Linear Regression & Logistic Regression

Machine Learning

- Machine learning means building algorithms based on a set of examples from the same phenomenon.
- Machine learning can also be seen as a process for solving real-world problems in considered that it consists of two parts:

  1. Data collection

  2. Building a statistical model based on data

Types of learning

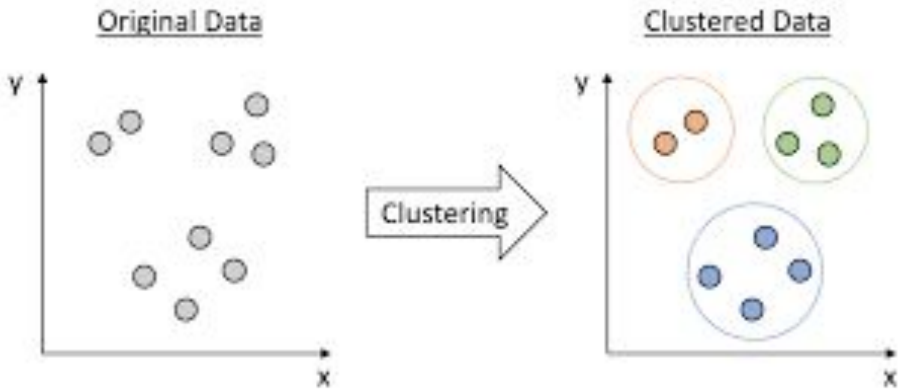
- Supervised learning
- Unsupervised learning
- Reinforcement learning

Supervised Learning

- The dataset is the collection of labeled examples; $\{(x_i, y_i)\}_{i=1,\ldots,N}$
- **Goal:** Learn a function to map x -> y
- Each element x_i among N is called a feature vector.
- A feature vector is a vector in which each dimension contains a value that describes the example somehow.
- The label y_i can be either an element belonging to
  - a finite set of classes $\{1, 2, \ldots, C\}$,
  - a real number,
  - a more complex structure, like a vector, a matrix, a tree, or a graph.

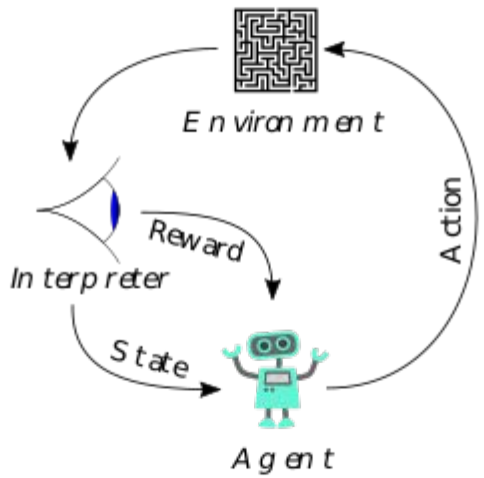
Unsupervised Learning

- In unsupervised learning, the dataset is a collection of unlabeled examples, $\{x_i\}_{i=1,\ldots,N}$ (No label!)
- **Goal:** Learn some underlying hidden structure of the data that can be used to solve a practical problem.



Reinforcement Learning

- Problems involving an **agent** interacting with an **environment**, which provides numeric **reward** signals
- **Goal:** Learn how to take actions in order to maximize reward



Types of learning

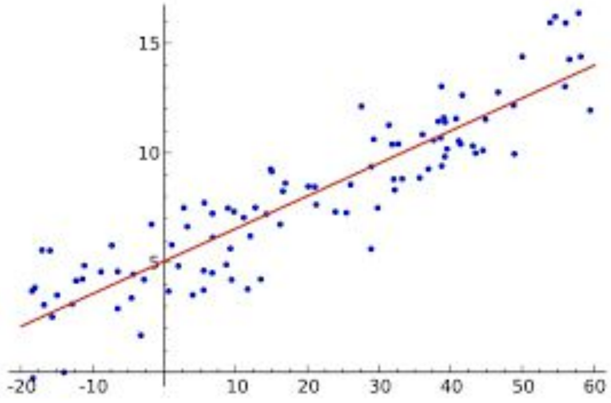
- Supervised learning. Examples:
  - Regression
    - In regression the output is **continuous**
  - Classification
    - For classification the output(s) is nominal
  - object detection
  - image captioning
  - etc
- Unsupervised learning
- Reinforcement learning

