THE UNIVERSITY OF MELBOURNE

Linear Regression

Semester 1, 2021

Ling Luo

Outline

- Regression
- Parameter Estimation
  - Optimisation
  - Computing the Optimal Solution
  - Gradient Descent
- Application and Evaluation

Categorical Features and Classes

- For classification problems, classes are categorical
- Attributes can be categorical or continuous
- Continuous attributes lead to different methods or strategies, e.g. Naïve Bayes with continuous attributes

- What if the **class** were continuous?

Regression

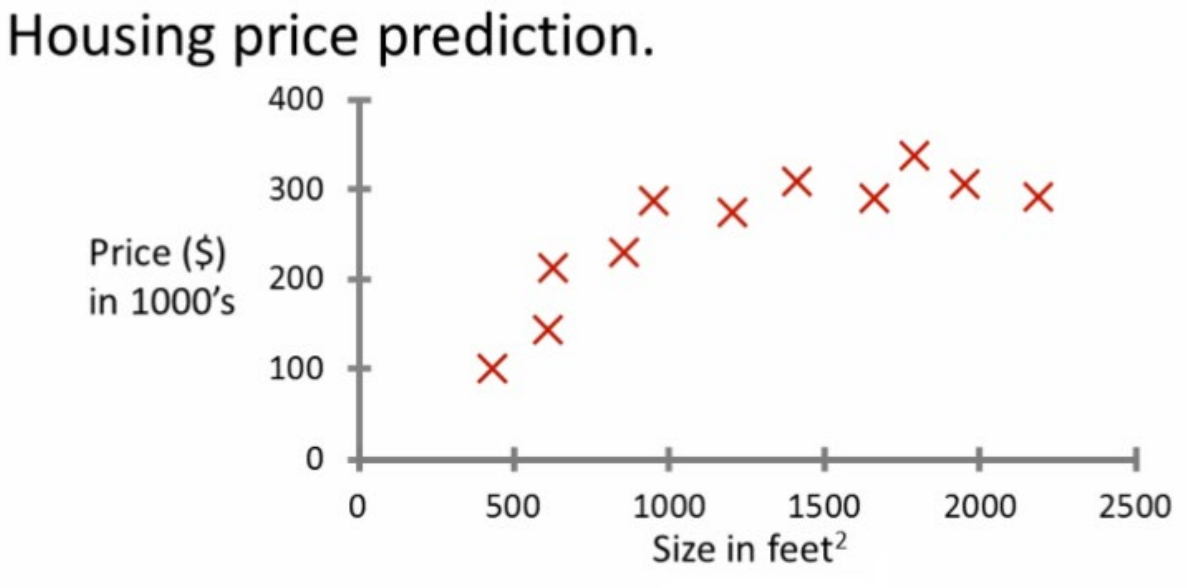
- Continuous attributes → continuous class
- Assuming a linear relationship between the attribute values a_k and the continuous class c, we have:

$$c = w_0 + \sum_{k=1}^{D} w_k a_k$$

where w_k is a weight corresponding to a_k

Regression

- Can we predict housing prices?
- A friend has a house which is 750 square feet. Can we estimate the price of this house?



