Introduction to Machine learning

Srinandan ("Sri") Dasmahapatra
Katayoun (Kate) Farrahi: Neural Networks lectures (weeks 9, 10)

- 80% examination, 20% coursework
- Lab sessions: Wed 1100-1300 odd-numbered weeks
- Python (anaconda), jupyter notebooks (remember to copy out code yourself, and run the code sequentially)
- Lots of mathematical exercises
Supervised Learning: labelled data

- Given data $\mathcal{D}$ construct model $f$ such that the distance between model output and real “output” is small
- If $y_n$ is discrete (e.g., +1/-1), $f$ is a classification model; if continuous, $f$ is a regression model
- Learning = model construction by minimising loss $L$

$$\mathcal{D} := \{(x_n, y_n)\}, \ n = 1, \ldots, N$$

$$x_n \rightarrow \text{Model} \rightarrow \hat{y}_n$$

$$\hat{y}_n = f(x_n)$$

$$L = \sum_{n=1}^{N} d(\hat{y}_n, y_n)$$
Unsupervised Learning: unlabelled data

- Group data into similar subsets (cluster label $c_n$)

- Representation learning — express data in transformed manner: $Z_n$ captures “essence”

$\mathcal{D} := \{x_n\}, n = 1, \ldots, N$

$$
\begin{align*}
D &:= \{x_n\}, n = 1, \ldots, N \\
x_n &\rightarrow \text{Clustering} \rightarrow c_n \\
x_n &\rightarrow \text{encode} \rightarrow z_n \rightarrow \text{decode} \rightarrow \hat{x}_n \\
z_n &= f_e(x_n), \hat{x}_n = f_d(z_n) \\
L &= \sum_{n=1}^{N} d(\hat{x}_n, x_n)
\end{align*}
$$
Geoffrey Hinton:

“The paradigm for intelligence was logical reasoning and the idea of what an internal representation would look like was it would be some kind of symbolic structure.

That has completely changed with these big neural nets.

We now think of internal representation as great big vectors and we do not think of logic as the paradigm for how to get things to work.

[…] reasoning is just sequences of such state vectors.”
ML is agnostic to origin of data; how can you lend credibility to results if you cannot appeal to domain-specific understanding?

ML methods: Regression and classification

Mathematical foundations

A. Optimisation — minimising loss; calculus

B. Linear algebra — conceptualise solutions

C. Probability theory — from distances to improbability; the data you have that you want to describe well, it’s the data you have not seen; typicality, generalisation

ML methods: Unsupervised and representation learning — clustering, PCA, kernel methods

ML methods: neural networks
Core idea in ML: Reduce mismatch between prediction and data (Empirical risk minimisation)

- Regression models minimise residuals — deviations of model predictions from outputs in training data
- Loss is square(length) of $N$-dimensional residual vector

\[
y_i = \hat{y}_i + (y_i - \hat{y}_i) = f(x_i) + r_i
\]

Loss function \( l(\theta, x_n, y_n) = (y_n - f(x_n; \theta))^2 = r_n^2 \)

\[
L(\theta) = \mathbb{E}_{(x_n, y_n) \sim \mathcal{D}} l(\theta, x_n, y_n) = \frac{1}{N} \sum_{n=1}^{N} r_n(\theta)^2
\]

Minimise this
Update weights to reduce loss

- Fit \( y = wx \) to data, slope \( w \) of the line is the weight to be learnt.
- Loss \( L = \) sum of squares of residuals, 3 possible residuals for input \( x_i \) shown.
- Update weights \( w^{(t)} \) to \( w^{(t+1)} \) so that \( L(w^{(t+1)}) < L(w^{(t)}) \).
- Change weights in the direction opposite to the slope of the loss function
  \[
  w^{(t+1)} = w^{(t)} + \eta \left( \frac{dL(w^{(t)})}{dw} \right), \quad \eta < 0
  \]
Reduce loss by gradient descent: optimisation

- Extend the same idea to higher dimensional weight spaces
- Evaluate partial derivatives for direction of weight changes (linear approx.)

\[
(\nabla_w L)_i = \frac{\partial L}{\partial w_i}
\]

loss = \((y - (w_1 x_1 + w_2 x_2))^2\)

\[
L(w) = \mathbb{E}_{(x_n, y_n) \sim \mathcal{D}} l(w, x_n, y_n)
\]

\[
w^{(t+1)} = w^{(t)} - \eta \nabla_w L, \text{ gradient descent (GD), batch}
\]

\[
w^{(t+1)} = w^{(t)} - \eta \nabla_w l(w, x_n, y_n), \text{ stochastic gradient descent (SGD)}
\]

\[
w^{(t+1)} = w^{(t)} - \eta \nabla_w \mathbb{E}_{n \sim S \subset \mathcal{D}} l(w, x_n, y_n), \text{ GD, minibatch}
\]
Linear Regression: solving for zero gradient of loss

