Linear Models

CM307 Session 4

Yuhua Li
liy180@cardiff.ac.uk

Contents

- Linear regression
- Model training
  - Ways of training
    - Closed-form solution
    - Gradient descent
  - Controlling model complexity
- Polynomial regression
- Regularized linear models
- Logistic regression

Machine Learning Approaches: recap

- Supervised learning: learning with a labelled training dataset.
  For example, malware detection with training set of already labelled software applications.
  - Classification
  - Regression
- Unsupervised learning: discovering patterns in unlabelled dataset.
  For example: clustering user groups on social network based on their shared interests.
  - Clustering
  - Anomaly detection
- Reinforcement learning: learning based on feedback or reward
  For example: learning to play chess by wining or loss

Supervised Learning

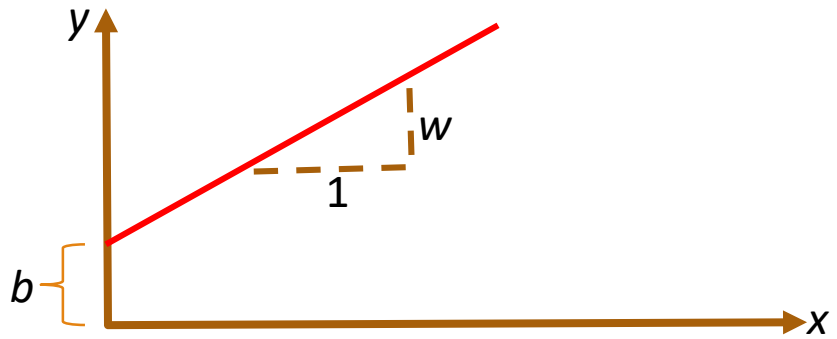
- Regression
  - Linear regression, polynomial regression
  - Ridge regression, LASSO, elastic net
  - Fuzzy logic systems (FL)
  - Artificial neural networks (ANN) / deep learning
  - …
- Classification
  - Logistic regression
  - Support vector machines (SVM)
  - Bayesian network
  - K nearest neighbours (K-NN)
  - Decision tree (DT)
  - Ensemble learning
  - …
- Note many techniques work for both classification and regression problems, e.g., FL, ANN, k-NN, DT, SVM.

Linear Regression

- A linear model is a sum of weighted variables that predicts a target output value given an input data instance.

$$y = wx + b$$

w – slope/gradient, b – intercept

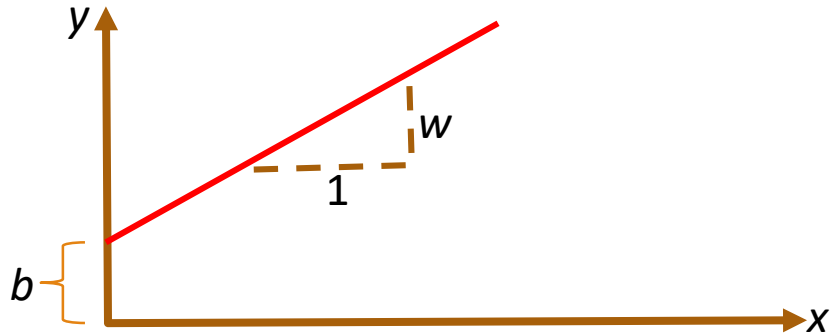


Linear Regression

- A linear model is a sum of weighted variables that predicts a target output value given an input data instance.

$$y = wx + b$$

w – slope/gradient, b – intercept

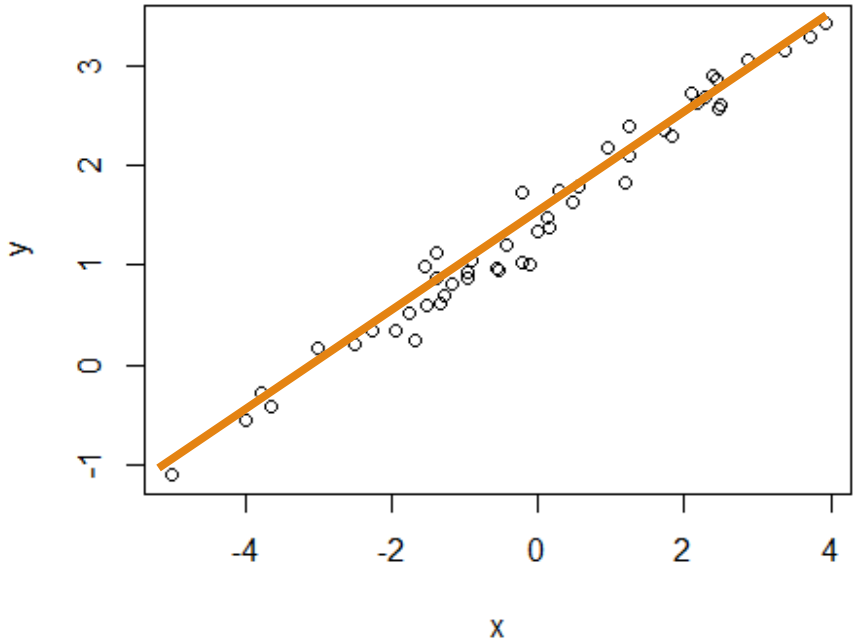


- Linear regression is to find a linear model for a given data set. In general form

