TTIC 31230, Fundamentals of Deep Learning

David McAllester, April 2017

Multilayer Perceptrons

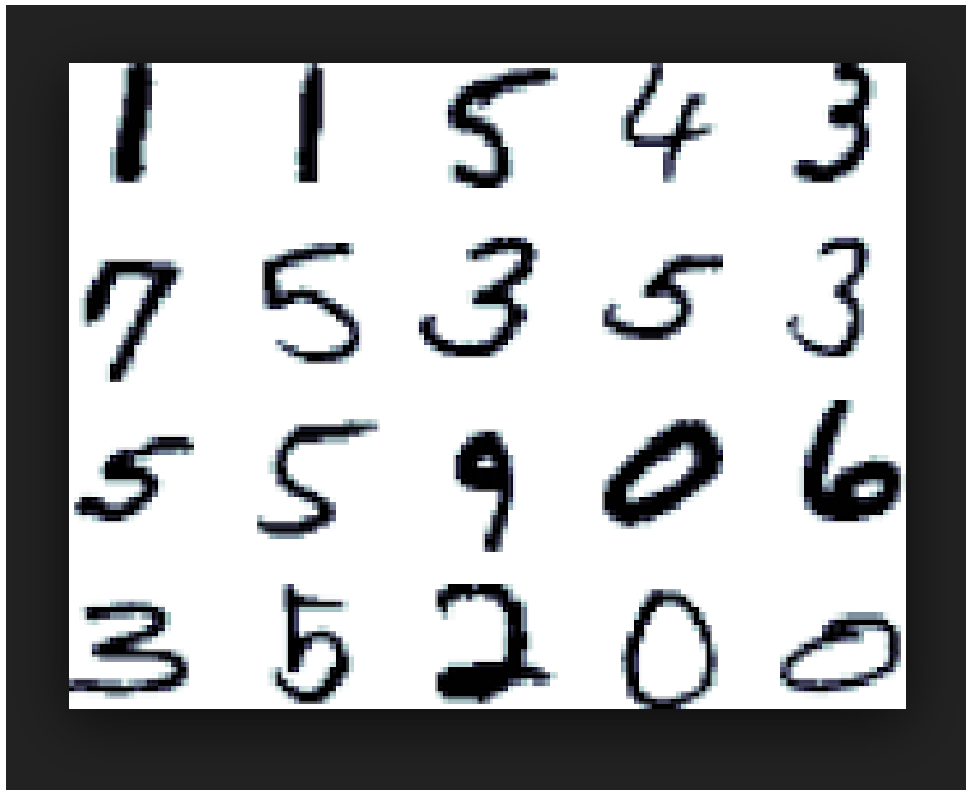
Stochastic Gradient Descent

Multiclass Classification

We will start by considering the problem of multiclass classification.

We consider the problem of taking an input x (such as an image of a hand written digit) and classifying it into some small number of classes (such as the digits 0 through 9).

MNIST



Multiclass Classification

Assume a data distribution D on pairs (x, y) for $x \in \mathbb{R}^d$ and $y \in \mathcal{C}$.

For MNIST x is a 28×28 image which we take to be a 784 dimensional vector giving $x \in \mathbb{R}^{784}$.

For MNIST $\mathcal{C}$ is the set $\{0, \ldots, 9\}$.

Assume a sample (x_0, y_0), $\ldots$, (x_{N-1}, y_{N-1}) drawn from D.

We want to use the sample to construct a rule for predicting y given x.

Class Scores

Assume a sample $(x_0, y_0),\ \ldots,\ (x_{N-1}, y_{N-1})$ drawn from D with $x \in \mathbb{R}^d$ and $y \in \{0, \ldots, K\}$.

For a new x we compute a score $s(\hat{y})$ for each possible label $\hat{y}$.

$$s(\hat{y}) = \sum_{j=1}^{d} W_{\hat{y},j}\; x_j + b_{\hat{y}}$$

or in vector notation

$$s = Wx + b$$

Here $W_{\hat{y},j}$ is the weight on component j of x for predicting class $\hat{y}$ and $b_{\hat{y}}$ is a "bias term" for class $\hat{y}$.