- Follow gradients until (local) minimum
- Solution for weights by setting gradient to zero

\[
0 = \frac{\partial}{\partial w} L(w) = \frac{1}{N} \frac{\partial}{\partial w} \sum_i (y_i - wx_i)^2 = \frac{2}{N} \sum_i (y_i - wx_i)(-x_i) \\
\implies w = \frac{\sum_i y_i x_i}{\sum_i x_i^2}
\]
Viewing distances as probabilities

- Different ways of conceptualising the algorithmic task
- **Minimising** squared loss equivalent to **maximising** (log-) likelihood with Gaussian errors
- Probabilistic methods suggest ways of combining evidence from multiple sources (including priors)

\[
p(r_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{r_i^2}{2\sigma^2}}
\]

Independent errors (multiply probabilities):

\[
-\ln \left( \prod_{i=i}^{N} p(r_i) \right) = \frac{1}{2\sigma^2} \sum_{i=1}^{N} r_i^2 + \text{const}
\]
Classification: discrete output

\[ \mathcal{D} := \{ (x_n, y_n) \}, \quad n = 1, \ldots, N \]

- Given training set represented by points labelled green and red ...
- ... where each point has two features \( X_1, X_2 \)
- ... find function \( f(x_1, x_2) \) that reproduces given labels

\[ \hat{y}_n = f(x_{n,1}, x_{n,2}) = \begin{cases} +1, & \text{red} \\ -1, & \text{green} \end{cases} \]
Find equation for decision boundary

- Every point on \((X_1, X_2)\) plane is divided into separate regions — one for each label
- Learning algorithm finds equation for boundary

\[
f(x_1, x_2; \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2
\]

\[
\begin{align*}
&f(x_{a,1}, x_{a,2}) > 0, & +1, \text{ red} \\
&f(x_{a,1}, x_{b,2}) < 0, & -1, \text{ green}
\end{align*}
\]
Choosing labels with high/low scores: max, softmax

- Items $x_{(n)}$ assigned red-/green-ness (+1/-1) scores of $s$
- If $s$ higher/lower than threshold $-w_0$, $x_{(n)}$ qualifies for a label (+1/-1)
- **Task**: learn weights $\mathbf{w}$ that minimises misclassification loss

$$\sum_{n=1}^{N} \max \left(0, -y_n (w_0 + w_1 x_{n,1} + w_2 x_{n,2})\right)$$

Replace piecewise linear function by smooth non-linear function

$$\max(a, b) = \begin{cases} 
a, & a > b \\
b, & a < b
\end{cases} \quad \rightarrow \quad \text{softmax}(a, b) = \ln \left( e^a + e^b \right) .$$

* Other names: max cost, hinge loss, ReLU (rectified linear unit)
Easier to do gradient descent with softmax

\[
\text{loss} = \sum_{n=1}^{N} \max (0, -y_n (w_0 + w_1 x_{n,1} + \cdots + w_p x_{n,p})) = \sum_n \max (0, -y_n (w_0 + \mathbf{w}^T \mathbf{x}_n)) \quad \text{positive if incorrect}
\]

\[
\text{trivial solution exists}
\]

\[
\text{Optimisation: } \arg \min_{w_0, \mathbf{w}} L(w_0, \mathbf{w}) = \sum_n \text{softmax} (0, -y_n (w_0 + \mathbf{w}^T \mathbf{x}_n))
\]

\[
= \sum_n \left( 1 + \exp \left( -y_n \begin{bmatrix} w_0 \\mathbf{w}^T \end{bmatrix}^T \begin{bmatrix} 1 \\ \mathbf{x}_n \end{bmatrix} \right) \right) = \sum_n (1 + \exp(-y_n \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_n)) \equiv L(\tilde{\mathbf{w}})
\]

\[
\text{gradient descent: } \tilde{\mathbf{w}}^{(t+1)} = \tilde{\mathbf{w}}^{(t)} - \eta \nabla_{\tilde{\mathbf{w}}} L,
\]
Reinterpret distances as probabilities

- Convert score into probability of class label
- Far from decision boundary, model predicts lower probability of error (more confident)

\[ \sigma(x; \beta) = \frac{1}{1 + e^{-\beta x}} \]