$$y = w_1 x_1 + w_2 x_2 + \cdots w_n x_n + b$$

$$= \mathbf{w}^T \mathbf{x} + b$$

$w_1, w_2, \cdots w_n$ – feature weights, b – bias



Least-Squares Linear Regression

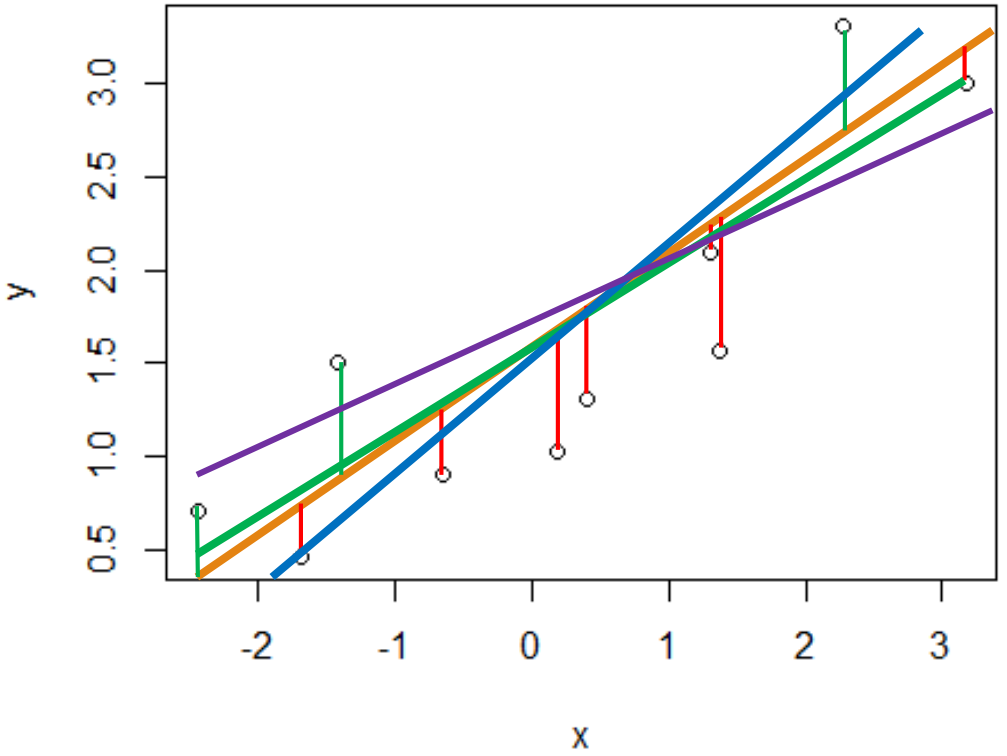
- Finds *w* and *b* that minimizes the mean squared error (MSE) of the linear model: the average of the sum of squared differences between predicted target $\hat{y}$ and actual target *y* values

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

$$= \frac{1}{N}\sum_{i=1}^{N}((\mathbf{w}^T\mathbf{x}_i + b) - y_i)^2$$

L refers to the objective function (aka, loss function, cost function, error function) of the model

- Finding the optimal model parameters $\boldsymbol{w} = (w_1, w_2, \cdots w_n)^T$ and b on a given dataset is called training.



Training of Linear Regression

- Parameters of a linear regression model can be set by training the mode to best fits the training dataset
- Two very different ways to train linear regression model
  - Using a direct “closed-form” equation through mathematical derivation → exact solution
  - Using an iterative optimization approach, called Gradient Descent (GD) → approximate solution

Linear Regression Training - closed form

- Include *b* as a parameter, $w_0 = b$, i.e., $\boldsymbol{w} = (w_0, w_1, w_2, \cdots w_n)^T$, we can rewrite the equation

$$\hat{y} = \mathbf{w}^T \mathbf{x}$$

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

$$= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{w}^T \mathbf{x}_i - y_i)^2$$

- Parameters that minimize $L(w)$ can be obtained by setting its gradient to zero:

$$\nabla_w L(w) = 0$$

gives: $\hat{\boldsymbol{w}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$

The above is called normal equation, where each row of *X* is a point, *y* is the column vector of target values

- Computational complexity

  Involve matrix inverse with complexity of $O(n^{2.4})$ to $O(n^3)$,

  where *n* is the number of features

- So computation is slow when *n* is large

Linear Regression Training – Gradient Descent

- Gradient Descent is to tweak parameters iteratively in order to minimize a cost function.
- Procedure of gradient descent

  Initialise parameters w with random values

  Repeat

    Calculate current value of the loss function $L(w)$

    Calculate gradient of the loss function at current step $\nabla_w L(w)$

    Update w using current gradient

$$w^{next} = w^{current} - \eta \nabla_w L(w)$$

  until satisfying termination condition

where η is the learning rate which determine the step size of parameter updating

$$\nabla_w L(w) = \frac{2}{N} X^T \cdot (X \cdot w - y)$$

Linear Regression Training – Gradient Descent

- As the loss function (MSE) of linear regression is convex, it doesn't have local minimum. Using Gradient Descent for training linear regression is guaranteed to approach arbitrarily close the global minimum.
- Gradient Descent scales well with the number of features; training a Linear Regression model when there are hundreds of thousands of features is much faster using Gradient Descent than using the Normal Equation.
- A round of training over the whole training set is called an epoch.

Gradient Descent: Tips

- Data normalization
  - When using Gradient Descent, you should ensure that all features have a similar scale
  - Normalization: scaled to fall within a small, specified range such as –1.0 to 1.0, 0.0 to 1.0 or $\mathcal{N}(0, 1)$
  - min-max normalization
    - For a variable X, $x \in X$, the range of X is is normalized from its original range $[min_X, max_X]$ to new range $[new\_min_X,\ new\_max_X]$

$$x' = \frac{new\_max_X - new\_min_X}{max_X - min_X}(x - min_X) + new\_min_X$$

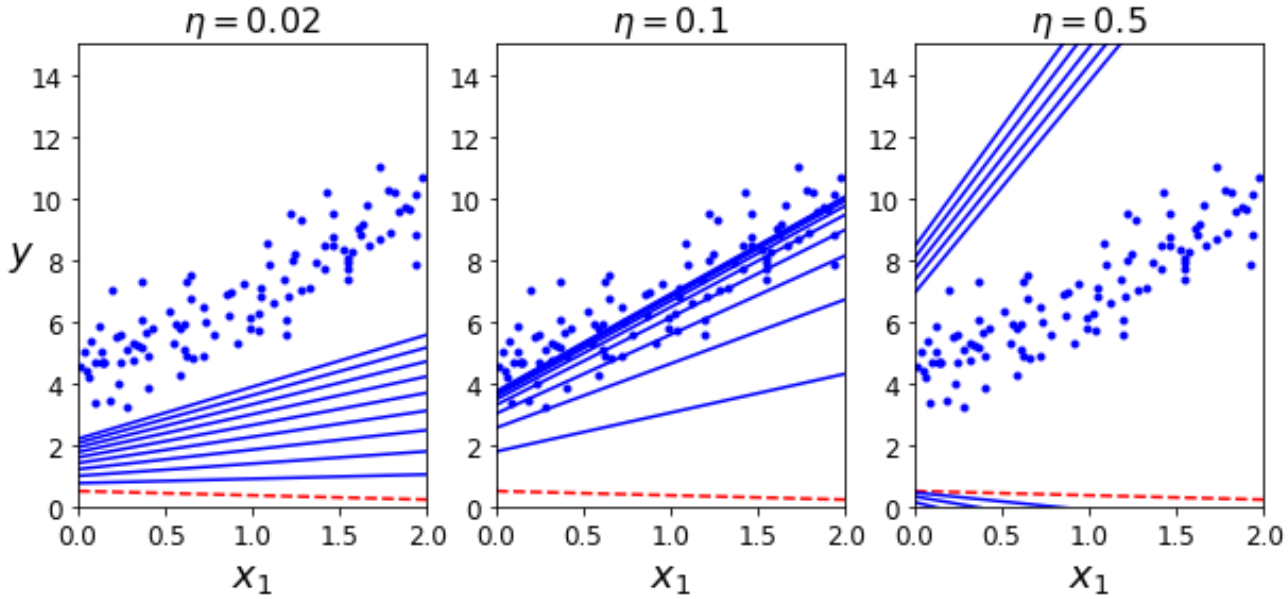
E.g., Normalising dataset (9, 22, 14, 2, 11) to new range [-1, 1] becomes (-0.3 1.0 0.2 -1.0 -0.1)

