Lecture 7: Neural Nets and the Learning Function

Agenda

- Construction of Neural Nets

- Distance Matrices

Source: Sections VII.1 and IV.10 in Linear Algebra and Learning from Data (2019) by **Gilbert Strang**

Construction of Neural Nets

Learning Function F (X,V) where x are the weights, and v are the feature vectors, the sample feature vectors (training dataset)

So those feature vectors V_0 come from the training data, either one at a time, if we're using stochastic gradient descent (discussed detail later) with mini-batch size 1

Or B at a time, if we're doing mini-batch of size B, or the whole thing, a whole epoch at once, if we're doing full-scale gradient vector

So those are the feature vectors, and these are the numbers in the linear steps, the weights

Weight matrix A_K multiply by V and bias vectors b_K that adds on to shift the origin. **Optimize X and V**

Take first step of learning Function F:

v_1=ReLU (F(A_1,b_1, v_0)) → non-Linear Step

ReLU (Rectified Linear Unit) is an activation function.

Construction of Neural Nets

Deep learning is Continuous Piecewise Linear (CPL) functions.
Linear for simplicity, continuous to model an unknown but reasonable rule, and
piecewise to achieve the nonlinearity that is an absolute requirement for real images and data

Here is a first construction of a piecewise linear function of the data vector v.
Choose a matrix A_1 and vector b1.
Then set to zero (this is the nonlinear step) all negative components of $A_1 v + b_1$.
Then multiply by a matrix A_2 to produce 10 outputs in $w = F(v) = A_2(A_1 v + b_1)_+ \ldots + A_{10}(A_{10} v + b_{10})$
That vector $(A_1 v + b_1)_+ ..$ forms a "hidden layer" between the input v and the output w.

Construction of Neural Nets

$(A_1 v_0 + b_1) \rightarrow$ linear step
$V_1 = \text{ReLU}\ (A_1 v_0 + b_1)$ non-linear step

Generally, $V_k = \text{ReLU}\ (A_{k-1} v_{k-1} + b_{k-1})$ where k=1, …l (l::layers}

1-Training sample

$$V_1 = \text{ReLU}\ (A_1 \begin{pmatrix} v_0 \end{pmatrix} + b_1)$$

$$V_l = A_l v_{l-1} + b_l$$

Don't do ReLU at the last layer, so it's just $A_l\ v_{l-1} + b_l$.

May not do a bias vector also at that layer, but you might, and this is the finally the output.

So this picture is clearer to distinguish between the weights

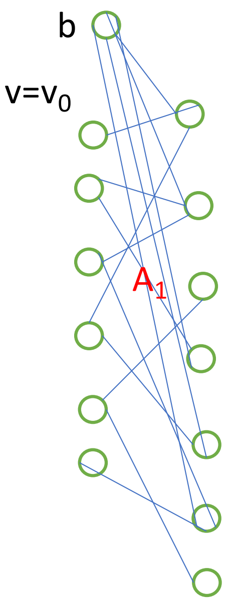
$x = A_1, b_1, A_2, b_2 \ldots . A_l . b_l$

x really stands for all the weights that we compute up to A_l, b_l, so that's a collection of all the weights

Often weights x's are undetermined because the number of X's in A's, b's are greater than the number of v's in the training sets

Construction of Neural Nets

1[st] layer:$v_1 = A_1 v_0 + b_1$



Choose x to min Loss Function L= $(\frac{1}{N})$ [$\sum_{i=1}^{N}$ F(x,vi)

L(x) = $(\frac{1}{N})$ [$\sum_{i=1}^{N}$ F(x,vi)−truei]

Question: do we use the whole function L at each iteration, or do we just pick only b, of the samples to look at iteration number K?
So this is the L(x) then added up over all v's. This is what the neural net produces. It's supposed to be close to the true

Popular Loss functions:

(1) Square loss= sum of $\| \|_2^2$ → regression

(2) L^1 loss = sum of $\| \|^1$ → Lasso

(3) Hinge loss (-1,1 Classification)

(4) Cross-entropy loss (neural nets)

Distance Matrices

Question: distances squared:: $\|x_i-x_j\|^2=d_{ij}$. Find positions x_j in R^d (also find d i.e, the dimension of the space).

$\|x_i-x_j\|^2$=given d_{ij} . Find x's. Given D ={d_{ij}} distances matrix, to find X matrix which gives the positions

In machine learning, you're given a whole lot of points in space, feature vectors (points) in a high-dimensional space, and those are related i.e., they are connected.