\( \beta = 1 \) red, \( \beta = (1/5) \) green
Class probability from max /softmax

\[
\frac{f(0, (x - 5))}{f(0, -(x - 5)) + f(0, (x - 5))}
\]

\[
\frac{f(0, -(x - 5))}{f(0, -(x - 5)) + f(0, (x - 5))}
\]
2-class sigmoid from Bayes’ rule

Joint distribution \( p(C_1, x) = p(C_1 | x)p(x) = p(x | C_1)p(C_1) \)

Bayes' rule relates conditionals: \( p(C_1 | x) = \frac{p(x | C_1)p(C_1)}{p(x)} \)

Data generated from either class: \( p(C_1 | x) = \frac{p(x | C_1)p(C_1)}{p(x | C_1)p(C_1) + p(x | C_2)p(C_2)} \)

\[
p(C_1 | x) = \frac{1}{1 + \frac{p(x | C_2)p(C_2)}{p(x | C_1)p(C_1)}} = \frac{1}{1 + \exp(-a)} \triangleq \sigma(a), \ a = \ln \left( \frac{p(x | C_2)p(C_2)}{p(x | C_1)p(C_1)} \right)
\]

Log odds \( \frac{p(C_2 | x)}{p(C_1 | x)} \), \( \sigma(-a) = 1 - \sigma(a), \ a = \ln \left( \frac{\sigma}{1 - \sigma} \right) \)
Learning or memorising? Bias-variance tradeoff

- Evaluate model performance on test data (generalisation)
- Training data set is only one glimpse of the world
- Different training sets can lead to different models with different model predictions and different residuals
- Characterise model on distribution of residuals trained on different subsets of available training data
- *k*-fold cross validation
Polynomial fits with high degrees tend to overfit

\[ \hat{y}(x; w) = w_0 + w_1 x + \cdots + w_d x^d \]

\[ \text{Loss}_\mathcal{D}(w) = \mathbb{E}_{(x_n, y_n) \sim \mathcal{D}}(y_n - \hat{y}(x_n; w))^2 \]

From C Bishop, PRML
Weights learned by minimising loss

<table>
<thead>
<tr>
<th></th>
<th>$M = 0$</th>
<th>$M = 1$</th>
<th>$M = 3$</th>
<th>$M = 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0^*$</td>
<td>0.19</td>
<td>0.82</td>
<td>0.31</td>
<td>0.35</td>
</tr>
<tr>
<td>$w_1^*$</td>
<td>-1.27</td>
<td>7.99</td>
<td></td>
<td>232.37</td>
</tr>
<tr>
<td>$w_2^*$</td>
<td></td>
<td>-25.43</td>
<td>-5321.83</td>
<td></td>
</tr>
<tr>
<td>$w_3^*$</td>
<td></td>
<td>17.37</td>
<td>48568.31</td>
<td></td>
</tr>
<tr>
<td>$w_4^*$</td>
<td></td>
<td></td>
<td>-231639.30</td>
<td></td>
</tr>
<tr>
<td>$w_5^*$</td>
<td></td>
<td></td>
<td>640042.26</td>
<td></td>
</tr>
<tr>
<td>$w_6^*$</td>
<td></td>
<td></td>
<td>-1061800.52</td>
<td></td>
</tr>
<tr>
<td>$w_7^*$</td>
<td></td>
<td></td>
<td>1042400.18</td>
<td></td>
</tr>
<tr>
<td>$w_8^*$</td>
<td></td>
<td></td>
<td>-557682.99</td>
<td></td>
</tr>
<tr>
<td>$w_9^*$</td>
<td></td>
<td></td>
<td>125201.43</td>
<td></td>
</tr>
</tbody>
</table>

From C Bishop, PRML
Length of weight vectors to control for high degree

\[ \text{length of weight vector} = \| w \|^2 = \sum_{i=0}^{d} w_i^2 \]

L2 norm: \( \ell_2(w) \triangleq \| w \|_2 = \| w \|^2 \)

\[ \text{Loss}_\mathcal{D}(w) = \mathbb{E}_{(x_n, y_n) \sim \mathcal{D}} (y_n - \hat{y}(x_n; w))^2 \]

\[ w^* = \arg \min_w \left( \text{Loss}_\mathcal{D}(w) + \lambda \ell_2(w) \right) \text{ regularisation} \]
Relative penalties for loss and weight length

From C Bishop, PRML
Supervised Learning: reduce loss

- Predict and correct using weight-adjustable functions
- Optimise weights using loss function — learning as optimisation
- Interpret weight spaces in terms of mathematical framework of linear algebra
- Interpret distances and loss metrics in terms of probability theory
- Evaluate performance in terms of generalisation (bias/variance)
- Introduce regularisation for generalisation