  - z-score normalization

$$x' = \frac{x - mean_X}{std_X}$$

Gradient Descent: Tips

- Leaning rate
  - Too small: take long time to reach the solution
  - Too big: may not converge to the solution, overshot

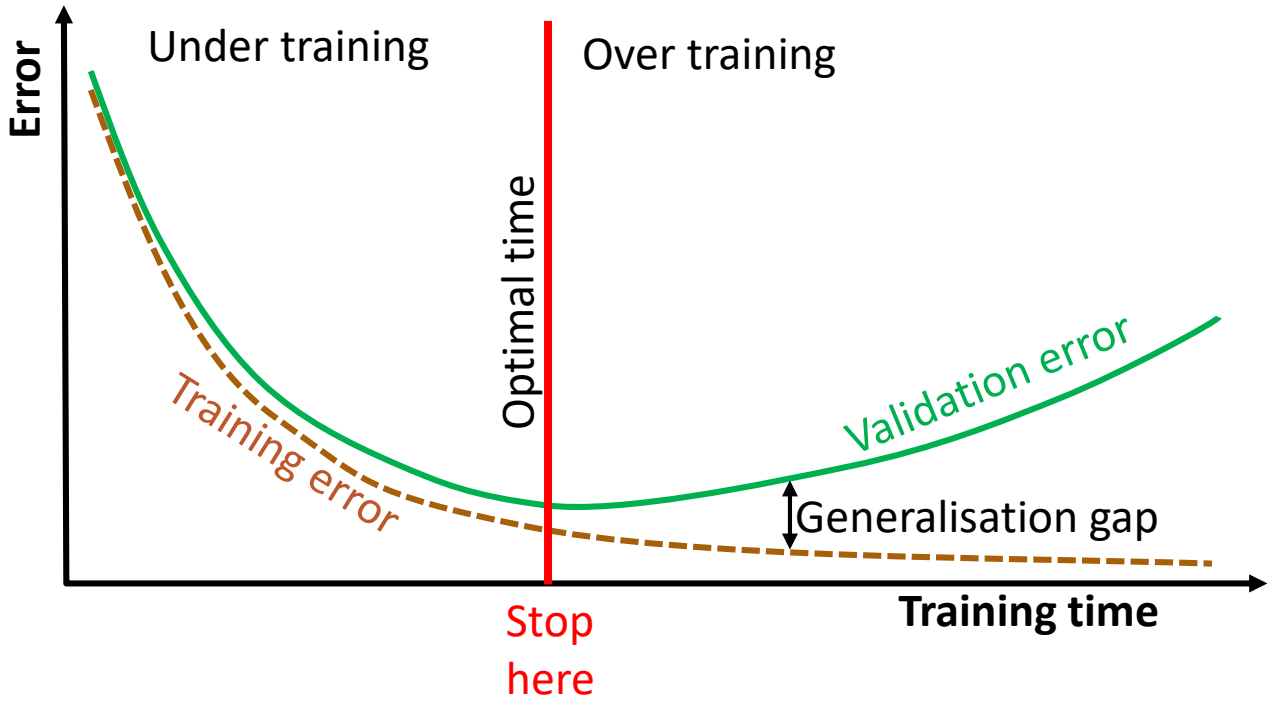


  - Learning schedule: use a function to update learning rate, i.e., start with a large learning rate and gradually decrease it at each step

$$\text{e.g., } \eta(t) = \frac{a}{b + t}$$

Gradient Descent: Tips

- Early stopping
  - Training can be terminated if the value of the loss function falls below pre-set values or the number of repetitions reaches the maximum number of iterations.
  - Use a validation set, stop training when the performance on validation set start decreasing or no improvement after a set number of iterations.

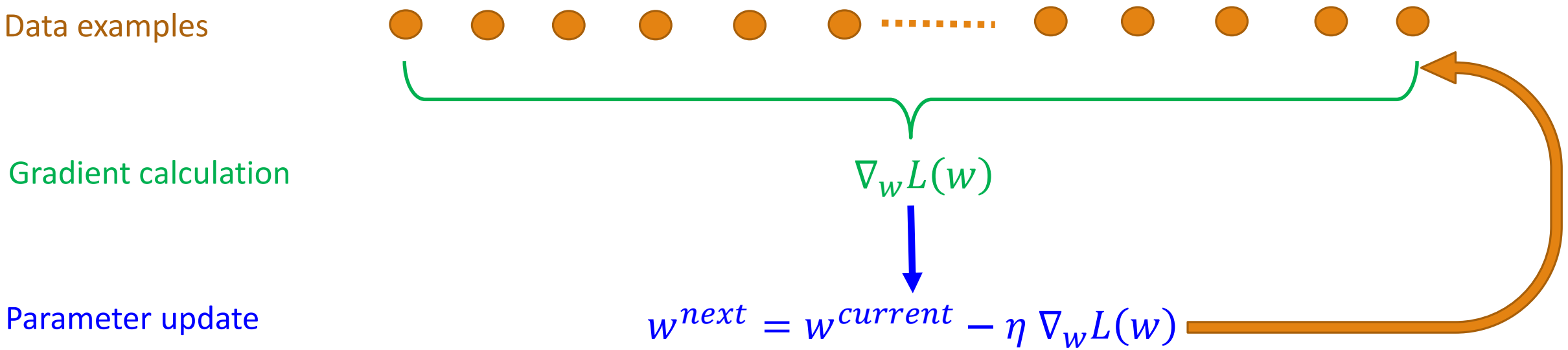


Ways of gradient decent

- There are 3 ways to calculate gradient of the cost function and update parameters depending how often this is done
  - Batch Gradient Descent
  - Stochastic Gradient Descent
  - Mini-batch Gradient Descent

