Regression

Outline

Regression	Univariate	Multivariate
Linear	✔	✔
Non-Linear	✔	✔

Linear models

- We consider the case $\mathbf{x} \in \mathbb{R}^d$ throughout this chapter
- Function $f : \mathbb{R}^d \to \mathbb{R}$ is *linear* if for some $\mathbf{w} \in \mathbb{R}^d$ it can be written as

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} = \sum_{j=1}^{d} w_j x_j$$

  and *affine* if for some $\mathbf{w} \in \mathbb{R}^d$ and $a \in \mathbb{R}$ we can write

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + a$$

- $\mathbf{w}$ is often called *weight vector* and a is called *intercept* (or particularly in machine learning literature, *bias*)

Linear models (2)

- Linear model generally means using an affine function by itself for regression, or as scoring function for classification
- The learning problem is to determine the parameters $\mathbf{w}$ and a based on data
- Linear regression and classification have been extensively studies in statistics

Univariate linear regression

- As warm-up, we consider linear regression in one-dimensional case $d = 1$
- We use square error and want to minimise it on training set $(x_1, y_1), \dots, (x_n, y_n)$
- Thus, we want to find $a, w \in \mathbb{R}$ that minimise

$$E(w, a) = \sum_{i=1}^{n} (y_i - (wx_i + a))^2$$

- This is known as *ordinary least squares* and can be motivated as maximum likelihood estimate for (w, a) if we assume

$$y_i = wx_i + a + \eta_i$$

where η_i are i.i.d. Gaussian noise with zero mean

Univariate linear regression (2)

- We solve the minimisation problem by setting the partial derivatives to zero
- We denote the solution by $(\hat{w}, \hat{a})$
- We have
$$\frac{\partial E(w,a)}{\partial a} = -2\sum_{i=1}^{n}(y_i - wx_i - a)$$
and setting this to zero gives
$$\hat{a} = \bar{y} - w\bar{x}$$
where $\bar{y} = (1/n)\sum_i y_i$ and $\bar{x} = (1/n)\sum_i x_i$
- This implies in particular that the point $(\bar{x}, \bar{y})$ is on the line $y = \hat{w}x + \hat{a}$

Univariate linear regression (3)

- Further,

$$\frac{\partial E(w,a)}{\partial w} = -2\sum_{i=1}^{n} x_i(y_i - wx_i - a)$$

- Plugging in $a = \hat{a}$ and setting the derivative to zero gives us

$$\sum_{i=1}^{n} x_i(y_i - wx_i - \bar{y} + w\bar{x}) = 0$$

  from which we can solve

$$\hat{w} = \frac{\sum_{i=1}^{N} x_i(y_i - \bar{y})}{\sum_{i=1}^{N} x_i(x_i - \bar{x})}$$

Univariate linear regression (4)

- Since
$$\sum_{i=1}^{n} \bar{x}(y_i - \bar{y}) = \bar{x}\left(\sum_{i=1}^{n} y_i - n\bar{y}\right) = 0$$
and
$$\sum_{i=1}^{n} \bar{x}(x_i - \bar{x}) = \bar{x}\left(\sum_{i=1}^{n} x_i - n\bar{x}\right) = 0$$
we can finally rewrite this as
$$\hat{w} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

- Notice that we have $\hat{w} = \sigma_{xy}/\sigma_{xx}$ where σ_{pq} is sample covariance between p and q:
$$\sigma_{pq} = \frac{1}{n-1}\sum_{i=1}^{n}(p_i - \bar{p})(q_i - \bar{q})$$

Useful trick

- In more general situation than univariate regression, it would often be simpler to learn just linear functions and not worry about the intercept term
- An easy trick for this is to replace each instance $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ by $\mathbf{x}' = (1, x_1, \ldots, x_d) \in \mathbb{R}^{d+1}$
- Now an affine function $f(x) = \mathbf{w} \cdot \mathbf{x} + a$ in $\mathbb{R}^d$ becomes linear function $g(x') = \mathbf{w}' \cdot \mathbf{x}'$ where $\mathbf{w}' = (a, w_1, \ldots, w_d)$
- If we write the set of instances $\mathbf{x}_1, \ldots, \mathbf{x}_n$ as an $n \times d$ matrix, this means adding an extra column of ones
- This is known as using homogeneous coordinates (textbook p. 24)

Useful trick (2)

- For most part we now present algorithms for learning linear functions (instead of affine)
- In practice, to run them on d-dimensional data, we add the column of ones and run the algorithm in $d + 1$ dimensions
- The first component of $\mathbf{w}$ then gives the intercept
- However sometimes we might still want to treat the intercept separately (for example in *regularisation*)

Multivariate linear regression

- We now move to the general case of learning a linear function $\mathbb{R}^d \to \mathbb{R}$ for arbitrary d
- As discussed above, we omit the intercept
- We still use the square loss, which is by far the most commonly used loss for linear regression
- One potential problem with square loss is its sensitivity to *outliers*
  - one alternative is absolute loss $\left|y - \hat{f}(x)\right|$
  - computations become trickier with absolute loss