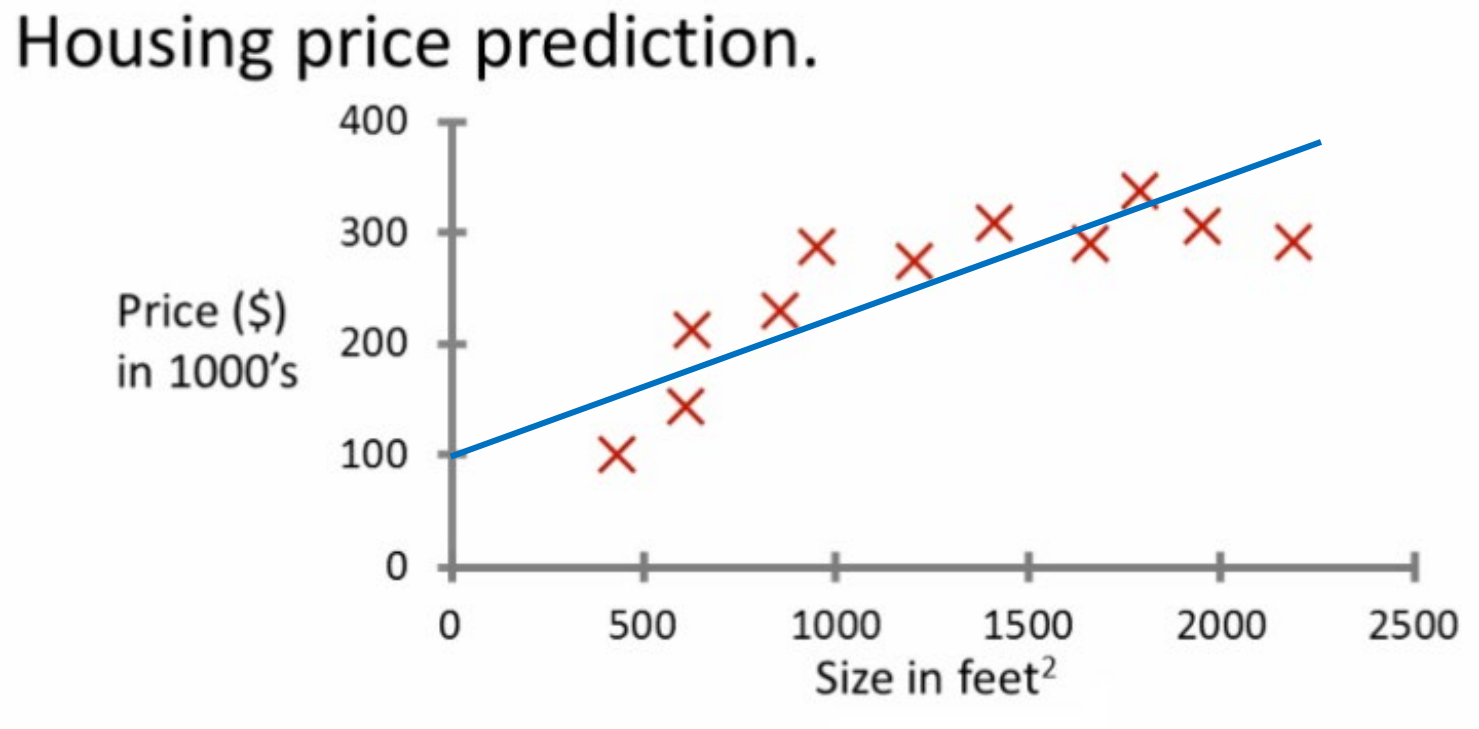
Linear Regression

- Linear regression captures a **linear relationship** between:
  - An **outcome variable** y (or called response variable, dependent variable, label) and
  - One or more **predictors** $x_1, \dots, x_D$ (or called independent variable, explanatory variable, feature)
- At its most basic, the relationship can be expressed as a *line*:

$$y = f(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_D x_D = \boldsymbol{\beta} \cdot \boldsymbol{x}$$

where $\boldsymbol{x} = [x_0, x_1, \dots, x_D], x_0 = 1$

Linear Regression



Linear Regression

- Linear functions capture changes in one variable that correlate linearly with changes in another variable.
- A simple assumption! They are less descriptive than non-linear functions, but permit simpler mathematical strategies.
- For some variables, this assumption makes sense.

  Example: The more umbrellas you sell, the more money you make. How much money you make is directly proportional to how many umbrellas you sell.

