Carnegie Mellon University

18-819F: Introduction to Quantum Computing 47-779/785: Quantum Integer Programming & Quantum Machine Learning

Introduction to Machine Learning

Lecture 03

2022.09.12.

Electrical & Computer ENGINEERING

TEPPER

USRA Universities Space Research Association

Agenda

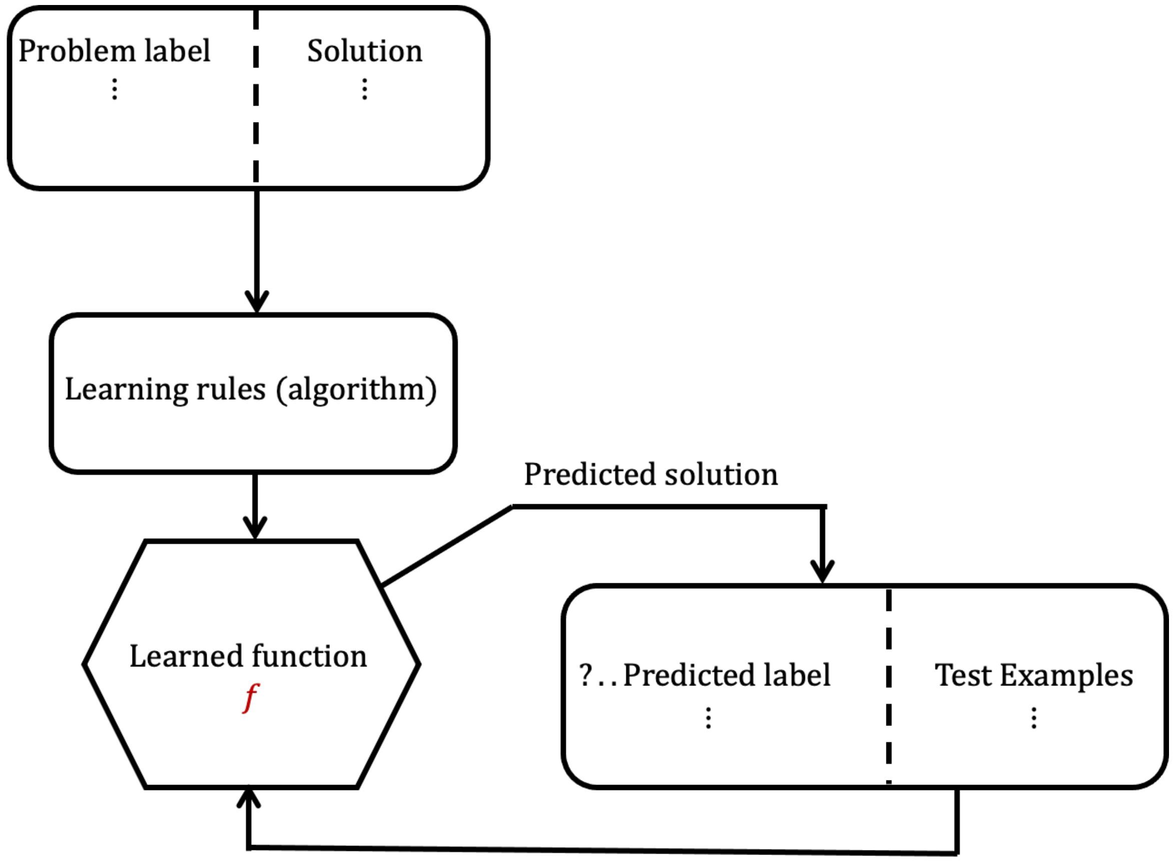
- Introduction to machine learning
  - Meaning of ordinary learning and its translation to machine learning
  - Classes of problems suitable for machine learning
  - Classification
- Regression
  - Linear regression
  - Logistic regression

Machine Learning

- The goal of machine learning is to predict a future based on what is learned about the past or about similar situations.
- What does it mean to learn?
- Learning invokes the notion of cumulative experiences that allow one to recognize a situation or be able to solve/handle a similar problem to one that has been seen in the past.
- The ability to generalize a problem-solving skill is what is generally meant by learning.
- Generalization is central to the concept of learning, and it is this model of learning that we attempt to build into our machines.
- Machine Learning is a framework for induction or inference of general conclusions based on particular examples or instances.