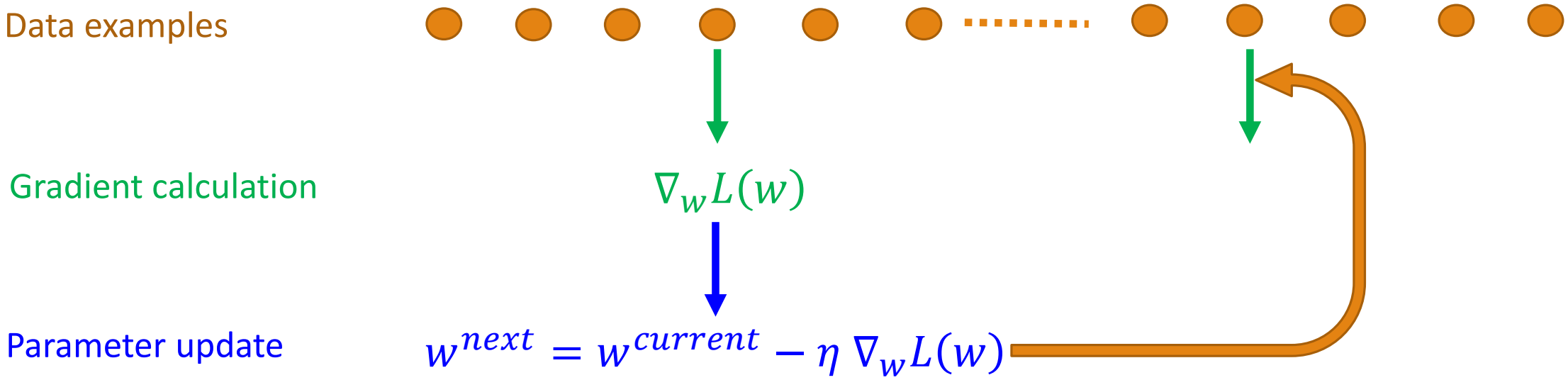
Batch Gradient Descent

- uses the whole batch of training dataset X to calculate the cost function and its gradient at every step
- Slow updating when training set is large



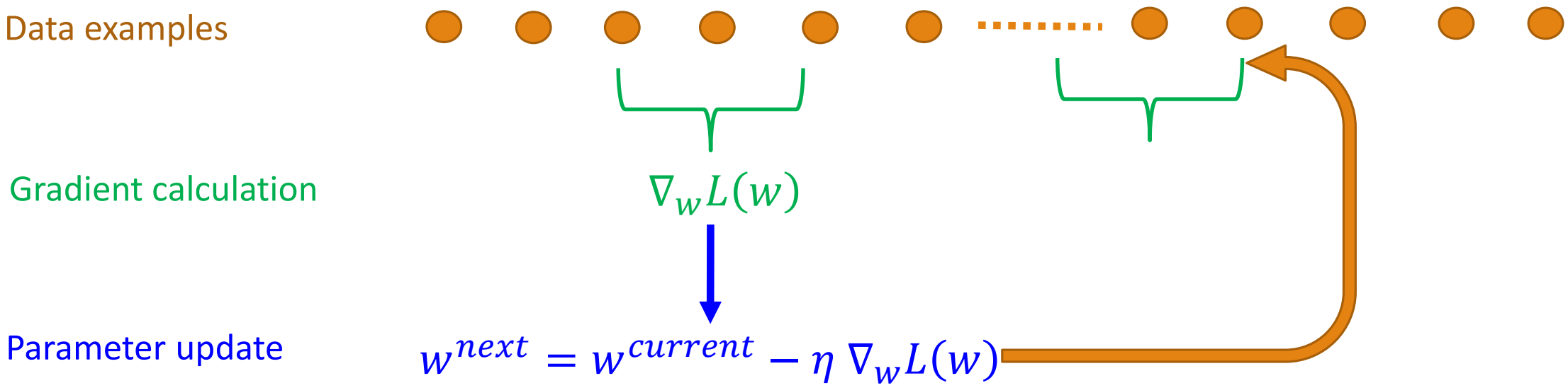
Stochastic Gradient Descent

- Picks a random instance in the training set at every step and computes the gradients based only on that single instance.
- Frequent updating



Mini-batch Gradient Descent

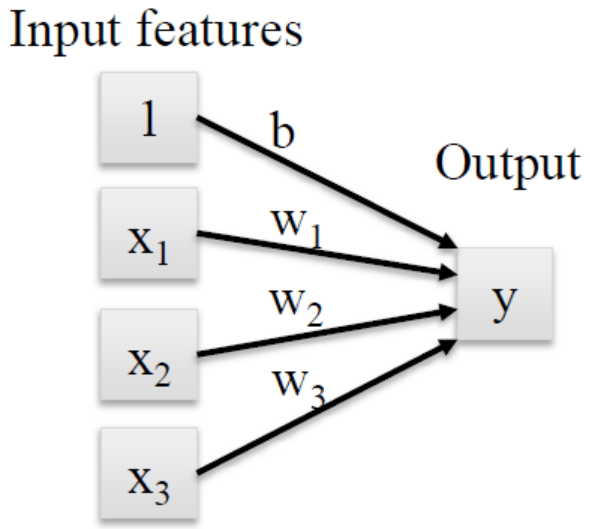
- At each step, instead of computing the gradients based on the full training set (as in Batch GD) or based on just one instance (as in Stochastic GD), Mini-batch GD computes the gradients on small random sets of instances called mini-batches.



Computational graph of linear regression

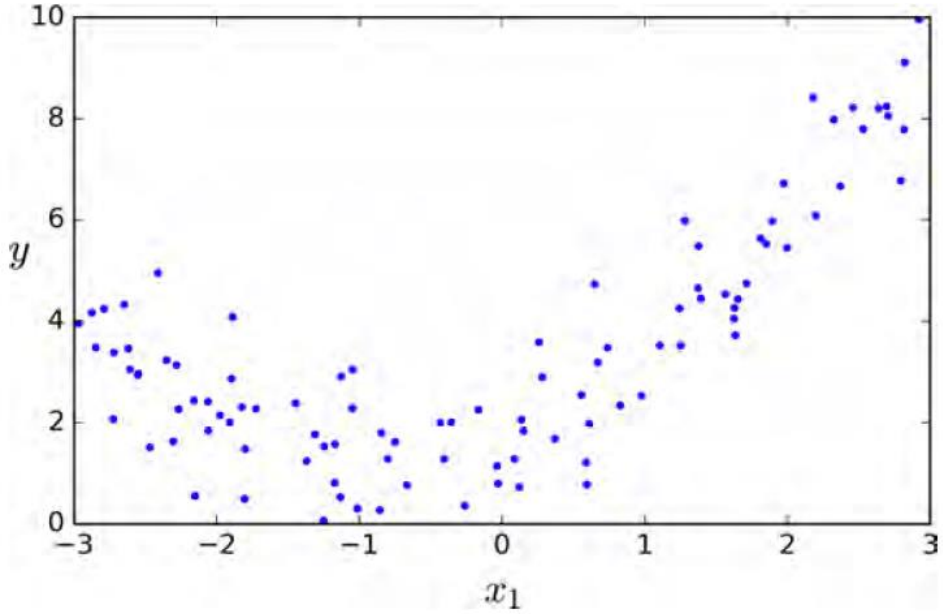
- $y = w_1 x_1 + w_2 x_2 + \cdots w_n x_n + b$
- Rewrite as