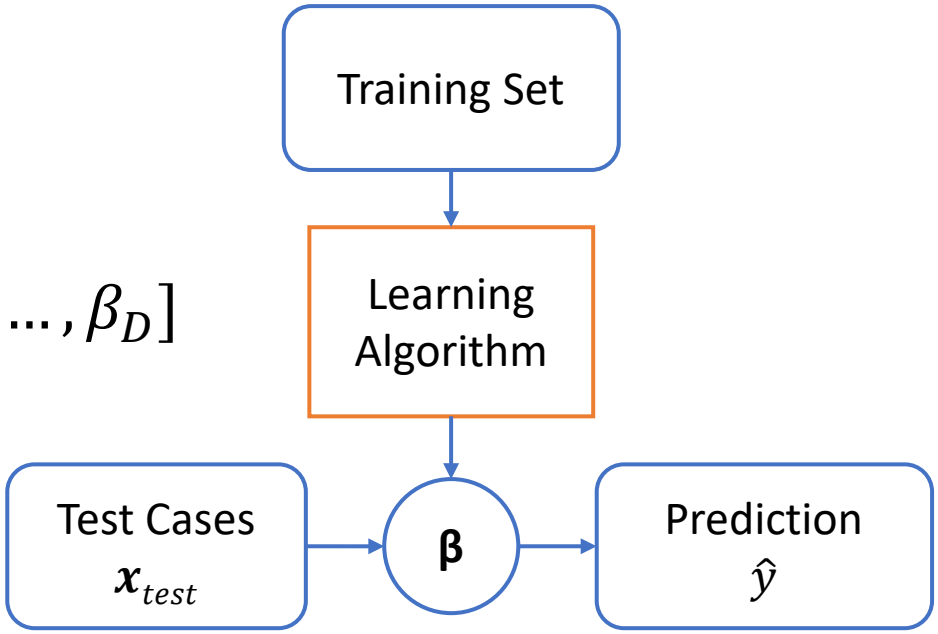
Training and Prediction

Training set: *N* examples
$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N),$
$\boldsymbol{x}_i = [x_{i0}, x_{i1}, x_{i2}, \ldots, x_{iD}]$

Find the optimal $\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2, \ldots, \beta_D]$

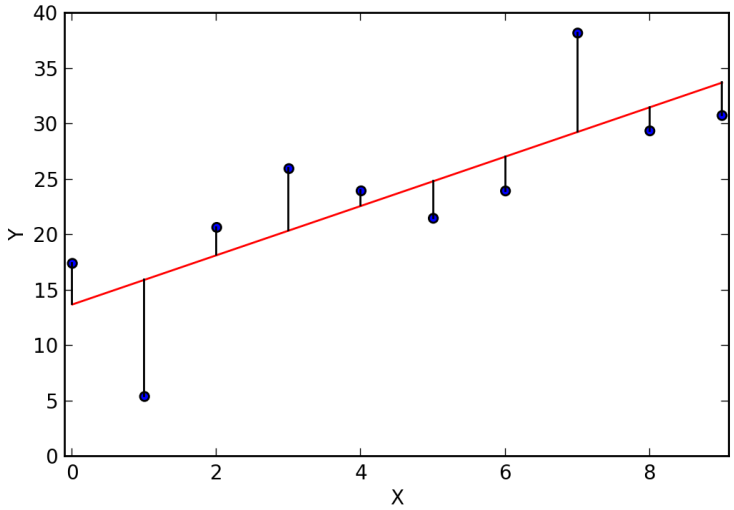
Predict a continuous valued output

$$\hat{y} = \boldsymbol{\beta} \cdot \boldsymbol{x}_{test}$$



Fitting the Model

- We want to choose the *best line*
  - Operationally, the line that minimises the *distance* between all points and the line.
  - Recall Euclidean distance: $d(A, B) = \sqrt{\sum_{k=1}^{D}(a_k - b_k)^2}$



Fitting the Model

- **Least squares method**: find the line that minimises the sum of the squares of the vertical distances between predicted $\hat{y}$ and actual y.
- **Minimise** the Mean Squared Error (MSE) of N data points