General Setup for Supervised Machine Learning

- The basic setup for machine learning is illustrated on the right.
- This could be the setup Netflix might implement as an algorithm for predicting the kinds of movies you might like to watch; they might write a program that is trained on the movies you have watched in the past. The data for the program would be the movies you have watched and rated. The algorithm's job is to find a function f that will map a new example to a corresponding prediction.
- The algorithm would then be evaluated (tested) on sample test data. This is similar to the way you will be tested in your final exams in any course you take this semester. If you pass, the assumption is that you will be able to solve similar problems your employer throws at you.



Types of Problems Solvable by Machine Learning

- Machine learning cannot be used for all types of problems you encounter in life.

- There are classes of problems that machine learning is good at; the problems must be carefully selected. Below are a few types of problem classes that machine learning can tackle.

- Regression – simple prediction of real numbers, for example, a stock price price next week based on what it was during the past 7 days.
- Ranking – this is when you try to put a set of things in order of relevance. This is what Google Search does; it responds to your query with a list of items ranked according to what the search engine believes is closest to you query.
- Binary classification – when you only want a simple yes or no answer. You could create an algorithm to predict whether students like to eat at Skibo cafe after you perform a survey (this is your training data).
- Multiclass classification – when you have a large basket of fruit (oranges, apples, kiwis and pears), you can write an algorithm to sort the fruit, each into its own class.

Formalizing learning

- To be useful in the context of machines, we must formalize the concept of learning

- At least three things must be defined that would allow one to teach a machine anything:
  - One must create a metric (measurement) for performance on a particular problem of interest.
  - The performance of an algorithm must be measured on unseen data.
  - There should be a relationship between the data the algorithm sees during training time and the data it sees during testing time; in the end, we also want the algorithm to be used only for similar but unseen data.

- A good metric to use for gauging the items above is a function called the loss function, $\mathcal{L}(.,.)$ with two arguments;

- For variables y and $\tilde{y}$ in the argument of the loss function, we expect the value of $\mathcal{L}(y, \tilde{y})$ to measure the error.

- For ordinary regression, the loss function is $\mathcal{L}(y, \tilde{y}) = (y - \tilde{y})^2 = |y - \tilde{y}|$ Eqn. (3.1)

Loss Functions for Binary and Multiclass Classification

- In binary classification, the loss function is a simple yes/no or 0/1 situation that can be written as

$$\mathcal{L}(y,\tilde{y}) = \begin{cases} 0 & \text{if } y = \tilde{y} \\ 1 & \text{otherwise} \end{cases} \qquad \text{Eqn. (3.2)}$$

- For multiclass classification, we can use a similar loss function to the binary classification case.
- Expected loss, $E(x,y)$, for input and output variables x and y can be written as

$$E(x,y) \longrightarrow D\left[\mathcal{L}\left(y, f(x)\right)\right] \qquad \text{Eqn. (3.3)}$$

- The expected loss is the average loss for random variables (x,y) drawn from a sample D. For a discrete probability distribution, the expectation would be written as