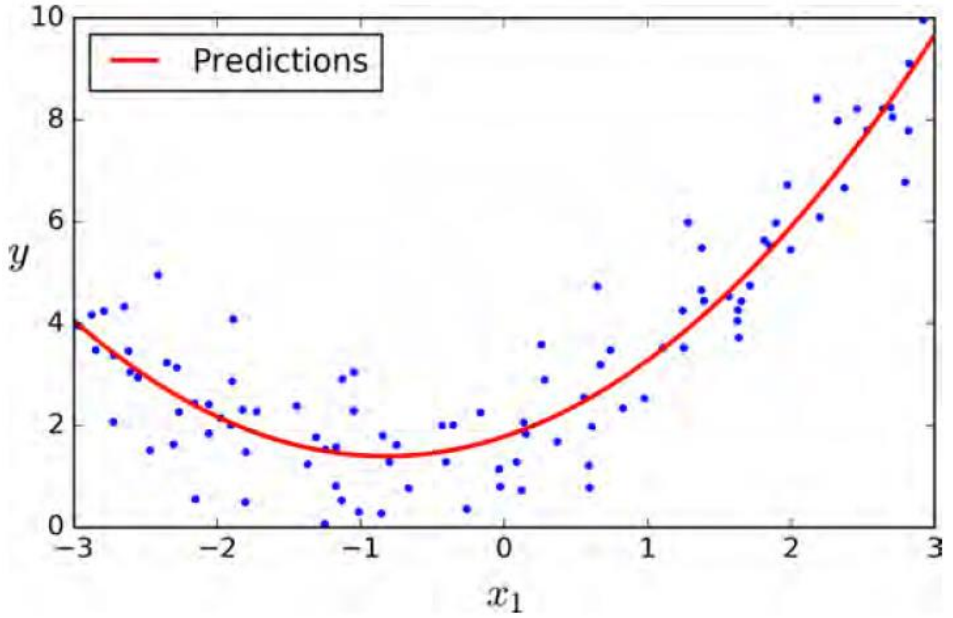
$$y = b\,1 + w_1 x_1 + w_2 x_2 + \cdots w_n x_n$$



Polynomial regression

- Add powers of each feature (and multiplications of features) as new features, then train a linear model on this extended set of features





e.g., for 2nd degree of univariate: $y = w_0 + w_1 x_1 + w_2 x_1^2 + N(\mu, \sigma^2)$

In particular, for the figures above: $y = 2 + 1x_1 + 0.5x_1^2 + N(0, 1)$

So x_1^2 can be consider a new feature. In fact, each introduced term of polynomial can be treated as a new feature in ordinary linear regression. Thus training methods of ordinary linear regression can be used for training of polynomial regression.

Figs courtesy of Géron 2017.

Polynomial regression

- Which degree?

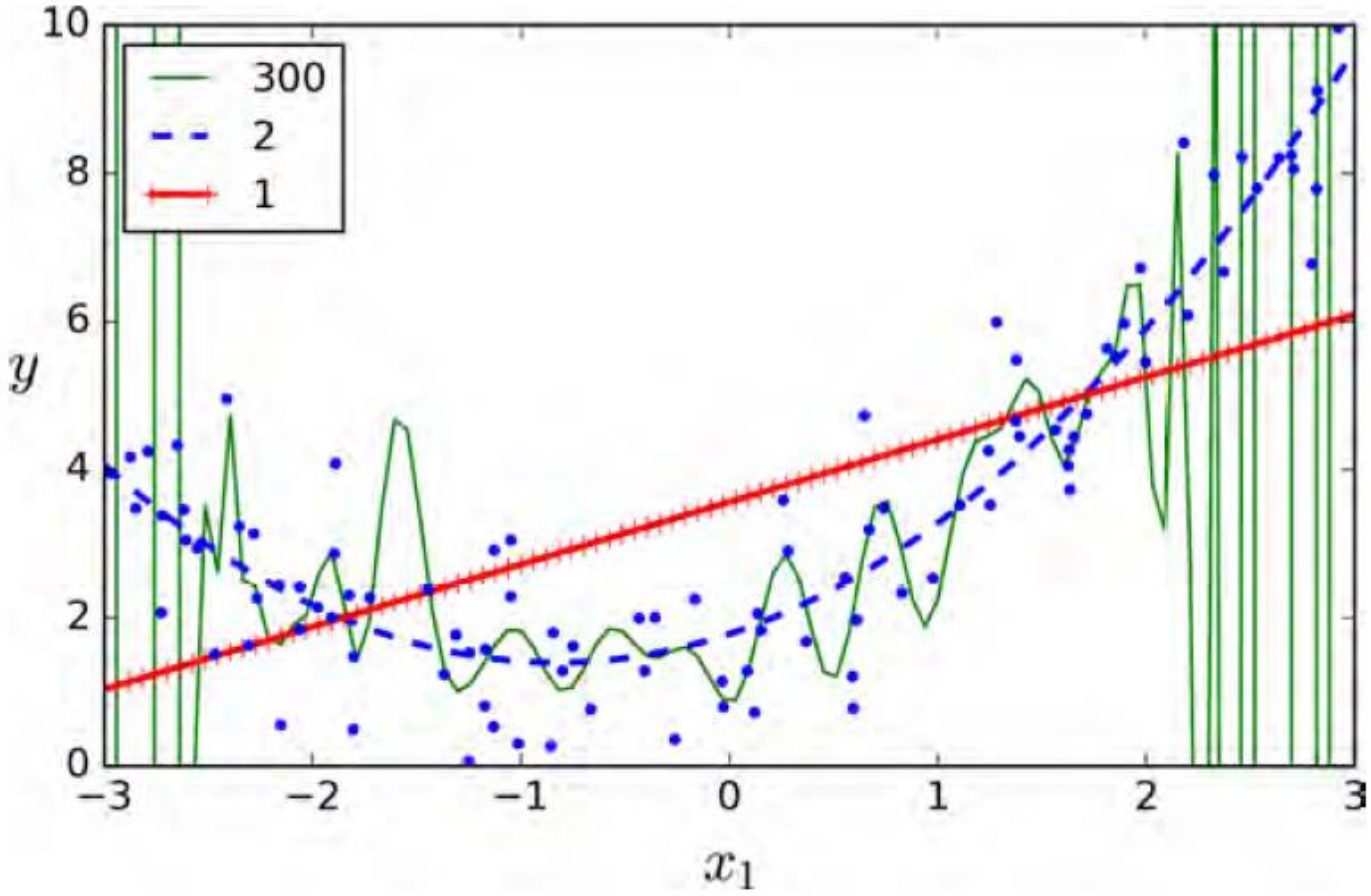


Fig courtesy of Géron 2017.