Regression

- In regression the output is **continuous**
- Function Approximation
  - Many models could be used – Simplest is linear regression
  - We are looking to build a linear model as follows:

$$f_{w,b}(\mathbf{x}) = \mathbf{wx} + \mathbf{b}$$

y
dependent
variable
(output)

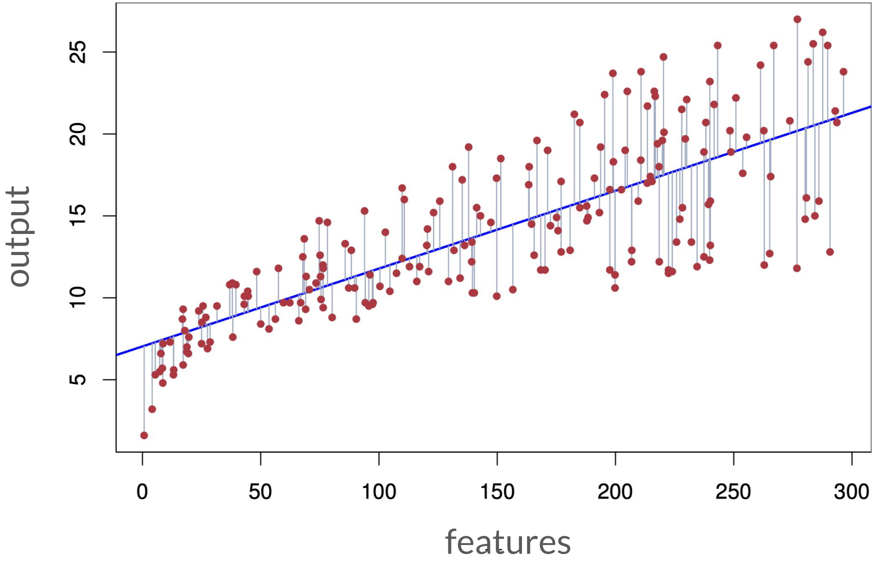


x – independent variable (input)

Linear Regression

- To find the best **w** and **b**, the optimization problem that needs to be solved come as follows:
- Minimize the cost function

$$\frac{1}{N}\sum_{i=1}^{N}\underbrace{(f_{w,b}(\mathbf{x_i}) - \mathbf{y_i})^2}_{\text{Loss function}}$$



Two approaches:

1. Direct solution
2. Gradient descent

Cost function:

$$J(w,b) = \frac{1}{N}\sum_{i=1}^{N} L(y^{(i)}, t^{(i)})$$

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

Two approaches:

1. Direct solution
2. Gradient descent

Cost function:

$$J(w,b) = \frac{1}{N}\sum_{i=1}^{N} L(y^{(i)}, t^{(i)})$$

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

Derivatives:

$$\frac{\partial J}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N} (y^{(i)} - t^{(i)})x_j^{(i)}$$

$$\frac{\partial J}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} (y^{(i)} - t^{(i)})$$

Direct solution

- Chain rule for derivatives:

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y}\frac{\partial y}{\partial w_j} = \frac{\mathrm{d}}{\mathrm{d}y}\left[\frac{1}{2}(y-t)^2\right]\cdot x_j = (y-t)x_j$$

$$\frac{\partial \mathcal{L}}{\partial b} = y - t$$

- Cost derivatives (average over data points):

$$\frac{\partial \mathcal{E}}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})\, x_j^{(i)}$$

$$\frac{\partial \mathcal{E}}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} y^{(i)} - t^{(i)}$$

Direct solution

- The minimum must occur at a point where the partial derivatives are zero.