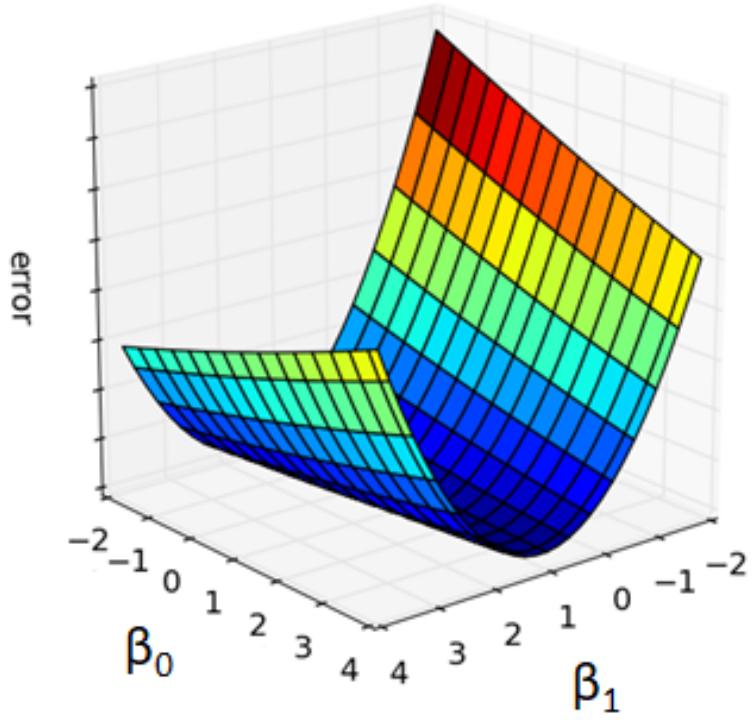
$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{\beta} \cdot \boldsymbol{x}_i)^2$$

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \mathrm{MSE}$$

loss function

Fitting the Model

- Good news! MSE is convex, the local minimum is a global minimum



Fitting the Model

- **Solution:** calculate partial derivatives of the loss function, with respect to $\boldsymbol{\beta}$, for *N* instances and *D* attributes
- Set $\frac{\partial}{\partial \beta_0}$ and all $\frac{\partial}{\partial \beta_k}$ to 0, and solve D+1 equations with D+1 unknowns

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{N} \sum_{i=1}^{N} (y_i - \boldsymbol{\beta} \cdot \boldsymbol{x}_i)^2$$

$$\frac{\partial}{\partial \beta_0} = -\frac{2}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i) \qquad \frac{\partial}{\partial \beta_k} = -\frac{2}{N} \sum_{i=1}^{N} x_{ik} (y_i - \hat{y}_i)$$

Parameter Estimation

- Optimisation
- Computing the Optimal Solution
- Gradient Descent

Optimisation

- Learning can be viewed as an **optimisation** problem
- Given an evaluation metric M (like Accuracy or Error), a dataset T, a feature representation $F(T)$, and a learner L with parameters θ
  - Maximise or minimise $M\big(\theta; L, F(T)\big)$, where learner L and dataset $F(T)$ are fixed
  - If M is error rate,

$$\hat{\theta} = \arg\min_{\theta \in \Theta} Error\ (\theta; L, F(T))$$

- Learning => finding parameters that optimise some criterion M for model L and feature representation $F(T)$

Computing the Optimal Solution

How to find the optimal solution?

$$\hat{\theta} = \arg\min_{\theta \in \Theta} Error\ (\theta; L, F(T))$$

- Analytic solution
- Exhaustive solution
- Grid search
- Iterative approximation

Computing the Optimal Solution

$$\hat{\theta} = \arg\min_{\theta \in \Theta} Error\ (\theta; L, F(T))$$

Analytic Solution

- closed form, can be computed exactly
- requires solving a system of equations

$$\frac{\partial(Error)}{\partial\theta_1} = 0, \frac{\partial(Error)}{\partial\theta_2} = 0, \dots, \frac{\partial(Error)}{\partial\theta_D} = 0$$

- e.g. for linear regression, $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$
- challenges: derivatives can be undefined or not calculable

Computing the Optimal Solution

$$\hat{\theta} = \arg\min_{\theta \in \Theta} Error\ (\theta; L, F(T))$$

Exhaustive solution

- For each $\theta \in \{k_1, k_2, \ldots, k_n\}$, calculate $Error\ (\theta; L, F(T))$
- For M parameters, each taking N **unique** values, it requires N^M train-evaluate cycles
- Works for methods with a small number of hyperparameters, taking a small number of values, e.g. KNN
  - Similarity measure: Euclidean, Manhattan, cosine,…
  - Voting strategy: majority, ID, ILD,…
  - Number of neighbours: 1, 2, 3, …