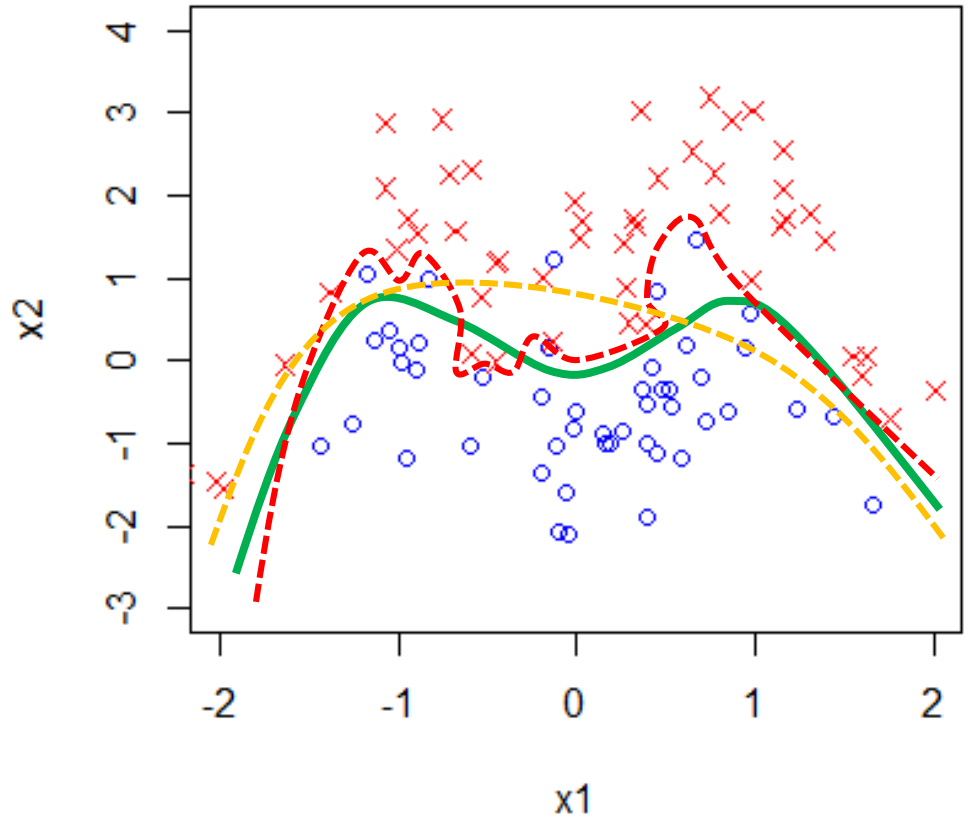
Important Issues in Machine Learning

- Generalisation, overfitting, underfitting
- Regularisation

Recap: Generalisation, overfitting, underfitting

- Generalisation is the ability to perform well on previously unseen data.

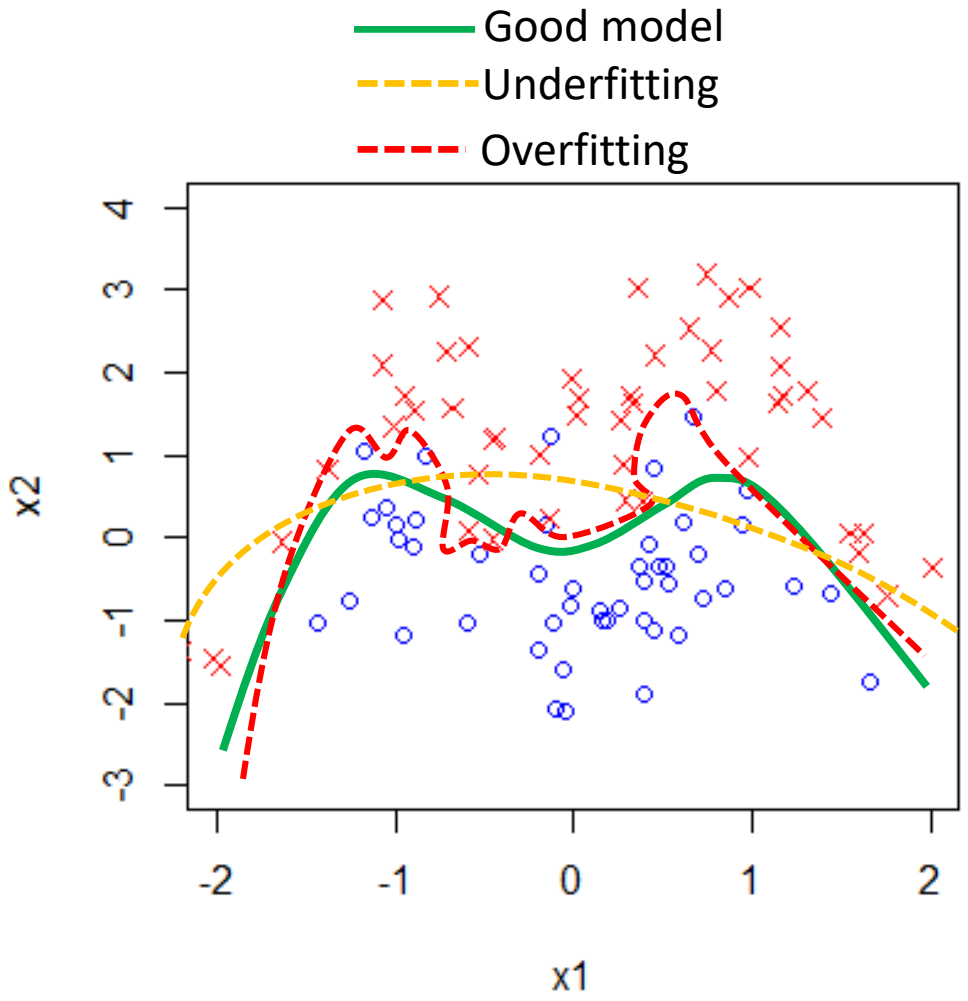
- The factors determining how well a learning algorithm will perform are its ability to
  - Make the training error small.
  - Make the gap between training and test error small.