They tend to fit on a surface in high-dimensional space, a low- dimensional surface in high-dimensional space

Let's recognize the connection between distances and positions:

$D=d_{ij}= \|x_i-x_j\|^2=(x_i-x_i)^T(x_i-x_j)=x_i^Tx_i - x_i^Tx_j - x_j^Tx_i +x_j^Tx_j$ (entries in D) (1)

$x_i^Tx_i$ produces a matrix with constant rows (no dependence on j). $x_j^Tx_j$ produces a matrix with constant columns (no dependence on i) (For detail See Appendix foil 12)

$\|x_i\|^2$ and $\|x_j\|^2$ in both of those matrices are on the main diagonal of $G = X^TX$

Those are the numbers in the column vector diag(G) (For detail See Appendix foil 12)

Distance Matrices

Middle terms $-2x_i^T x_j$ in (1) in last foil, $-2G = -2X^TX$

*R*ewrite (1) as an equation for the matrix *D,* using the symbol **1** for the column vector of *n* ones

That gives constant columns and $\mathbf{1}^T$ gives constant rows

So, $D = \mathbf{1}\,\mathrm{diag}(G)^T - 2G + \mathrm{diag}(G)\,\mathbf{1}^T$ (2)

(Note: deduction of equation 2 is given in the next foil and For detail See Appendix in last foil)

Given D Find X // actually find $X^TX=G$ then find X from G

We'll find X^TX

Because we have dot products of X's. Find out what $x_i.x_j$ is.

Let's call this matrix G for the dot product matrix, and then find X from G.

Now let us say diagonal matrix $D_{ii}=(x_i,x_i)$. Let us write an equation for G with dot matrix X^TX

Distance Matrices

Solve equation (2) (from last foil) for $G = X^T X$

Place first point at the origin : $x_1 = 0$. For every $\|x_i - x_1\|^2$ is $\|x_i\|^2$
First column d_i of D (which is given) is the same as

$\mathrm{diag}(X^TX) = \mathrm{diag}(G) = (\|x_1\|^2, \|x_2\|^2, \dots, \|x_n\|^2) \rightarrow \mathrm{diag}(G) = d_1$ and $\mathrm{diag}(G)\,\mathbf{1}^T = d_1\mathbf{1}^T$

$$X^TX = G = -\tfrac{1}{2} D + \tfrac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} d \end{pmatrix}^T + \tfrac{1}{2} \begin{pmatrix} d \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}^T$$

Every colm   Every row

Now G comes from D. G is positive semidefinite provided the distances in D satisfy the triangle inequality

Ref:Menger: *Amer. J. Math.* 53; Schoenberg: *Annals Math.* 36

Matrix form

$$D = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}^T + \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}^T - 2XX^T$$

This is the key equation

$$XX^T = \tfrac{1}{2}\left[D - \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} d_1 \\ : \\ d_i \end{pmatrix}^T - \begin{pmatrix} d_1 \\ : \\ d_i \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}^T\right]$$

Distance Matrices

Given XX^T Find X (nxn)

Two leading candidates

(1) Evaluate of $XX^T = Q\Lambda Q^T$
(2) Elimination on XX^T

Find X up to an orthogonal transformation, as XX^T is symmetric

XX^T is positive or semidefinite, this is semidefinite. Given a semidefinite matrix and find a square root. Matrix is the XX^T and find X

There are many candidates, because if you find one i.e., any QX .
Because Q^TQ in there, it's the identity.

Distance Matrices

(1) Take $X = Q\sqrt{\Lambda}Q^T = X^T$
$XX^T = (Q\sqrt{\Lambda}Q^T)(Q^T\sqrt{\Lambda}Q) = \Lambda = I$=identity matrix

(2) Elimination on XX^T = LDU (L, a lower triangular, times D, the pivots, times U, the upper triangle)
= LDL^T (U is replaced by L^T)

Then $X = \sqrt{D}L^T$ (This is the Cholesky Factorization)

(Note: when X^TX, then X^TX coming correctly. X^T will be L^T. Transpose will give L. Square root of D will be $\sqrt{D}$. We'll give the D, and then the L^T is right)

Appendix

D= d_{12} = $||x_1 - x_2||^2 = (x_1 - x_2)^2 = x_1^2 - 2\, x_1 x_2 - x_2^{\,2} = x_1^T.x_1 - 2\, x_1^T x_2 + x_2^T.x_2$

$x_1^T.x_1 = \begin{pmatrix} x_1 \\ 0 \end{pmatrix} (x_1 \quad 0)$ and $2x_1^T.x_2 = 2 \begin{pmatrix} x_1 \\ 0 \end{pmatrix} (x_2 \quad 0)$ and $x_2^T.x_2 = \begin{pmatrix} x_2 \\ 0 \end{pmatrix} (x_2 \quad 0)$

$\begin{pmatrix} x_1 \\ 0 \end{pmatrix} (x_1 \quad 0) = \mathbf{1}\ \mathbf{diag} \left[\begin{pmatrix} x_1 \\ 0 \end{pmatrix} (x_1 \quad 0) \right]^{\mathbf{T}} = \mathbf{1} \begin{pmatrix} x_1.x_1^T \\ \mathbf{0} \end{pmatrix}^{\mathbf{T}}$

Similarly,

D= $d_{12} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \mathbf{diag} \begin{pmatrix} x_1^2 \\ \mathbf{0} \end{pmatrix}^{\mathbf{T}} - 2 \begin{pmatrix} x_1 \\ 0 \end{pmatrix} (x_2 \quad 0) + \mathbf{diag} \begin{pmatrix} x_1^2 \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}^{\mathbf{T}}$

D = $\mathbf{1}$ diag(G)^T- 2G + diag(G) $\mathbf{1}^T$ where G= $x_1^T.x_2$