Computing the Optimal Solution

$$\hat{\theta} = \arg\min_{\theta \in \Theta} Error\ (\theta; L, F(T))$$

Grid Search

- For each numerical $\theta \in R$
  - Identify boundaries of range R_-, R_+
  - Divide range $[R_-, R_+]$ into equal-width samples
  - calculate $Error\ (\theta; L, F(T))$ for each sample
- More samples: closer to optimal estimate, but more time required

Computing the Optimal Solution

$$\hat{\theta} = \arg\min_{\theta \in \Theta} Error\ (\theta; L, F(T))$$

Iterative approximation

1. **Initialisation**: set $iter = 0$, guess θ^0
2. **Evaluate**: compute $Error\ (\theta^{iter}; L, F(T))$
3. **Termination**: decide whether to stop
4. **Update**: compute θ^{iter+1} based on θ^{iter}; increase

$$iter := iter + 1$$

5. **Repeat**: Go to Step 2

Gradient Descent

$$\hat{\theta} = \arg\min_{\theta \in \Theta} Error\ (\theta; L, F(T))$$

Gradient Descent

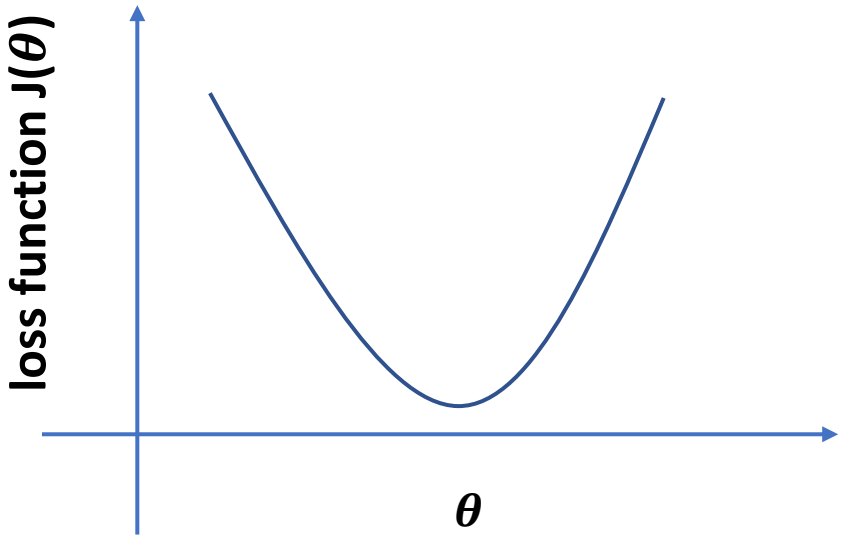
$$\theta^{iter+1} := \theta^{iter} - \alpha \nabla Error(\theta^{iter}; L, F(T))$$

$$\theta_k^{iter+1} := \theta_k^{iter} - \alpha \frac{\partial\ Error(\theta_k^{iter})}{\partial \theta_k^{iter}}$$

- Find the direction: $\nabla Error(\theta)$ is a vector of partial derivative terms, which measures the slope (=gradient) of the function
- Each update reduces the error slightly

