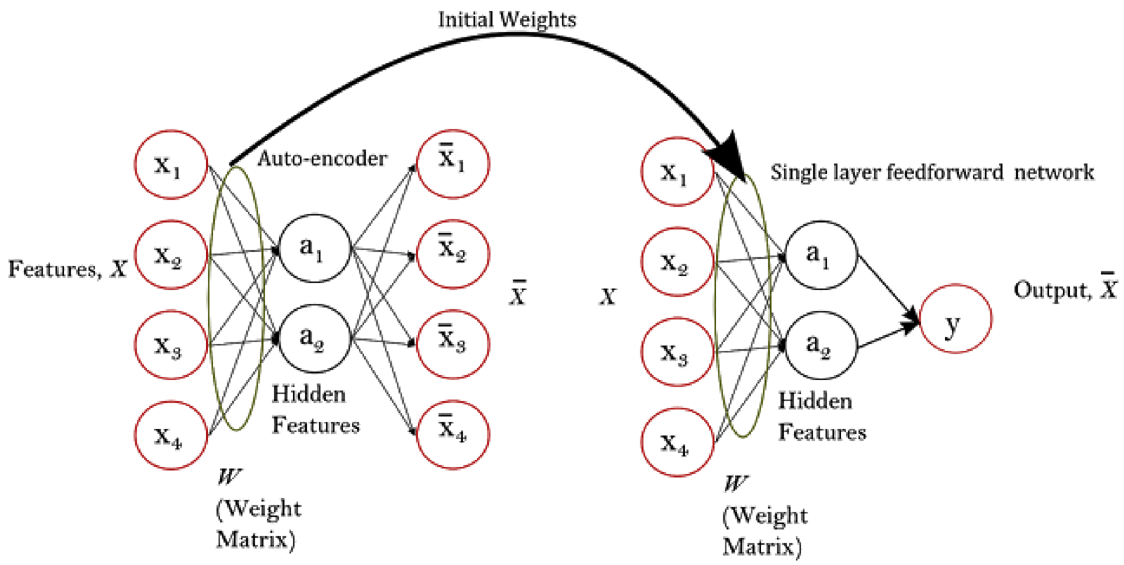
An ***AutoEncoder*** is comprised of (LeCun (1987); Hinton, Zemel (1994)):

- *Feature-extracting function:* This function $h : \mathbb{R}^p \to \mathbb{R}^K$ maps the covariates to a new representation and is also known as the *encoder*
- *Reconstruction function:* This function (Darren labeled this function h^{-1} to be suggestive, but he doesn't mean that $h^{-1}(h(x)) = x$) $h^{-1} : \mathbb{R}^K \to \mathbb{R}^p$ is also known as the *decoder* and it maps the representation back into the original space

Goal: Optimize any free parameters in the encoder/decoder pair that minimizes reconstruction error

AutoEncoders are depicted below:





Of course this means some sort of implicit or explicit constraint need be imposed to not learn the identity function.

This comes about via a combination of

- ... Regularization

  (Usually called a regularized auto-encoder.)
- ... Dimensional constraint

  (Usually called a classical auto-encoder.)

All flavors essentially reduce to solving the following optimization problem (perhaps with constraints)

$$\min \sum_{i=1}^{n} \ell(X_i, h^{-1}h(X_i))$$

3.1 Classic AutoEncoders

As auto-encoders were first presented in the context on neural networks, they tend to have the following linear (really, it is affine with the inclusion of a bias term) form:

Let $W \in \mathbb{R}^{p \times K}$ (with $K < p$) be a matrix of weights. Each linear combination of an input vector X is fed through a nonlinear function σ, creating:

$$h(X) = \sigma(W^\top X) \in \mathbb{R}^K$$

The output layer is then modeled as a linear combination of these inputs (there is no restriction that the same matrix to be used in h and h^{-1}. Keeping them the same is known as *weight-tying.*)

$$h^{-1}(h(X)) = Wh(X) = W\sigma(W^\top X) \in \mathbb{R}^p$$

Then... given inputs $X_1, \ldots, X_n$, the weight matrix W is estimated by solving the (non convex) optimization problem:

$$\min_{W \in \mathbb{R}^{p \times K}} \sum_{i=1}^{n} ||X_i - Wh(X_i)||^2$$

If $\sigma(X) \equiv X$, then $h(X) = W^\top X$ and we've recovered the PCA program

(In the sense that we've recovered the same subspace.)

The framework is determined by the relative sizes of K and p

- If $K < p$, the rank constraint provides a *bottleneck* in the network that forces the learning of structure (e.g. PCA)
- If $K > p$, the representation is overcomplete and some regularization is needed
  (e.g. sparse coding)
  Regularization comes about in several ways
  - Adding a regularization term on the *parameters* to the objective function
  - Corrupting the inputs before auto-encoding and comparing to uncorrupted inputs
    (This is known as a *denoising auto-encoder*)
  - Adding a regularization term on the *Jacobian* of the encoder to the objective function
    (This is known as a *contractive auto-encoder*)

3.2 Modern Deep Learning

Modern deep learning generalize the previous definition in several ways

(See Le, Ranzato, Monga, Devin, Chen, Dean, Ng (2012) for details)

- They use multiple hidden layers, leading to a hierarchy of dictionaries
- Include nonlinearities that can be computed faster (such as $\sigma(x) = x_+$
- The encoding is applied to local patches of images (or signals) and these patches might be forced to have the same weights, imposing *weight-sharing* or a *convolutional* structure