Machine Learning 2018 – Linear Regression

Hong, ONG Xuan

November 24, 2018

Contents

What is covered in this lecture

- How do we define a linear regression model?
- How do we learn the parameters of our model $f(\mathbf{x})$ from training examples?
- How do we extend linear models to nonlinear models?

Possible applications

- Predict tomorrow's stock market **prices** given current market conditions and other possible side information.
- Predict the **age** of a viewer watching a given video on YouTube.
- Predict the **temperature** at any location inside a building using weather data, time, door sensors, etc.
- Predict the **salaries** of graduate students given GPAs, number of social activities, gender, living location, etc.
- Predict the **number of users** sharing your post on Facebook based on your friend list, hashtag popularity, previous posts, etc.

Could you see the trend?

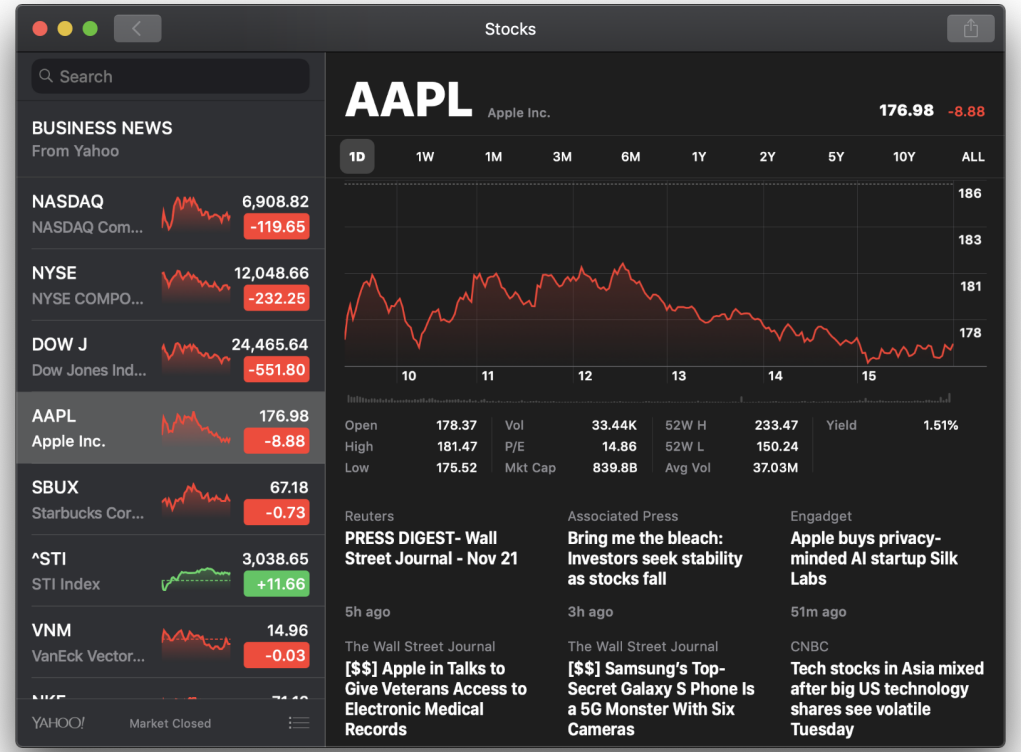

Figure: Apple stock prices (AAPL)

- Normally, we will use stock analysis techniques such as Fibonacci retracement, candlestick, bull/bear signal, etc.
- Can we use Machine Learning methods to help us **automate** the whole process with acceptable results?

How to solve these problems using Machine Learning?

1. Define the problem (e.g. **predicting** some outcome).
2. Collecting the appropriate **data set**.
3. Choose the right machine learning algorithm.
4. Define **evaluation metrics** of the model (e.g. Accuracy, AUC, Precision, Recall, etc.)