Gradient Descent

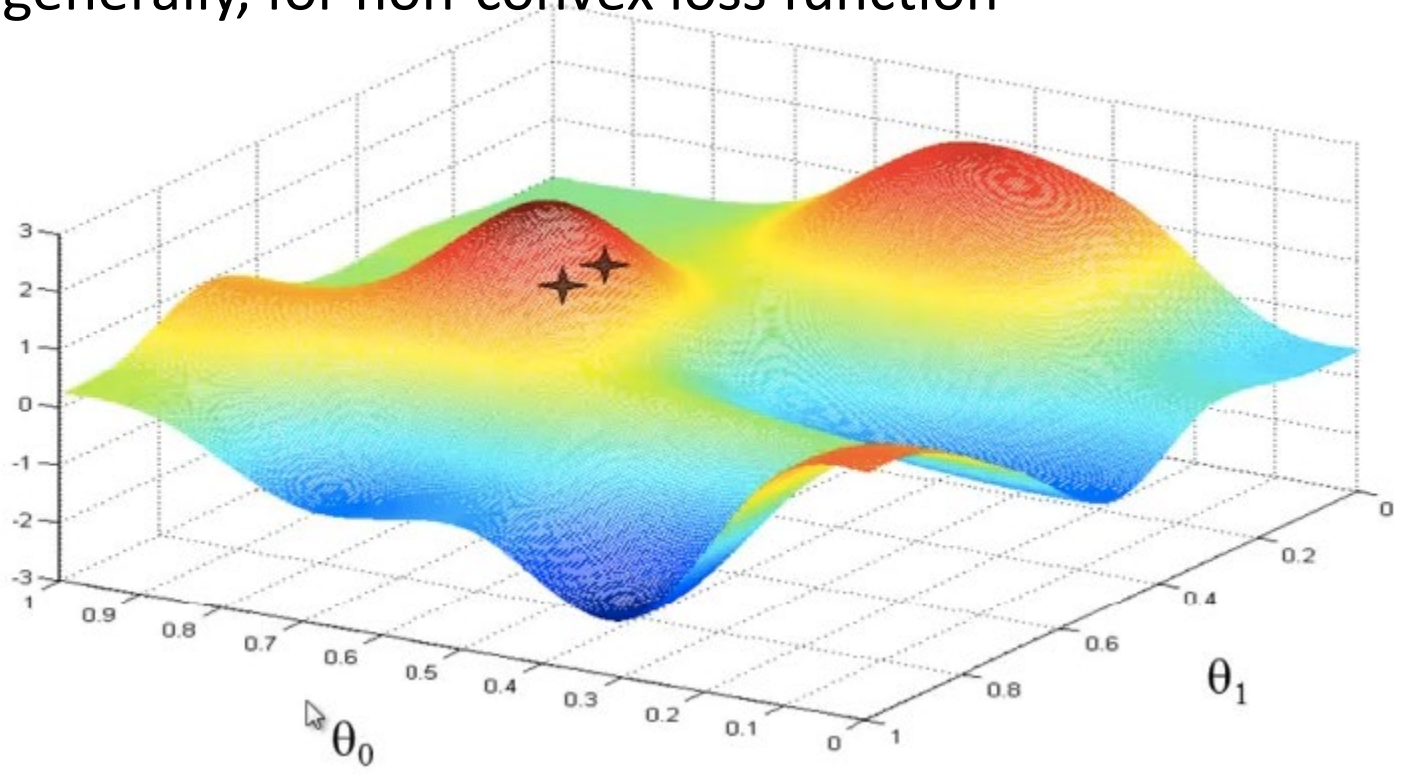
- α is learning rate, which defines how big a step to update θ_k^{iter}
  - If α is too small, the algorithm might be slow.
  - If α is too large, you might miss the minimum.