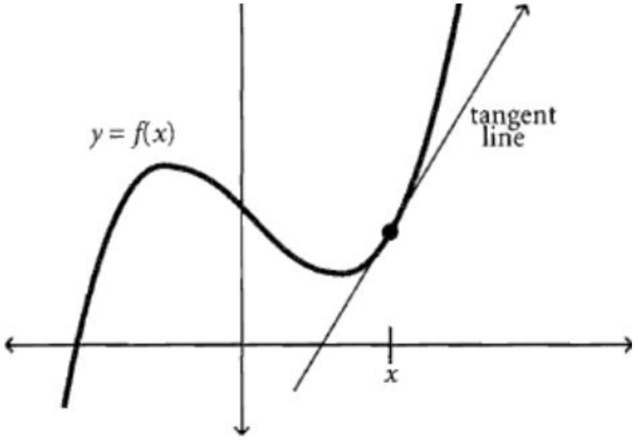
$$\frac{\partial \mathcal{E}}{\partial w_j} = 0 \qquad \frac{\partial \mathcal{E}}{\partial b} = 0.$$

- This turns out to give a system of linear equations, which we can solve efficiently.
- Optimal weights:

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$$

Gradient descent

- Gradient descent is an iterative algorithm, which means we apply an update repeatedly until some criterion is met.
- We initialize the weights to something reasonable (e.g. all zeros) and repeatedly adjust them in the direction of steepest descent.



Gradient descent

- The following update decreases the cost function:

$$w_j \leftarrow w_j - \alpha \frac{\partial \mathcal{E}}{\partial w_j}$$
$$= w_j - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})\, x_j^{(i)}$$

- α is a learning rate. The larger it is, the faster $\mathbf{w}$ changes.
  - We'll see later how to tune the learning rate, but values are typically small, e.g. 0.01 or 0.0001

Multiple features - vectorization

- This gets its name from the gradient:

$$\frac{\partial \mathcal{E}}{\partial \mathbf{w}} = \begin{pmatrix} \frac{\partial \mathcal{E}}{\partial w_1} \\ \vdots \\ \frac{\partial \mathcal{E}}{\partial w_D} \end{pmatrix}$$

  - This is the direction of fastest increase in $\mathcal{E}$.
- Update rule in vector form:

$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \alpha \frac{\partial \mathcal{E}}{\partial \mathbf{w}} \\ &= \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})\, \mathbf{x}^{(i)} \end{aligned}$$

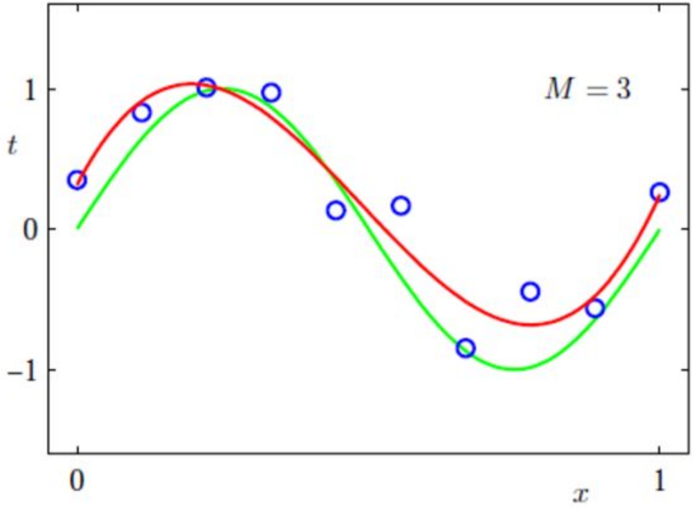
Gradient descent

- Why gradient descent, if we can find the optimum directly?
  - GD can be applied to a much broader set of models
  - GD can be easier to implement than direct solutions, especially with automatic differentiation software
  - For regression in high-dimensional spaces, GD is more efficient than direct solution (matrix inversion is an $\mathcal{O}(D^3)$ algorithm).

Let’s implement it …

Future studies: Polynomial Regression

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$



-Pattern Recognition and Machine Learning, Christopher Bishop.

Logistic Regression

Binary Classification

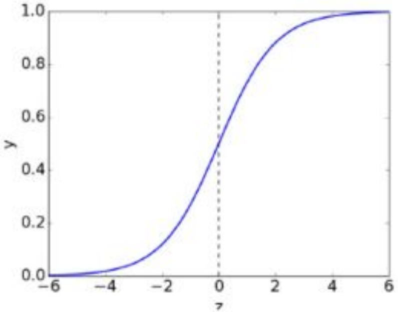
Logistic regression is not really a regression. Rather, it is a learning algorithm for classification.

y_i is a discrete number and indicates the class number. The goal is to build a **linear** model from the labeled data set .