Softmax

We can convert the scores to probabilities using a Gibbs distribution

$$P(\hat{y}) = \frac{1}{Z}\, e^{s(\hat{y})}$$

$$Z = \sum_{\hat{y}} e^{s(\hat{y})}$$

Softmax

In vector notation

$$P = \mathrm{softmax}\ Wx + b$$

$$(\mathrm{softmax}\ s)_i = \frac{1}{Z}\, e^{s_i}$$

$$Z = \sum_i e^{s_i}$$

Log Loss

we have

$$P_{W,b}(\cdot|x) = \text{softmax } Wx + b$$

We can define our "error" or "loss" to be negative log probability of the true label.

$$\textbf{loss}(P(y|x)) = -\log P(y|x) = \log \frac{1}{P(y|x)}$$

We want

$$W^*, b^* = \underset{W,b}{\text{argmin}} \;\; \mathrm{E}_{(x,y)\sim D} \left[\log \frac{1}{P_{W,b}(y|x)} \right]$$

Multiclass Logistic Regression

For now we consider

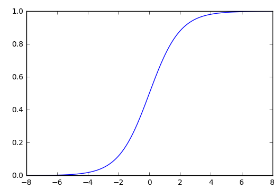
$$W^*, b^* = \operatorname*{argmin}_{W,b} \ \ell_{\mathrm{train}}(W, b)$$

$$\ell_{\mathrm{train}}(W, b) = \frac{1}{N} \sum_n \log \frac{1}{P_{W,b}(y_n | x_n)}$$

This is multiclass logistic regression.

Multi Layer Perceptrons (MLPs)

$$\sigma(u) = \frac{1}{1+e^{-u}}$$



$$\begin{aligned} L^0 &= \sigma(W^0 x + b^0) \\ L^1 &= \mathrm{softmax}(W^1 L^0 + b^1) \end{aligned}$$

In the first equaiton σ is applied to each component of the vector $W^0x + b^0$. In general we will use the notation $f(x)$ where f is a scalar function and x is a vector (or tensor) to denote the vector (or tensor) that results from applying f to each element of x.

MLPs

$$
\begin{aligned}
L^0 &= \sigma(W^0 x + b^0) \\
L^1 &= \text{softmax}(W^1 L^0 + b^1)
\end{aligned}
$$

Here L^0 and L^1 are vectors. We will call the elements of L^0 "channels" (also called units or neurons).

The elements of L^1 are the class probabilities.

We now learn W^0, b^0, W^1 and b^1.

A More General Setting

Consider a system of parameters Θ.

For a two-layer MLP for MNIST we have that Θ is a tuple (W^0, b^0, W^1, b^1).

Consider a scalar loss function $\ell(\Theta, x, y)$.

For MNIST we have

$$\ell(\Theta, x, y) = -\log P_\Theta(y|x) = \log \frac{1}{P_\Theta(y|x)}$$

This is a very common loss function.

Optimizing the Loss Function

We consider minimizing the loss on the training data.

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \; \frac{1}{N} \sum_{i=1}^{N} \ell(\Theta, x_i, y_i)$$

We will do this by gradient descent.

Gradients with Respect to Systems of Parameters

$\nabla_\Theta\, \ell(\Theta, x, y)$ denotes the partial derivative of $\ell(\Theta, x, y)$ with respect to the parameter system Θ.

For a scalar loss $\ell(\Theta, x, y)$ we have that $\nabla_\Theta\, \ell(\Theta, x, y)$ has the same shape (scalar, vector, or tensor) as Θ.

For each real number component of Θ there is a corresponding component of $\nabla_\Theta\, \ell(\Theta, x, y)$ giving the partial derivative of $\ell(\Theta, x, y)$ with respect to that component of Θ.

Here can think of Θ as a single vector with

$$(\nabla_\Theta\, \ell(\Theta, x, y))_i = \partial \ell(\Theta, x, y)/\partial \Theta_i$$

Total Gradient Descent

$$\ell_n(\Theta) = \ell(\Theta,\ x_n,\ y_n)$$

$$\ell(\Theta) = \frac{1}{N}\sum_{n=0}^{N-1} \ell_n(\Theta)$$

We want: $\Theta^* = \underset{\Theta}{\operatorname{argmin}}\, \ell(\Theta)$

repeat:

$$\Theta \;\texttt{-=}\; \eta\, \nabla_\Theta\, \ell(\Theta)$$

Stochastic Gradient Descent (SGD)

repeat: Select n at random.

$$\Theta \;\texttt{-=}\; \eta \, \nabla_\Theta \, \ell_n(\Theta)$$

SGD can make progress with only a small subset of the training data.

Note that

$$\mathrm{E}_n \left[\nabla_\Theta \, \ell_n(\Theta)\right] = \sum_n P(n) \, \nabla_\Theta \ell_n(\Theta)$$

$$= \nabla_\Theta \, \ell(\Theta)$$

SGD for MLPs

Consider an MLP

$$\Theta = (W^0, b^0, W^1, b^1)$$

$$L^0 = \sigma(W^0 x_n + b^0)$$

$$L^1 = \mathrm{softmax}(W^1 L^0 + b^1)$$

$$\ell(\Theta, x, y) = -\log(L^1_y)$$

We now need to be able to compute $\frac{\partial \ell(\Theta,x,y)}{\partial \Theta_k}$ for all scalar parameters Θ_k. To be continued ...

END