Choose the right machine learning algorithm

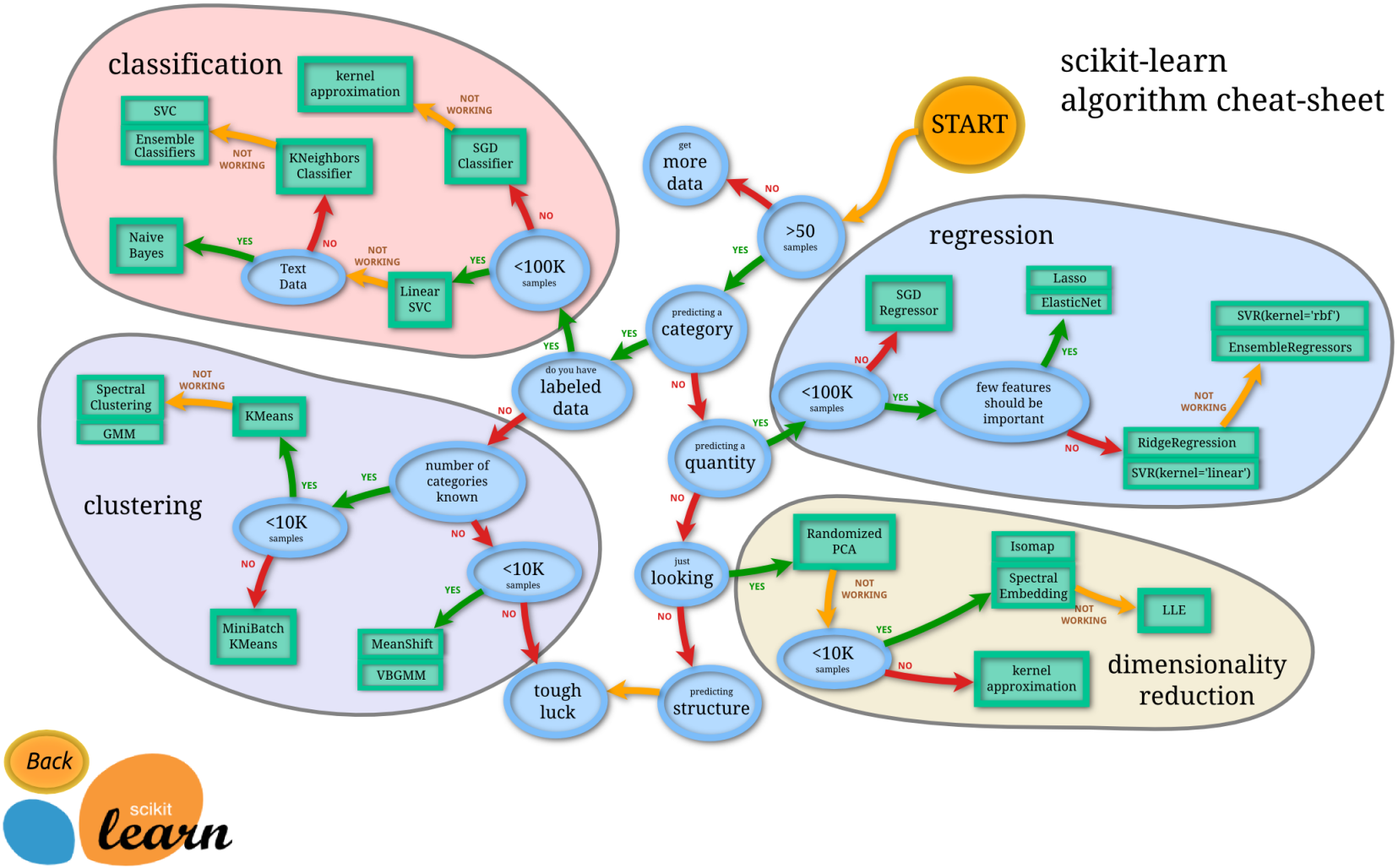

Figure: Machine learning map (sklearn)

