TAM 598 Lecture 13:

Supervised Learning – Linear Regression via Least Squares

Announcements:

- HW 3 covers lectures 8-12; due on Mar 12
- One recorded class over spring break
- No class Monday Mar 24th

supervised learning - given pairs of x and y observations, find the map between x and y.

Classification - when y is discrete → aka logistic regression
regression - when y is continuous

I. Linear Regression via Least Squares - Supervised Learning

given n d-dimensional inputs $\underset{\sim}{x}_{1:n} = \{\underset{\sim}{x}_1, \underset{\sim}{x}_2, \ldots, \underset{\sim}{x}_n\}$ ← features

and n outputs: $y_{1:n} = \{y_1, \ldots, y_n\}$ ← targets

"linear" means that the map from $\underset{\sim}{x}$ to y is linear

Two common ways to train a supervised learning model

(1) optimization - find parameters by maximizing the likelihood of the data (or minimize the neg. log likelihood, the loss)

(2) probabilistic inference: (a) Bayes to get posterior given data, (b) use posterior to make predictions

II. Linear Regression with a single variable - Example

a synthetic data set x, y :

$$y_i = -0.5 + 2x_i + 0.1\,\epsilon_i$$

↓ input

$x_i \sim \mathcal{U}([0,1])$

↳ gaussian noise

$\epsilon_i \sim N(0,1)$

use least squares to fit to the linear model

$$y = w_0 + w_1 x$$

$\underset{\sim}{w} = (w_0, w_1)^T$ parameters

by minimizing the square loss

$$L(\underset{\sim}{w}) = \sum_{i=1}^{n} \left(y_i - w_0 - w_1 x_i \right)^2 = \| \underset{\sim}{y} - \underset{\approx}{X}\, \underset{\sim}{w} \|^2$$

$$L(\underset{\sim}{w}) = \sum_{i=1}^{n} \left(y_i - w_0 - w_1 x_i \right)^2 = \| \underset{\sim}{y} - \underset{\approx}{X}\,\underset{\sim}{w} \|^2$$

$\underset{\sim}{y}$ = vector of observations $= (y_1 \dots y_n)$

$\underset{\sim}{w}$ = weight vector $= (w_0, w_1)^T$

$\underset{\approx}{X}$ = design matrix $= \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$ $n \times 2$

LHS

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \end{bmatrix} = \begin{bmatrix} 4 & x_1+x_2+x_3+x_4 \\ \Sigma x_i & \Sigma x_i^2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} \Sigma y_i \\ \Sigma x_i y_i \end{bmatrix}$$

RHS

$$\underset{\approx}{X}^T \underset{\approx}{X}\,\underset{\sim}{w} = \underset{\approx}{X}^T \underset{\sim}{y}$$

$$L = \| \underset{\sim}{r} \|^2 = \underset{\sim}{r}^T \underset{\sim}{r}$$

to minimize:

$$\nabla_{\underset{\sim}{w}} L = 2 \| \underset{\sim}{r} \| \frac{d\|\underset{\sim}{r}\|}{dw} = 0$$

Note: $\frac{d\|r\|}{dr} = \text{sign}(r)$

So $\frac{dL}{d\underset{\sim}{w}} = 2 \| \underset{\sim}{r} \| \frac{d\|\underset{\sim}{r}\|}{dw}$

$= 2 \| \underset{\sim}{r} \| \frac{d\|\underset{\sim}{r}\|}{dr} \frac{dr}{dw}$

$= 2 \| \underset{\sim}{r} \| \text{sign}(\underset{\sim}{r}) \left(-\underset{\approx}{X} \right)$

$= 2 \left(\underset{\sim}{y} - \underset{\approx}{X}\,\underset{\sim}{w} \right) \left(-\underset{\approx}{X} \right)$

$\Rightarrow 0 = \underset{\sim}{y} - \underset{\approx}{X}\,\underset{\sim}{w}$

$\underset{\approx}{X}\,\underset{\sim}{w} = \underset{\sim}{y}$

$\underset{\approx}{X}^T \underset{\approx}{X}\,\underset{\sim}{w} = \underset{\approx}{X}^T \underset{\sim}{y}$ solution

Example 2: how things can go wrong

try instead: $y_i = -0.5 + 2x_i + 2x_i^2 + \epsilon_i$

and we still try to fit a linear model.

This is **underfitting**

- model is too simple to capture underlying patterns.
- often happens when model has high bias, performs poorly on training & unseen test data

III. Polynomial Regression

$$y_i = -0.5 + 2x_i + 2x_i^2 + \epsilon_i$$

Now we try $y = w_0 + w_1 x + w_2 x^2$ and minimize the square loss

$$L(w_0, w_1, w_2) = \sum_{i=1}^{n} \left(y_i - w_0 - w_1 x_i - w_2 x_i^2 \right)^2$$

$$L(\underset{\sim}{w}) = \| \underset{\sim}{y} - \underset{\approx}{X}\, \underset{\sim}{w} \|^2$$

where $\underset{\sim}{w} = (w_0, w_1, w_2)^T$ (3×1)

$$\underset{\approx}{X} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \quad (n \times 3)$$