Generalisation, overfitting, underfitting

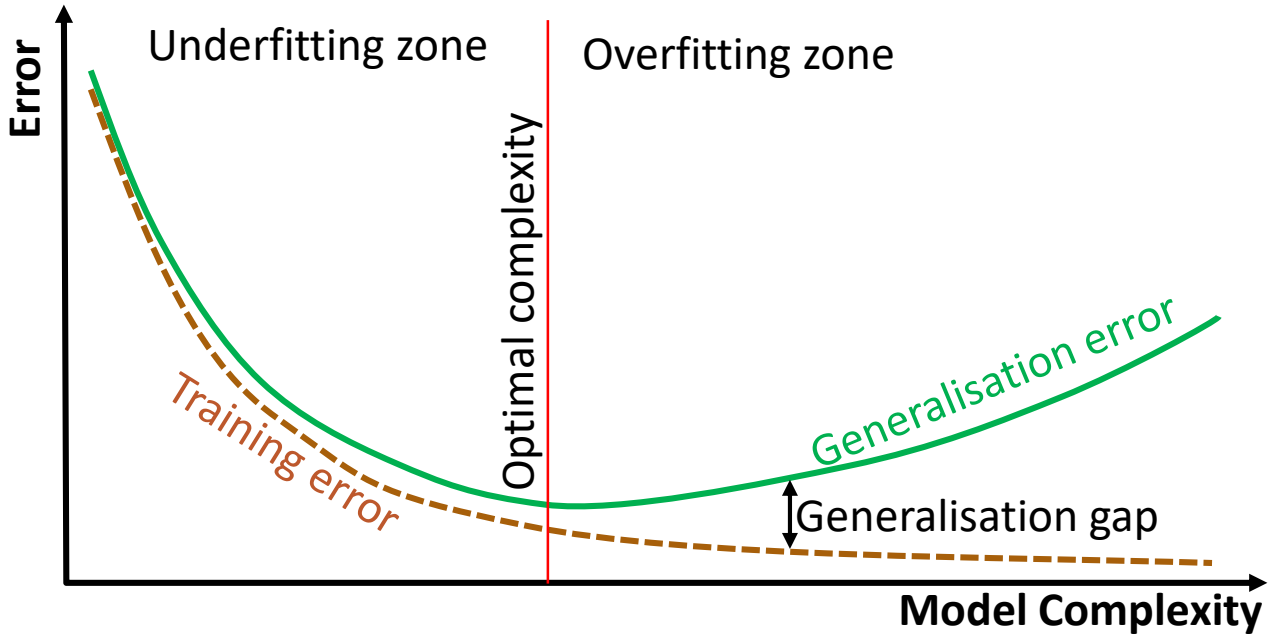
- Underfitting occurs when the model is not able to obtain a sufficiently low error on the training set.

- Overfitting occurs when the gap between the training error and test error is too large.



Generalisation, overfitting, underfitting

- Typical relationship between model complexity and error



Overfitting

- Overfitting symptom
  - High accuracy on training data, low accuracy on unseen data

- Ways to deal with overfitting
  - regularisation,
  - more training data,
  - dimensionality reduction,
  - dataset shift → model adaptation.

Regularisation

- Regularisation is any modification to a learning algorithm that is intended to reduce its generalisation error but not its training error.

- Regularization prevents overfitting by restricting the model, typically to reduce its complexity.

- A common way of regularisation is to add a penalty term to its original loss function, with respect to model parameters.

Regularized Linear Models

- Ridge regression
- LASSO
- Elastic net

Ridge Regression

- Adds L2 penalty to the cost function of plain linear regression
  - Ridge regression learns *w*, *b* using the same least-squares criterion but adds a penalty for large variations in w parameters

$$L(w,b) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}_i - y_i)^2 \boxed{+\alpha\sum_{j=1}^{p} w_j^2}$$

- The parameter $\alpha > 0$ controls amount of regularization (default = 1.0). Larger values specify stronger regularization.
- This kind of parameters like α in above equation is called hyperparameters, which need to be set by users before training process.
- The same learning algorithms (closed form or gradient descent) can be used to find model parameters ***w***.

LASSO regression

- Adds L1 penalty

  LASSO (least absolute shrinkage and selection operator)

  - Minimize MSE and the sum of the absolute values of the parameters

$$L(w,b) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}_i - y_i)^2 \boxed{+\alpha\sum_{j=1}^{p}|w_j|}$$

  - This has the effect of setting parameter weights in *w* to zero for the least influential variables. This is called a sparse solution: a kind of feature selection.

Elastic Net

- Adds L1 and L2 penalty

  Elastic net plays in the middle between ridge regression and LASSO

$$L(w,b) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}_i - y_i)^2 \; +\gamma\,\alpha\sum_{j=1}^{p}|w_j| + (1-\gamma)\,\alpha\sum_{j=1}^{p}w_j^2$$

  where $\gamma \in [0,1]$ is L1 ration.

  When $\gamma = 0$, Elastic Net is equivalent to Ridge Regression

  When $\gamma = 1$, Elastic Net is equivalent to Lasso Regression

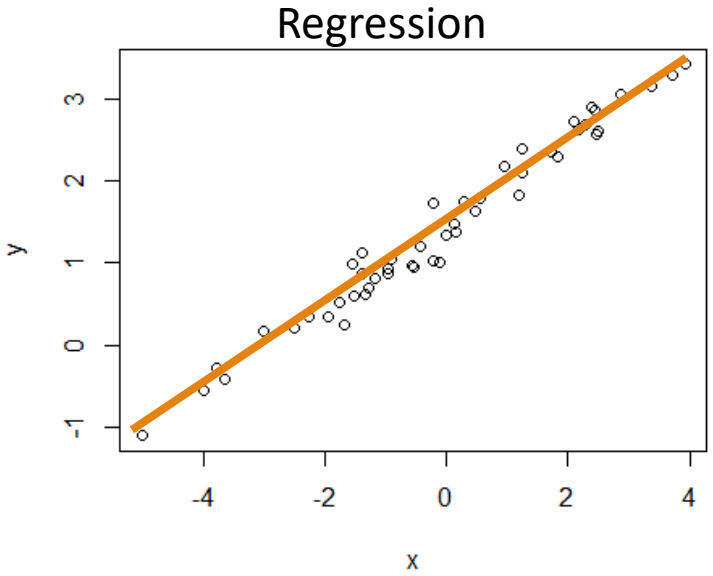
- Once the parameters are learned, the prediction formula for all ridge regression, LASSO regression and elastic net is the same as plain linear regression, i.e., $\hat{y} = \mathbf{w}^T\mathbf{x}$.

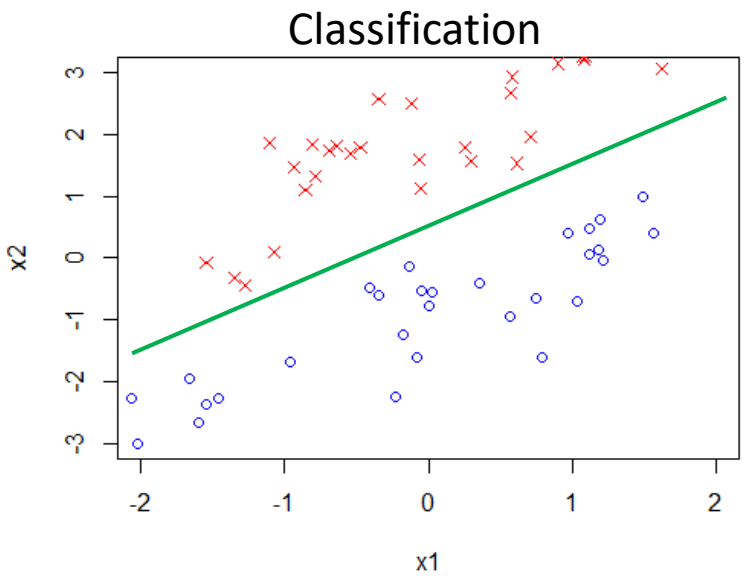
Which regression model?