Gradient Descent

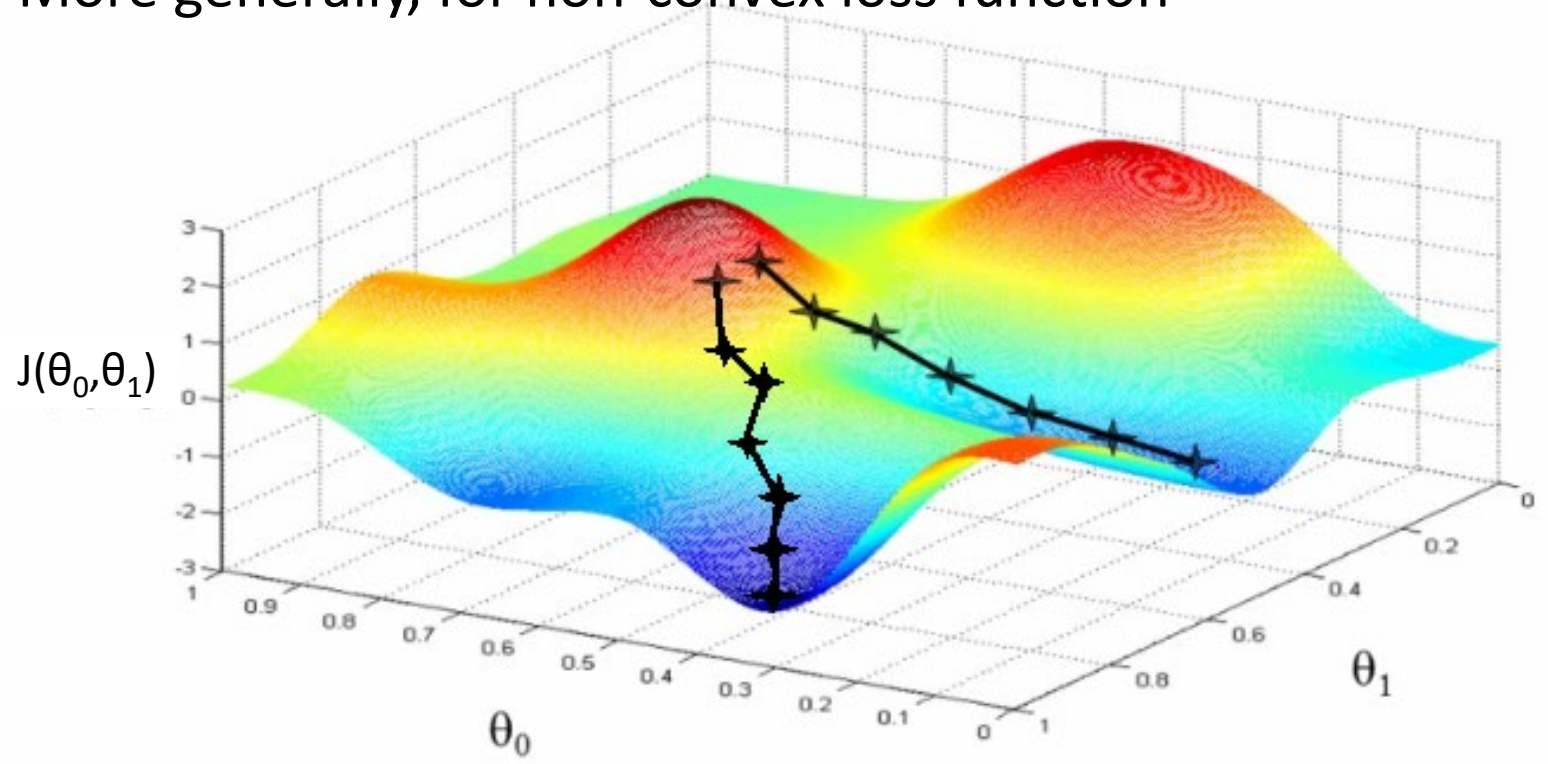
More generally, for non-convex loss function

$J(\theta_0,\theta_1)$



Gradient Descent

More generally, for non-convex loss function



Gradient Descent

For linear regression

$$\beta_k^{iter+1} := \beta_k^{iter} - \alpha \frac{\partial\, Error(\beta_k^{iter})}{\partial \beta_k^{iter}} = \beta_k^{iter} + \frac{2\alpha}{N} \sum_{i=1}^{N} x_{ik}(y_i - \widehat{y_i^{iter}})$$

Gradient descent steps for iteration $iter$

- Calculate prediction $\widehat{y_i^{iter}}$ for each training instance
- Compare prediction with actual value y_i
- Multiply by the corresponding attribute value x_{ik}
- Update weight β_k after all the training instances are processed

