Advanced Machine Learning Subsidary Notes

Lecture 6: Understand Mappings

Adam Prügel-Bennett

April 21, 2020

1 Keywords

- Mappings, Eigenvectors

2 Main Points

2.1 Inverse Problems

- Much of machine learning can be viewed as solving an inverse problem
- We collect data about the world by performing a series of measurements
- Our task is to infer properties of the world from the data

2.2 Over-Constrained Problems

- We can have contradictory data that our model cannot explain
- This may arise because
  - We have errors in the data
  - Our data contains insufficient information
  - Our model is too simple
- If we have more training data than free parameters this is likely to occur
- We typically solve this by introducing a loss function we minimise
- A classic example is to minimise the squared error

2.3 Under-Constrained Problems

- We can also be in a situation when many models (learning machines) explain the data
- This will typically happen when we have more free parameters than data
- Here we have to choose a particular model
- To do this requires (implicitly or explicitly) making additional assumptions
- For high-dimensional inputs we can be over-constrained in some directions and under-constrained in others

2.4 Ill-Conditioning

- Even when we are not under-constrained our inverse can be very sensitive to the data
- That is small errors can be strongly magnified
- Ill-condition leads to high variance in the bias-variance sense and hence poor generalisation

2.5 Linear Regression

- In linear regression we try to fit a linear model $y_i = \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w}$ (or in matrix form $\boldsymbol{y} = \mathbf{X}\,\boldsymbol{w}$)
- We use a squared error (so can cope with conflicting constraints)
- If we have more training examples than parameters the solution is given by the pseudo-inverse $\boldsymbol{w} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\boldsymbol{y}$
- It we have less training examples than parameters (or we are unlucky in that the training examples don't span the full space) then the problem is under-constrained and there are an infinity of solutions
- Even when we have more training examples than parameters the problem can be ill-conditioned

2.6 Eigen-Systems

- We can understand ill-conditioning for linear regression by the eigen-decomposition of $\mathbf{M} = \mathbf{X}^{\mathsf{T}}\mathbf{X}$
- This should be revision
- You know that an *eigenvector*, $\boldsymbol{v}$, satisfies $\mathbf{M}\,\boldsymbol{v} = \lambda\,\boldsymbol{v}$
- For a symmetric matrix there are n real orthogonal eigenvectors
- You can prove they are orthogonal
- **Orthogonal Matrices**
  - Putting the n eigenvectors into a matrix $\mathbf{V}$ with columns $\boldsymbol{v}_i$ we obtain an orthogonal matrix
  - The defining property of an orthogonal matrix is $\mathbf{V}^{\mathsf{T}}\mathbf{V} = \mathbf{V}\mathbf{V}^{\mathsf{T}} = \mathbf{I}$
  - They correspond to rotations (with a possible reflection)
- **Matrix Decomposition**
  - We can decompose a symmetric matrix, $\mathbf{M}$ as

$$\mathbf{M} = \mathbf{V}\,\boldsymbol{\Lambda}\,\mathbf{V}^{\mathsf{T}}$$

  - Where $\boldsymbol{\Lambda}$ is a diagonal matrix of eigenvalues of $\mathbf{M}$ (i.e. $\Lambda_{ii} = \lambda_i$)
  - $\mathbf{V}$ is the orthogonal matrix made up of the eigenvectors of $\mathbf{M}$
  - We can interpret the mapping of a symmetric matrix $\mathbf{M}$ as equivalent to
    1. a rotation defined by $\mathbf{V}^{\mathsf{T}}$
    2. scaling of the i^{th} component by λ_i and
    3. a rotation backwards given by V$
  - Equivalently if we work in a coordinate system defined by the eigenvectors of $\mathbf{V}$ (this forms an orthonormal basis set) then we just rescale in the directions $\boldsymbol{v}_i$ by λ_i
    * A symmetric matrix just squashes or expands in different orthogonal directions (this is what the eigensystem captures)

- **Inverse Matrices**
  - The inverse of a symmetric matrix $\mathbf{M}$ is given by
$$\mathbf{M} = \mathbf{V}\,\mathbf{\Lambda}^{-1}\,\mathbf{V}^{\mathsf{T}}$$
  - Where $\mathbf{\Lambda}^{-1}$ is a diagonal matrix with elements $\Lambda_{ii}^{-1} = 1/\lambda_i$
  - This is only defined if $\mathbf{M}$ if all the eigenvalues of $\mathbf{M}$ are non-zero ($\mathbf{M}$ is said to be full rank)
  - If λ_i is very small then $1/\lambda_i$ is large and in taking the inverse $\mathbf{M}^{-1}\boldsymbol{x}$ any component of $\boldsymbol{x}$ in the direction $\boldsymbol{v}_i$ will get magnified by $1/\lambda_i$
  - For linear regression we invert $\mathbf{M} = \mathbf{X}^{\mathsf{T}}\mathbf{X}$
    * in directions where the training examples don't vary much the associated eigenvalue will be small and the inverse inherently unstable

3 Exercises

3.1 Linear Regression

- Derive the formula for the weight vector in linear regression

4 Experiments

4.1 Eigensystems

- In either Matlab/Octave or python generate random matrices and check the matrix identities

```
X = randn(5,4)   % generate a mock designer matrix with 5 inputs of length 4
M = X'*X         % compute a symmetrix matrix
[V.L] = eig(M)   % compute eigenvalues
V*L*V'           % should be identical to M
V*V'             % should be the identity matrix (up to rounding precision)
V'*V             % should be the identity matrix (up to rounding precision)
x = randn(4,1)   % generate a random column matrix of length 4
y = randn(4,1)   % generate another random column matrix of length 4
xp = V*x         % apply V to x
yp = V*y         % apply V to y
norm(x)          % compute Euclidean norm of x
norm(xp)         % should be the same as Euclidean nor of xp
x'*y             % compute inner product of x and y
xp'*yp'          % compute inner produce of xp and yp (should be the same as above)

Z = rand(4,5)    % consider a designer matrix where we would have more unknowns the examples
W = Z'*Z         % compute a covariance type matrix (except we don't subtract the mean
eig(W)           % compute eigenvalues (one should be 0 up to machine precision)
```