- Model should have appropriate capacity to approximate the data, regularisation is an effective way to control model complexity.
- Generally model incorporating regularization achieve better performance, so plain linear regression (no regularisation) should be avoided.
- Use ridge regression if all features are useful.
- Use LASSO if you suspect some/many features are not useful, so those features could be eliminated.
- Use elastic net if no intuition about feature usefulness is obvious. However, there are 2 hyperparameters, α and γ, need to be decided.
- Introduce polynomial terms for data with non-linear characteristics.

Logistic Regression

- Regression vs. classification



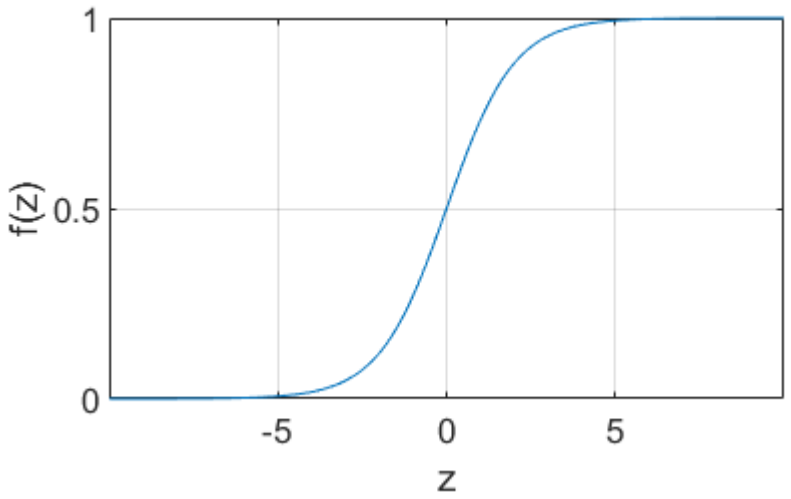


- Logistic regression is a linear model for classification
- It transforms the output of ordinary linear regression using logistic function, so given the name logistic regression

Logistic Regression

- Logistic function (aka sigmoid function)

$$f(z) = \frac{1}{1 + e^{-z}}$$



- Logistic Regression

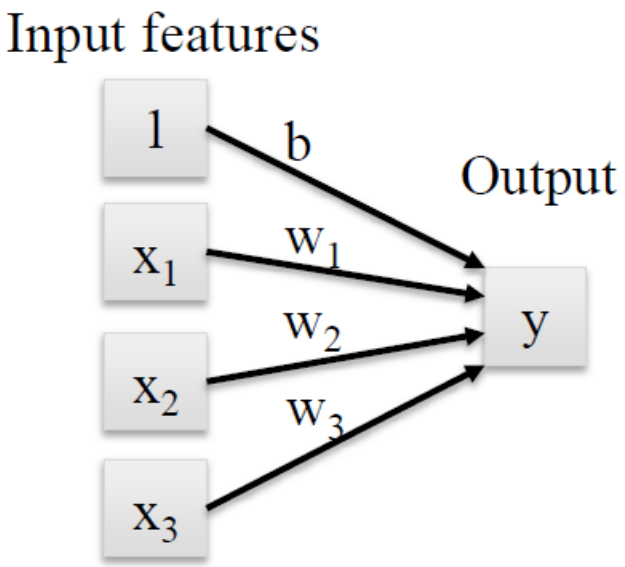
$$y = f(b + w_1 x_1 + w_2 x_2 + \cdots w_n x_n)$$

$$= \frac{1}{1 + e^{-(b + w_1 x_1 + w_2 x_2 + \cdots w_n x_n)}}$$

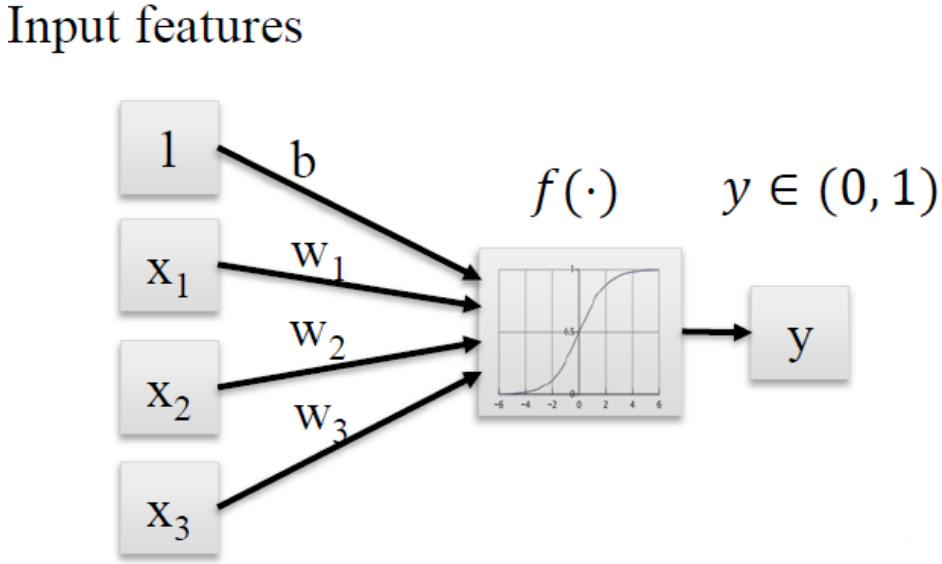
Logistic Regression

- The logistic function transforms real-valued input to an output number y between 0 and 1, interpreted as the probability the input object belongs to the positive class, given its input features $(x_1, x_2, \cdots x_n)$.
- If its output value is greater than a threshold (usually 0.5), then the object belongs to positive class, otherwise negative class.

Linear regression



Logistic regression



Logistic Regression Training

- Cost function (log loss)

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}log\left(\hat{p}^{(i)}\right) + \left(1 - y^{(i)}\right)log\left(1 - \hat{p}^{(i)}\right)\right]$$

- No known closed-form equation to compute the value of parameters that minimizes this cost function
- So gradient descent (BGD, SGD, mini-batch GD) is used to find parameters.
- The cost function is convex, so Gradient Descent (or any other optimization algorithm) is guaranteed to find the global minimum

Reading

- Aurélien Géron. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems. O'Reilly, 2017.
  Chapter 4

- Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press, 2016.
  Sections 4.3, 5.1-5.3