Multivariate linear regression (2)

- We assume matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ has n instances $\mathbf{x}_i$ as its rows and $\mathbf{y} \in \mathbb{R}^n$ contains the corresponding labels y_i

- We write

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$$

  where the *residual* $\epsilon_i = y_i - \mathbf{w} \cdot \mathbf{x}_i$ indicates error that weight vector $\mathbf{w}$ makes on data point $(\mathbf{x}_i, y_i)$

- Our goal is to find $\mathbf{w}$ which minimises the sum of squared residuals

$$\sum_{i=1}^{n} \epsilon_i^2 = \|\boldsymbol{\epsilon}\|_2^2$$

Multivariate linear regression (4)

- Write $\mathbf{y}_0 = \mathbf{X}\mathbf{w}$, so our goal is to minimise $\|\epsilon\|_2 = \|\mathbf{y} - \mathbf{y}_0\|_2$
- Since $\mathbf{w} \in \mathbb{R}^d$ can be anything, $\mathbf{y}_0$ can be any vector in the linear span S of the *columns* of $\mathbf{X}$
- In other words, $\mathbf{y}_0 \in S = \mathrm{span}(\mathbf{c}_1, \ldots, \mathbf{c}_d)$ where $\mathbf{c}_j = (x_{1j}, \ldots, x_{dj})$ is jth column of $\mathbf{X}$ and

$$\mathrm{span}(\mathbf{c}_1, \ldots, \mathbf{c}_d) = \left\{ w_1\mathbf{c}_1 + \cdots + w_d\mathbf{c}_d) \mid \mathbf{w} \in \mathbb{R}^d \right\}$$

Multivariate linear regression (5)

- Since S is a linear subspace of $\mathbb{R}^n$, the minimum of $\|\mathbf{y} - \mathbf{y}_0\|_2$ subject to $\mathbf{y}_0 \in S$ occurs when $\mathbf{y}_0$ is the projection of $\mathbf{y}$ to S
- Therefore in particular $\mathbf{y} \cdot \mathbf{c}_j = \mathbf{y}_0 \cdot \mathbf{c}_j$ for $j = 1, \ldots, d$
- Since $\mathbf{y} \cdot \mathbf{c}_j = (\mathbf{X}^{\mathrm{T}}\mathbf{y})_j$, we write this in matrix form as

  $$\mathbf{X}^{\mathrm{T}}\mathbf{y} = \mathbf{X}^{\mathrm{T}}\mathbf{y}_0 = \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w}$$

  where we have substituted back $\mathbf{y}_0 = \mathbf{X}\mathbf{w}$
- Multiplying both sides by $(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}$ gives the solution

  $$\hat{w} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

Multivariate linear regression (6)

- If the columns $\mathbf{c}_j$ of $\mathbf{X}$ are linearly independent, the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is of full rank and has an inverse
- For $n > d$ this is true except for degenerate special cases
- $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is a $d \times d$ matrix, and inverting it takes $O(d^3)$ time
- For very high dimensional problems the computation time may be prohibitive

Nonlinear models by transforming the input

- *Linear* regression can also be used to fit models which are *nonlinear* functions of the input
- Example: For fitting a degree 5 polynomial

$$y_i = f(x_i) = w_0 + w_1 x_i + w_2 x_i^2 + w_3 x_i^3 + w_4 x_i^4 + w_5 x_i^5$$

... create the input matrix

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 & x_1^5 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 & x_2^5 \\ 1 & x_3 & x_3^2 & x_3^3 & x_3^4 & x_3^5 \\ 1 & x_4 & x_4^2 & x_4^3 & x_4^4 & x_4^5 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \end{pmatrix}, \quad \text{and} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \end{pmatrix}$$

Nonlinear predictors by transforming the input (2)

- We can also explicitly include some interaction terms, as in

$$y_i = f(\mathbf{x}_i) = w_0 + w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i1} x_{i2}$$

using the following input matrix:

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & x_{11}x_{12} \\ 1 & x_{21} & x_{22} & x_{21}x_{22} \\ 1 & x_{31} & x_{32} & x_{31}x_{32} \\ 1 & x_{41} & x_{42} & x_{41}x_{42} \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}, \quad \text{and} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \end{pmatrix}$$

Regularised regression

- If dimensionality d is high, linear models are actually quite flexible
- We can avoid overfitting by minimising not just the squared error $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$ but the regularised cost

$$\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

  where $\lambda > 0$ is a constant (chosen e.g. by cross validation)
- By increasing λ we decrease variance but increase bias
- This allows us to *sometimes* get sensible results even in the case $n < d$

Regularised regression (2)

- Minimising cost function

$$\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

is known as *ridge regression* and has closed form solution

$$\hat{w} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

- Popular alternative is *lasso* where we minimise

$$\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1$$

- Replacing 2-norm with 1-norm encourages *sparse* solutions where many weights w_i get set to zero

- There is no closed form solution to lasso, but efficient numerical packages exist