$\underset{\sim}{y} = (y_1 \cdots y_n)^T$ $(n\times 3)$

another linear system

$$\underset{\approx}{X}^T \underset{\approx}{X}\, \underset{\sim}{w} = \underset{\approx}{X}^T \underset{\sim}{y}$$

solve for $\underset{\sim}{w}$

linear in $\underset{\sim}{w}$

Example 2 : what if we used a higher degree polynomial?

$$\underset{\sim}{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^e \\ 1 & x_2 & x_2^2 & & x_2^e \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^e \end{bmatrix} \qquad (n \times e)$$

$$\underset{\sim}{w} = (w_0, w_1, \ldots, w_e)^T \qquad (e \times 1)$$

if e becomes too large: overfitting

model is too complex relative to amount of training data

model learns not only the true patterns, but also noise & fluctuations.

performs very well on training data, but poorly on new, unseen data

IV The Generalized Linear Model

$$y(\underset{\sim}{x}; \underset{\sim}{w}) = \sum_{j=1}^{m} w_j \phi_j(\underset{\sim}{x}) = \underset{\sim}{w}^T \underset{\sim}{\phi}(\underset{\sim}{x})$$

weights $\underset{\sim}{w}$

$\underset{\sim}{\phi} = (\phi_1, \ldots, \phi_m)^T$ arbitrary basis functions

model is linear in $\underset{\sim}{w}$, but basis functions $\phi(\underset{\sim}{x})$ can be non-linear

eg) our polynomial model : $\phi_1(x) = 1 \quad \phi_2(x) = x \quad \phi_3(x) = x^2$

eg) multivariate linear regression where $\underset{\sim}{x} = (x_1, \ldots, x_d)$

$$y = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d$$

is also a gen. linear model where $\phi_1(\underset{\sim}{x}) = 1$

$\phi_2(\underset{\sim}{x}) = x_1$

$\phi_3(\underset{\sim}{x}) = x_2$

$\vdots$

eg) multi-dimensional polynomials

$$\phi_j(\underset{\sim}{x}) = \sum_{\alpha \in A_j} \beta_\alpha \underset{\sim}{x}^\alpha$$

eg) radial basis functions

$$\phi_j(\underset{\sim}{x}) = \exp\left\{ - \frac{\| \underset{\sim}{x} - \underset{\sim}{x}_j \|^2}{2\ell^2} \right\}$$

eg) Fourier series

$$\phi_{2j}(x) = \cos\left(\frac{2\pi j}{L} x\right)$$

$$\phi_{2j+1}(x) = \sin\left(\frac{2\pi j}{L} x\right)$$

How to fit the generalized model using least squares:

quadratic loss function

$$\mathcal{L}(\underset{\sim}{w}) = \sum_{i=1}^{N} \left[y(\underset{\sim}{x}_i; \underset{\sim}{w}) - y_i \right]^2$$

$$= \left\| \underline{\underline{\Phi}}\, \underset{\sim}{w} - \underline{y} \right\|^2$$

$$= \left(\underline{\underline{\Phi}}\, \underset{\sim}{w} - \underline{y} \right)^T \left(\underline{\underline{\Phi}}\, \underset{\sim}{w} - \underline{y} \right)$$

$\underline{\underline{\Phi}}$ = design matrix $\quad \underline{\underline{\Phi}} \in \mathbb{R}^{N \times M}$, $\Phi_{ij} = \phi_j(\underset{\sim}{x}_i)$

N = # observations ; M = # basis functions

To minimize the loss:

① Take $\nabla_{\underset{\sim}{w}} \mathcal{L}(\underset{\sim}{w})$, set $= 0$, solve for $\underset{\sim}{w}$

solution: $\left(\underline{\underline{\Phi}}^T \underline{\underline{\Phi}}\right) \underline{w} = \left(\underline{\underline{\Phi}}^T\right) \underline{y}$ (as before)

- solve with numpy.linalg.lstsq
- provide $\underline{\underline{\Phi}}$, $\underline{y}$ and it returns $\underline{w}$

IV. Measures of Predictive Accuracy

(1) Training and test datasets

- not a good idea to test how good your model is using the training dataset
- need to test model on unseen data - a test data set
- often, take your data and split (eg 70% train, 30% test)
- can plot predictions vs. observations

(2) Mean-squared error - scalar measure of goodness of fit

$$MSE = \frac{1}{n_t} \sum_{i=1}^{n_t} \left[y_{t,i} - w^T \phi(x_{t,i}) \right]^2$$

hard to understand the absolute meaning of MSE, so we can instead use relative MSE (RMSE)

$$\text{RMSE} = \frac{\text{MSE of your model}}{\text{MSE of simplest possible model}}$$

simplest: $y_{\text{simplest}} = \hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} y_i$

if RMSE < 1 $\longrightarrow$ you are doing better than the simplest possible model

RMSE > 1 $\longrightarrow$ the simplest possible model outperforms your

(3) Coefficient of Determination R^2

$R^2 = 1 - \text{RMSE}$: tells us what % of the variance of the test data is explained by your model