In binary classification is obvious that there is no reason to predict values outside [0,1]. Let's squash y into thisinterval.

Logistic Activation Function

- The logistic function is a kind of sigmoidal, or S-shaped, function:



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

- A linear model with a logistic nonlinearity is known as log-linear:

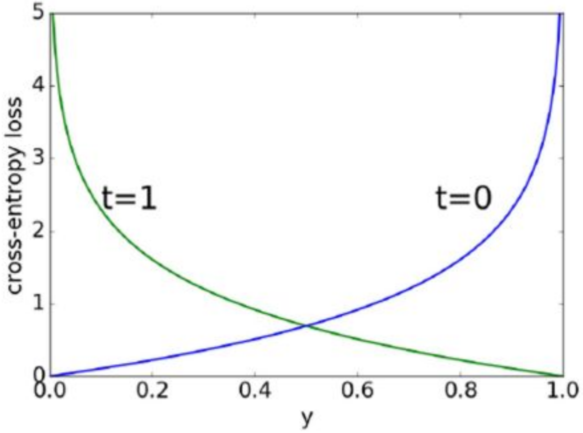
$$z = \mathbf{w}^\top \mathbf{x} + b$$
$$y = \sigma(z)$$

- Used in this way, σ is called an activation function, and z is called the logit.

Cross-entropy Loss

- Because $y \in [0, 1]$, we can interpret it as the estimated probability that $t = 1$.
- Cross-entropy loss captures this intuition:

$$\mathcal{L}_{\mathrm{CE}}(y, t) = \begin{cases} -\log y & \text{if } t = 1 \\ -\log(1 - y) & \text{if } t = 0 \end{cases}$$
$$= -t \log y - (1 - t) \log(1 - y)$$



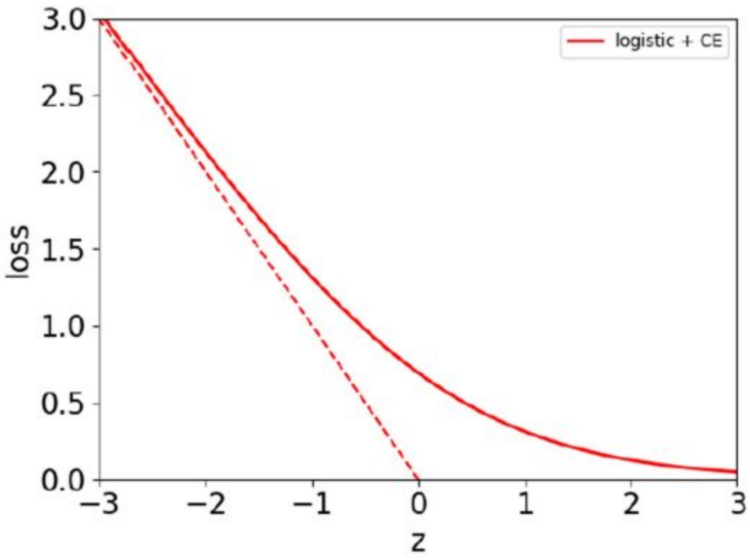
Logistic Regression

$$z = \mathbf{w}^\top \mathbf{x} + b$$

$$y = \sigma(z)$$

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

$$\mathcal{L}_{\text{CE}} = -t \log y - (1 - t) \log(1 - y)$$



Gradient descent approach

Now to compute the loss derivatives:

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{L}_{\mathrm{LCE}}}{\mathrm{d}z} &= \frac{\mathrm{d}}{\mathrm{d}z}\left[t\log(1+e^{-z}) + (1-t)\log(1+e^{z})\right] \\
&= -t\cdot\frac{e^{-z}}{1+e^{-z}} + (1-t)\frac{e^{z}}{1+e^{z}} \\
&= -t(1-y) + (1-t)y \\
&= y - t
\end{aligned}$$

Comparison of gradient descent updates

- Linear regression:

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} \left(y^{(i)} - t^{(i)}\right) \mathbf{x}^{(i)}$$

- Logistic regression:

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} \left(y^{(i)} - t^{(i)}\right) \mathbf{x}^{(i)}$$