Application and Evaluation

Linear Regression Application

- Example on a dataset with more features

PRP = - 56.1 + 0.049 MYCT+ 0.015 MMIN+ 0.006 MMAX+ 0.630 CACH- 0.270 CHMIN+ 1.460 CHMAX

Table: CPU dataset

MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	PRP
125	256	6000	256	16	128	199
29	8000	32000	32	8	32	253
29	8000	32000	32	8	32	253
29	8000	32000	32	8	32	253
29	8000	16000	32	8	16	132
26	8000	32000	64	8	32	290
23	16000	32000	64	16	32	381
…	…	…	…	…	…	…

Categorical Attributes

- We can easily map nominal attributes onto numeric attributes through binarisation or one-hot encoder
- If we treat each resulting binary feature as continuous, we can use linear regression as it is

Evaluation

- It doesn't make sense to evaluate numeric prediction tasks in the same manner as classification tasks
  - true positive matches (direct hits) are an unreasonable expectation
  - regression can make use of the inherent ordering and scale of the outputs
- There are many scoring metrics for regression tasks, all of which are based on the **absolute or relative difference** between the predicted value $\hat{y}_i$ and actual y_i value of test instances

Evaluation Metrics

- Mean Squared Error (MSE):

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- Root Mean Squared Error (RMSE):

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

- Root Relative Squared Error (RRSE): relative to baseline $\bar{y} = \frac{1}{N}\sum y_i$

$$\sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}}$$

Evaluation Metrics

- Correlation Coefficient (Pearson's correlation): statistical correlation between predicted and actual values

$$r = \frac{S_{\hat{Y}Y}}{\sqrt{S_{\hat{Y}} S_Y}}$$

$$S_{\hat{Y}Y} = \frac{\sum_i (\hat{y}_i - \bar{\hat{y}})(y_i - \bar{y})}{N - 1}$$

$$S_{\hat{Y}} = \frac{\sum_i (\hat{y}_i - \bar{\hat{y}})^2}{N - 1}, S_Y = \frac{\sum_i (y_i - \bar{y})^2}{N - 1}$$

Evaluation Metrics

- Which metric to use?

  The relative ranking of methods across the different metrics is reasonably stable, such that the actual choice of metric isn't crucial

	A	B	C	D
RMSE	67.8	91.7	63.3	**57.4**
RRSE	42.2	57.2	39.4	**35.8**
CC	0.88	0.88	0.89	**0.91**

Beyond Linear Relationship

- Linear regression is an intuitive model, with relatively easy implementation
- But can we really expect a linear relationship between feature values and target variables?
- Other regression methods
  - regression trees
  - locally weighted linear regression
  - support vector regression
  - …

Beyond Linear Relationship

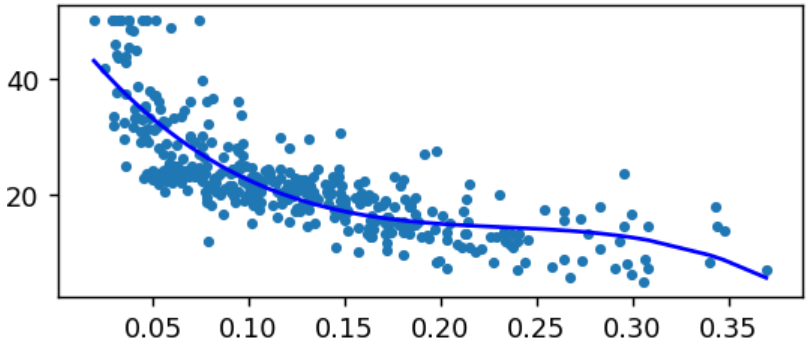
- Polynomial of x as input variables

$$\overrightarrow{\varphi(x)} = [1, x, x^2, x^3, \dots, x^D]$$

$$\mathbf{z} = [z_0, z_1, z_2, z_3, \dots, z_D]$$

$$\hat{y} = \boldsymbol{\beta} \cdot \boldsymbol{z}$$

- Can capture nonlinear relationship regarding x



Summary

- What is linear regression, and how does it operate?
- How is learning a type of optimisation?
- What is gradient descent, and why is it used for linear regression?
- How can we evaluate regression tasks?