Modeling process

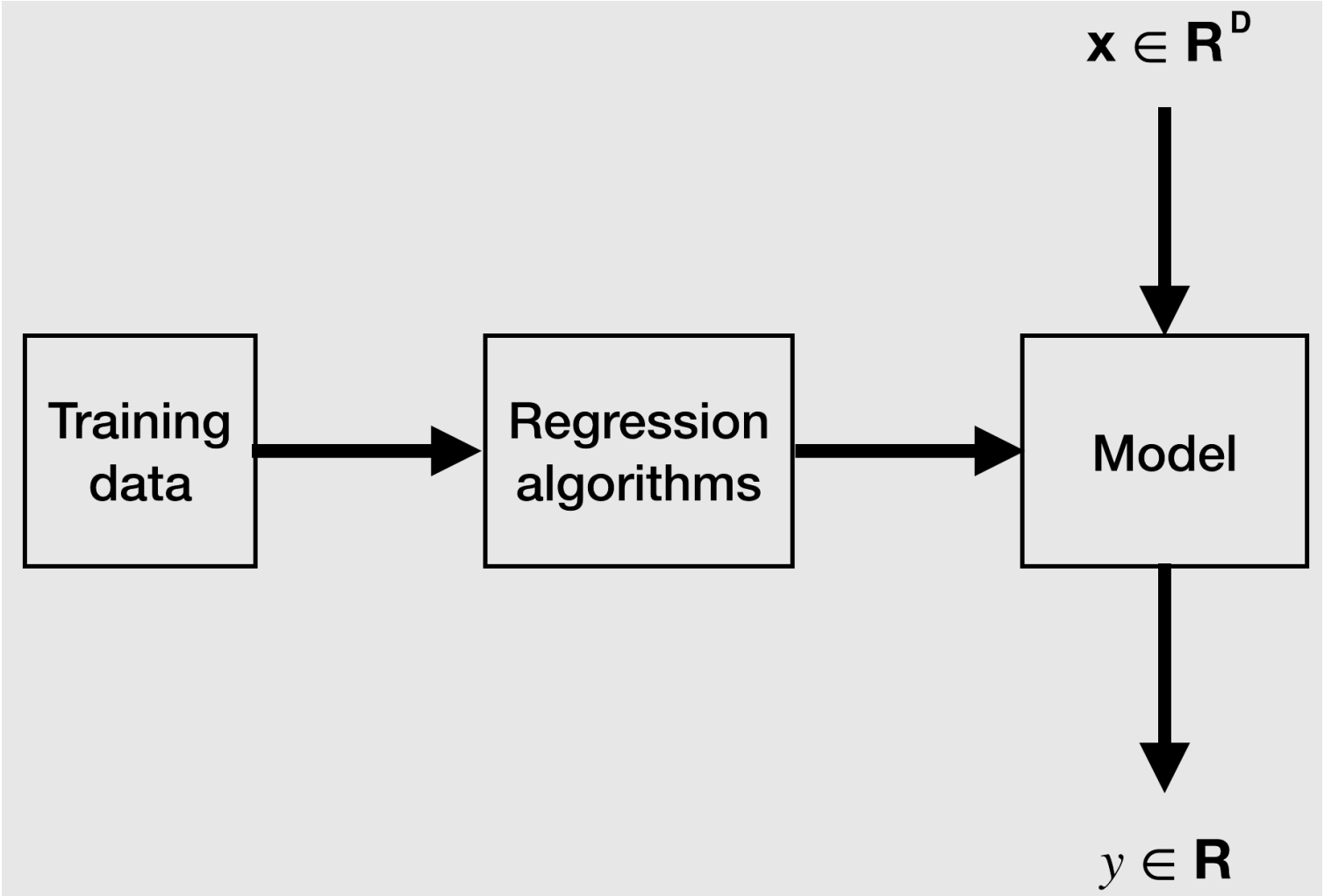

Figure: Regression process

Key points of model

- Need data to build prediction model (training process).
- Could predict *unseen data* in the future (**generalization**).
- **"No Free Lunch"** theorem states that there is no one model that works best for every problem.

Could you see the trend?

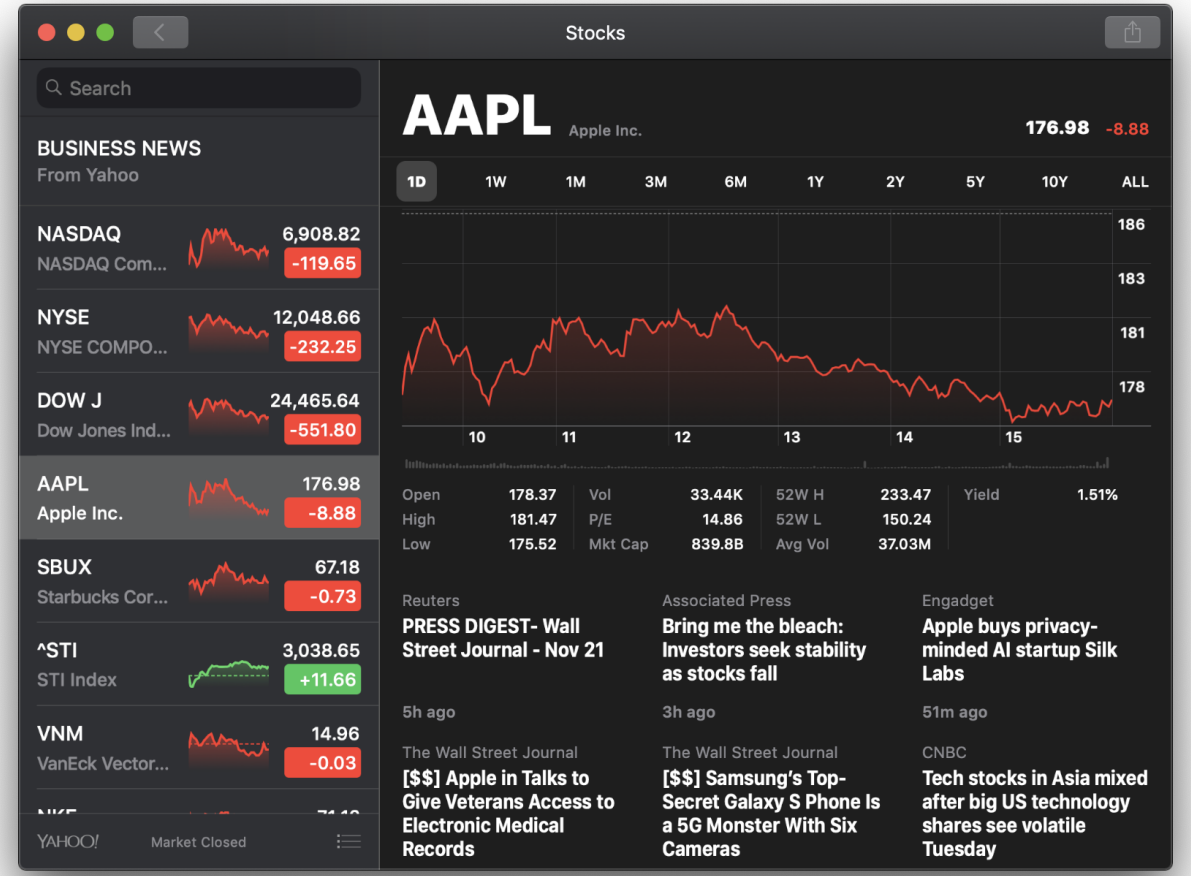

Figure: Apple stock prices (AAPL)

How to draw a straight line that express the trend of data?

Figure: There are many possible straight lines for predicting trends

Contents

Linear function

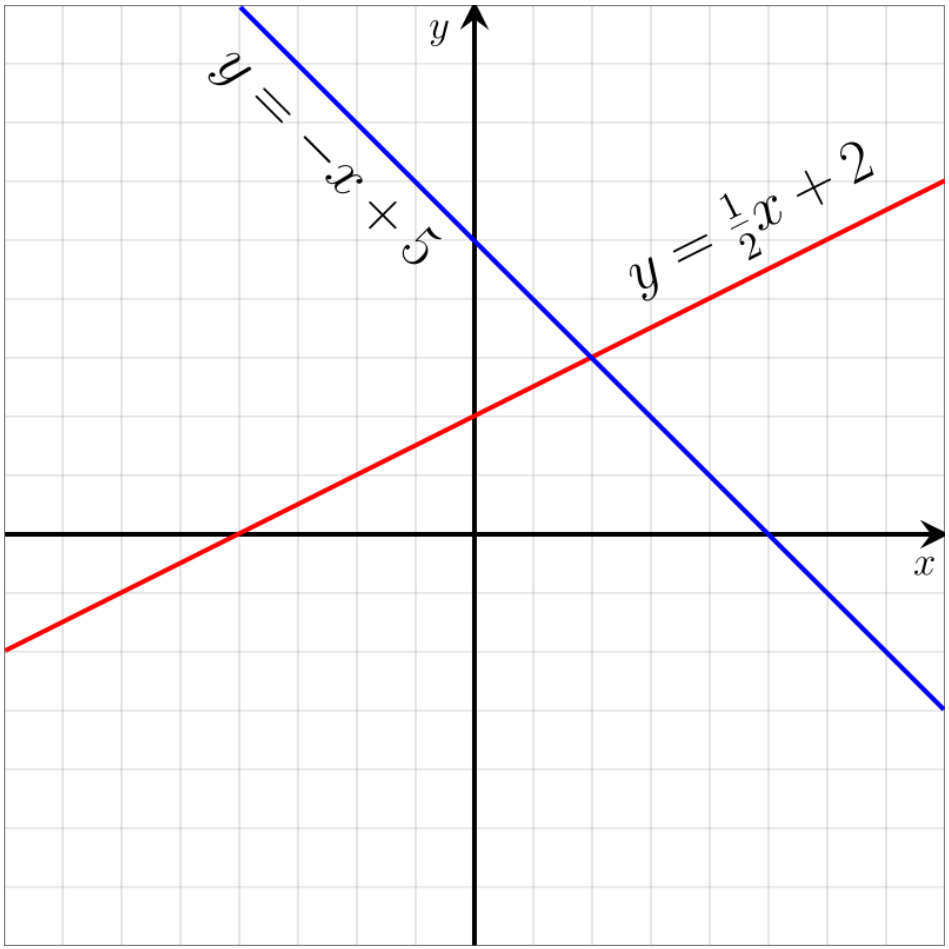

Figure: Linear function (https://en.wikipedia.org/)

Linear model for regression

Linear model for regression is a linear combination of the input variables. It assumes the dependency of the response variable y on the explanatory variables $\mathbf{x}$ is linear.

Formula

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + ... + w_D x_D = w_0 + \sum_{j=1}^{D} w_j x_j$$

where

- $y \in \mathbf{R}$: response variable, dependent variable, outcome.
- D: number of dimensions of the input vector $\mathbf{x}$.
- $\mathbf{x} = (x_1, ..., x_D)^T$: input vector (explanatory variable, independent variable, features).
- $\mathbf{w} = (w_0, ..., w_D)$: parameters.
- $D + 1$: total number of parameters.

Hyperplane

Linear is a **straight line** in 2 dimensions space, a **plane** in 3 dimensions space, and a hyperplane in D-dimensional space.

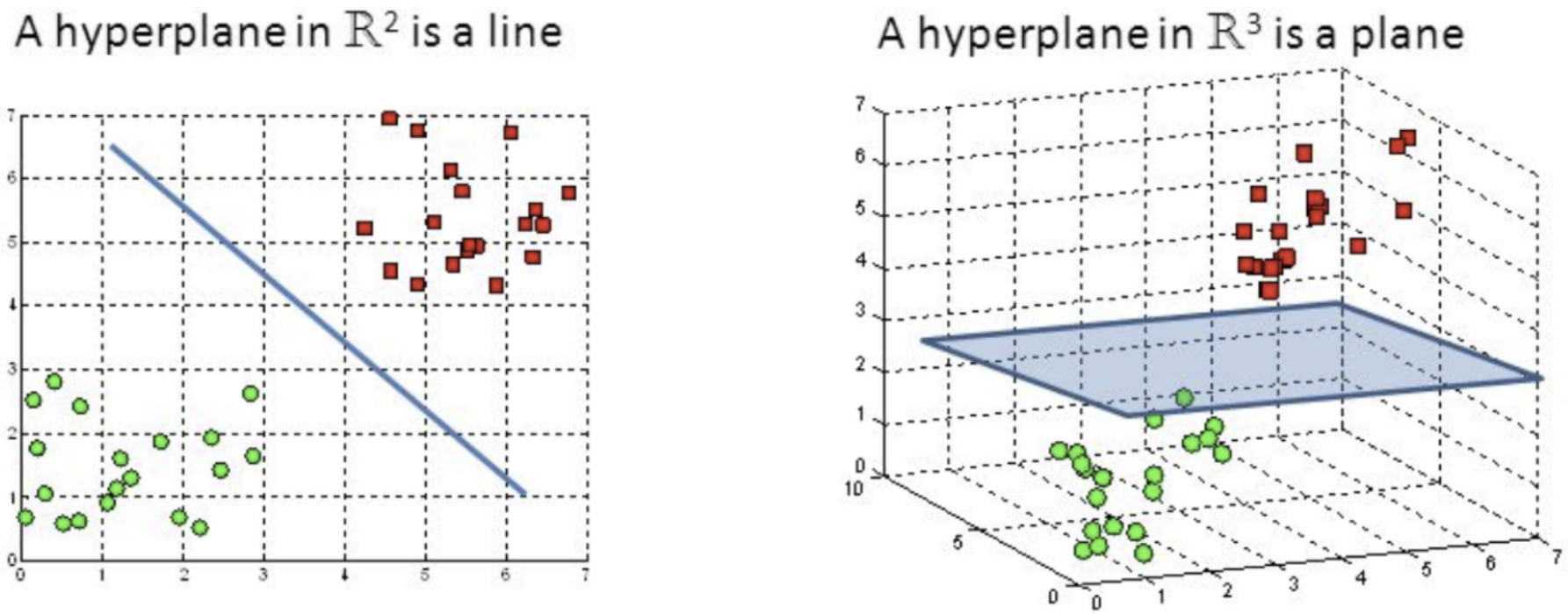

Figure: https://towardsdatascience.com/

Linear regression in one picture

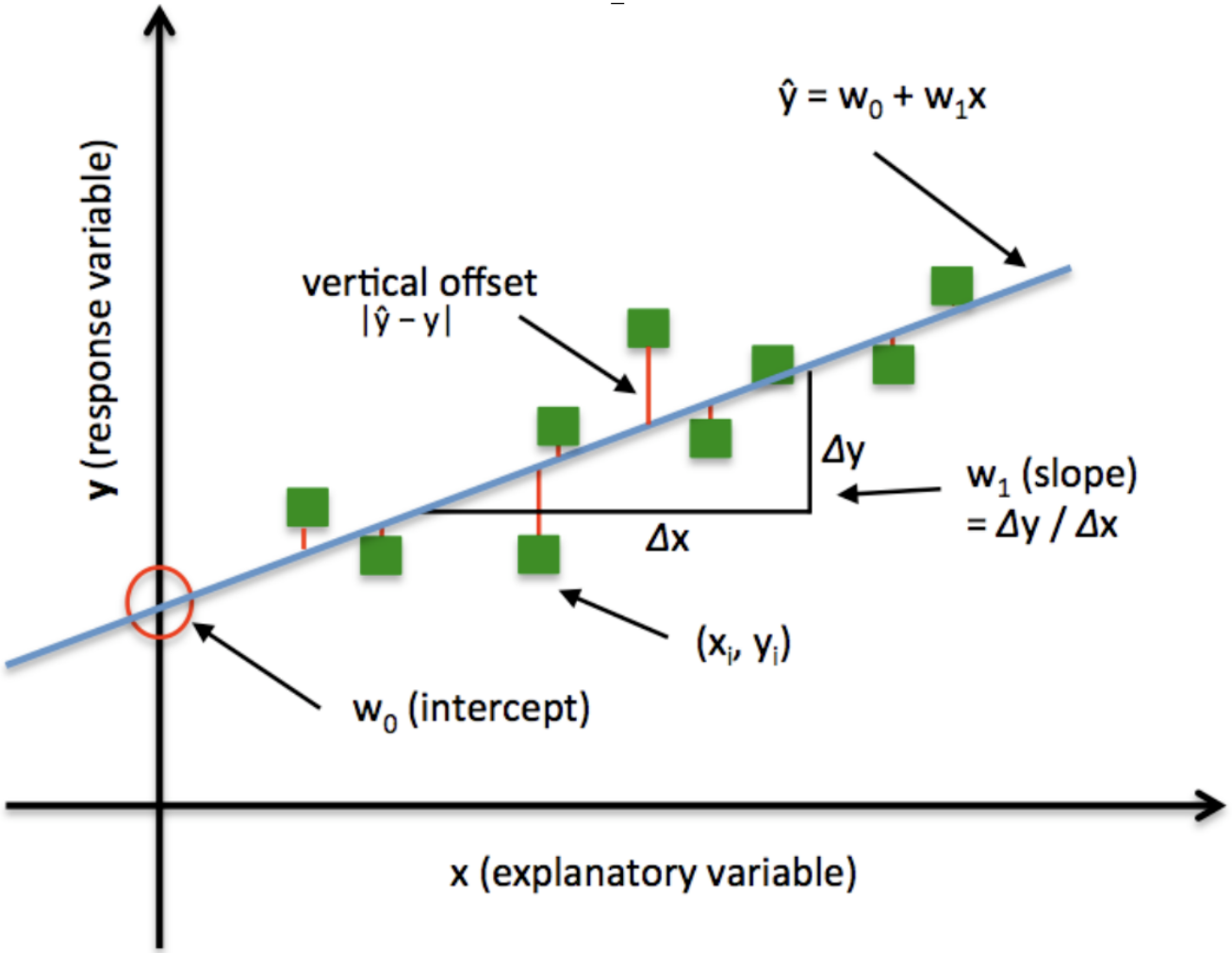

Figure: http://rasbt.github.io/

Loss function

Given the features $\mathbf{x}$, the predicted value of y, $\hat{y}$, is given by $\hat{y} = f(\mathbf{x}) = w_0 + \sum_{j=1}^{D} w_j x_j$

Loss function

A loss function is a measure of how good a prediction model does in terms of being able to predict the expected outcome.

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

where N is the number of training examples.

Our goal: find parameters $\mathbf{w}$ that minimize the loss function. How?

Contents

Using Ordinary Least Squares

We have

$$L(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y^i - \mathbf{w}\mathbf{x}^{(i)})^2$$

where we let $x_0^{(i)} = 1$ to simplify the notation.

Our goal is to find $\hat{\mathbf{w}}$:

$$\hat{\mathbf{w}} = argmin_{\mathbf{w}} L(\mathbf{w}) = argmin_{\mathbf{w}} \left(\frac{1}{2}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2\right)$$

Using Ordinary Least Squares

$$\begin{aligned} L(\mathbf{w}) &= (y - \mathbf{X}\mathbf{w})^T (y - \mathbf{X}\mathbf{w}) \\ &= y^T y - 2\mathbf{w}^T \mathbf{X}^T y + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w}. \end{aligned}$$

Setting the gradient to 0:

$$\begin{aligned} \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} &= -2\mathbf{X}^T y + 2\mathbf{X}^T \mathbf{X}\mathbf{w} = 0 \\ &\iff \mathbf{X}^T \mathbf{X}\mathbf{w} = \mathbf{X}^T y \\ &\iff \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T y. \end{aligned}$$

Notes:

- The Hessian in this case is $2\mathbf{X}^T\mathbf{X}$, which is a positive semidefinite matrix.
- The matrix $\mathbf{X}^T\mathbf{X}$ must be invertible and difficult to scale with high dimension input vector.
- The case in which $\mathbf{X}^T\mathbf{X}$ is non-invertible will be addressed later.

Solve with Gradient descent

Gradient descent algorithm

Initialize $\mathbf{w} = [0, ..., 0]$;
for $t = 1, ..., T$ **do**
  $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla L(\mathbf{w})$
end

- η: step size
- $\nabla L(\mathbf{w})$: gradient

Cons: requires the entire set of data samples to be loaded in memory, since it operates on all of them at the same time

Solve with Stochastic Gradient descent

Stochastic Gradient descent algorithm

Initialize $\mathbf{w} = [0, ..., 0]$;
for $t = 1, ..., T$ **do**
  for $(x, y) \in D_{train}$ **do**
    $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla L(x, y, \mathbf{w})$
  end
end

- Pros: during learning, compute $L(x, y, \mathbf{w})$ before updating $\mathbf{w}$, so require less memory.
- Cons: requires a number of hyperparameters such as the regularization parameter and the number of iterations.

How about complex data set?

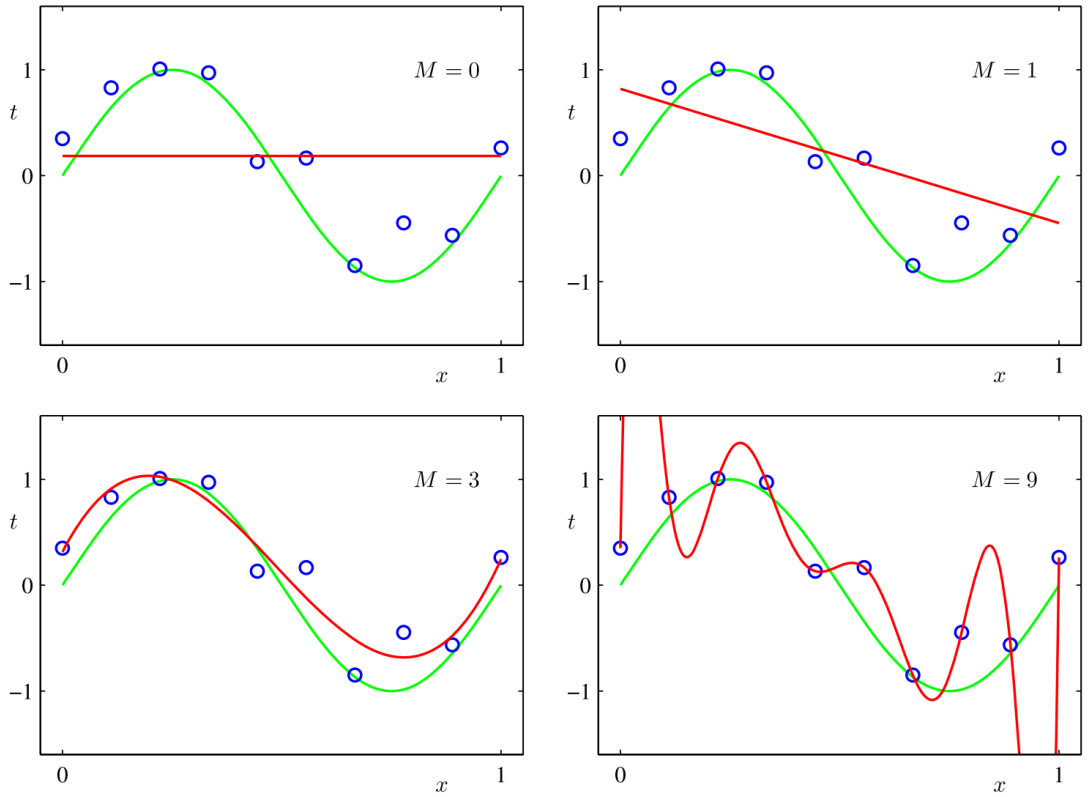

Figure 1.4 Plots of polynomials having various orders M, shown as red curves, fitted to the data set shown in Figure 1.2.

Figure: C. Bishop, Pattern Recognition and Machine Learning

Contents

Basis function

Extend the class of models by considering linear combinations of fixed **nonlinear functions** of the form

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$

where $\phi_j(\mathbf{x})$ are known as basis functions. Identity "basis function" is $\phi(\mathbf{x}) = \mathbf{x}$.

Some basis function

Polynomial basis function

$$\phi_j(x) = x^j$$

Gaussian basis function

$$\phi_j(x) = exp\{-\frac{(x-\mu_j)^2}{2s^2}\}$$

Sigmoidal basis function

$$\phi_j(x) = \sigma(\frac{x-\mu_j}{s})$$

where $\sigma(a)$ is the logistic sigmoid function defined by

$$\sigma(a) = \frac{1}{1+exp(-a)}$$

Some basis function

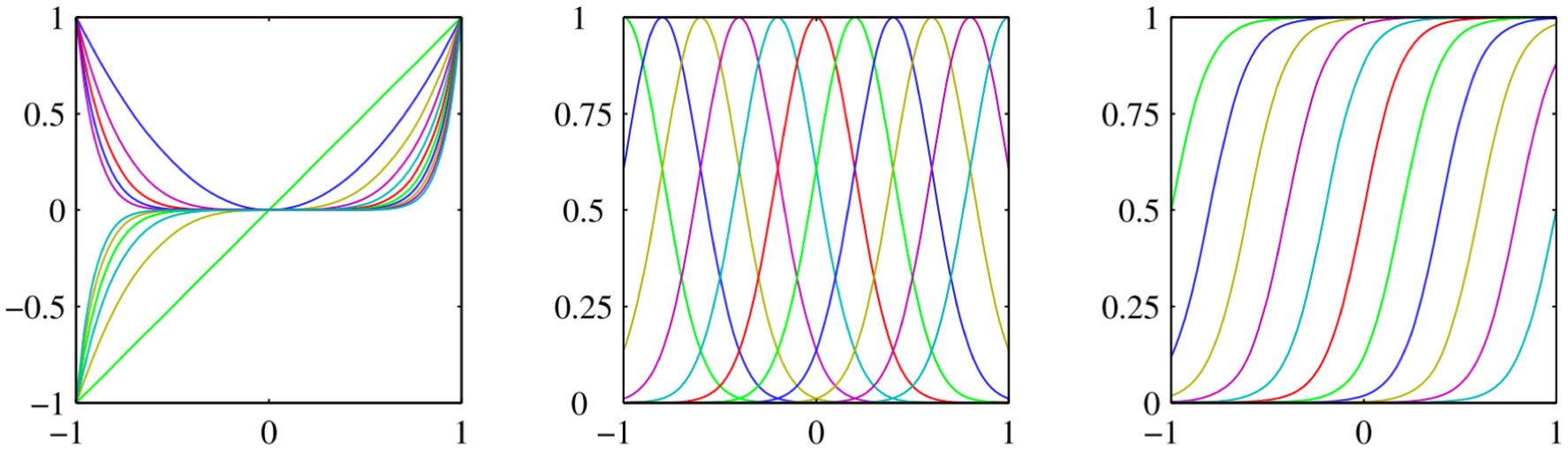

Figure 3.1 Examples of basis functions, showing polynomials on the left, Gaussians of the form (3.4) in the centre, and sigmoidal of the form (3.5) on the right.

Figure: C. Bishop, Pattern Recognition and Machine Learning

What is not covered in this lecture

- Bias and variance.
- Overfitting and regularization (Lasso L1, Ridge L2).
- Regression for multiple outcome.

Contents

References

[1] Bishop, C. M. (2013). Pattern Recognition and Machine Learning. Journal of Chemical Information and Modeling (Vol. 53).
[2] Wikipedia. Gradient descent.
[3] Wikipedia. Ordinary least squares.
[4] Wikipedia. Stochastic gradient descent.