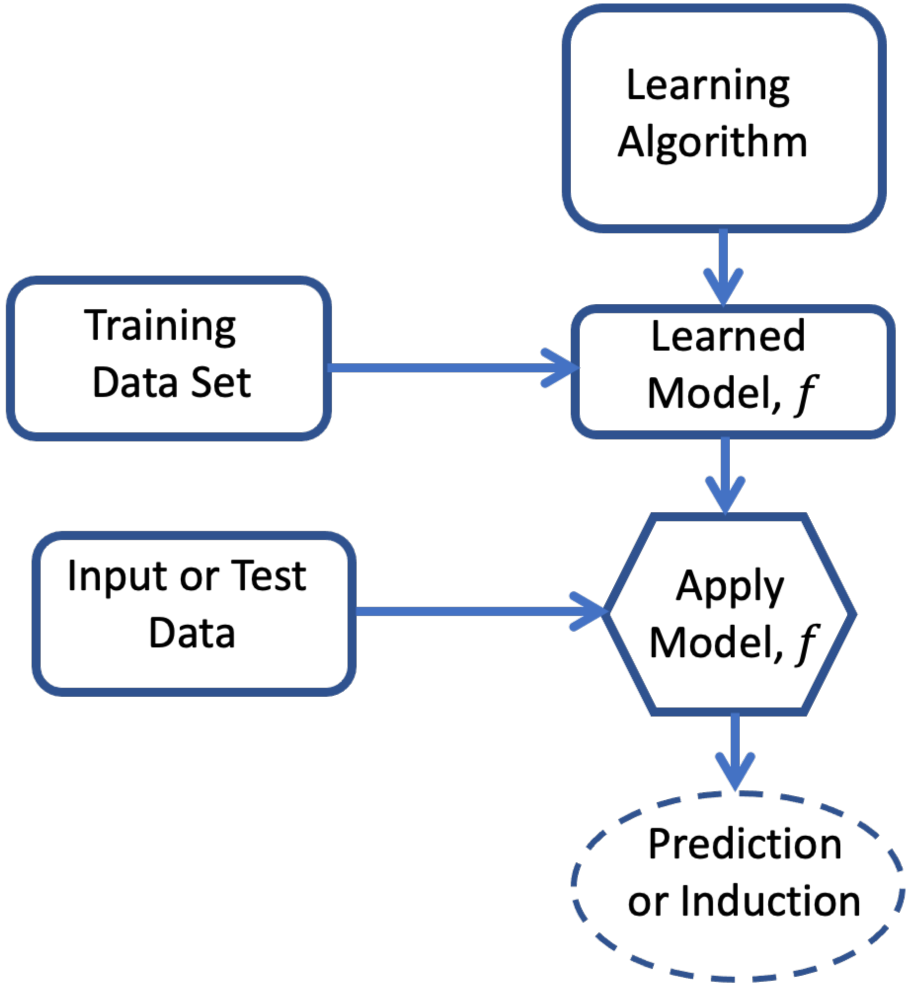
$$E(x,y) \longrightarrow D\left[\mathcal{L}\left(y, f(x)\right)\right] = \sum_{(x,y)\in D}\left[D(x,y)\mathcal{L}\left(y, f(x)\right)\right] \qquad \text{Eqn. (3.4)}$$

- Note that D is a discrete, finite distribution such as $[(x_2,y_1),(x_2,y_2)\ldots(x_N,y_N)]$ with equal weight in each sample, $1/N$, this means the average loss is then

$$E(x,y) \longrightarrow D\left[\mathcal{L}\left(y, f(x)\right)\right] = 1/N\sum_{n=1}^{N}\left[\mathcal{L}\left(y_n, f(x_n)\right)\right] \qquad \text{Eqn. (3.5)}$$

Building classifiers

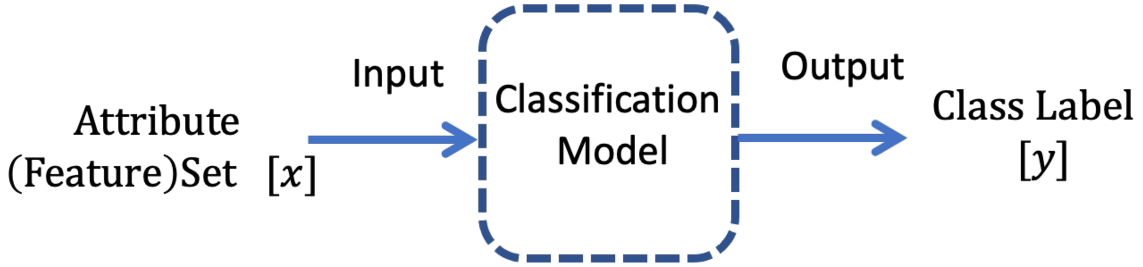
- Classifiers can be built in several ways, depending on the problem at hand. The most popular approaches include:
  - Tree-based classifiers
  - Rule-based classifiers
  - Support vector networks
  - Bayes classifiers
  - Neural networks
- We will only have occasion to discuss at some length the Bayes classifier in this course.
- Evaluation of performance of the classification model can be on the counts of the test records correctly and incorrectly predicted by the model.



Classification

- Classification assigns objects to one of several predefined categories. The main task is to learn a target function f that maps each attribute set of x to one of predefined class labels y. A classification model can serve as an explanatory tool to distinguish between objects of different classes. It can also serve as a predictive tool to label unknown records.

- Classification techniques are best used for predicting or describing data sets with binary or nominal categories.



Performance evaluation of a classifier by a confusion matrix

- Classifiers evaluated by the number of correct and incorrect predictions often use a confusion matrix for computing the accuracy of the model.

- In a two-class problem, a relevant confusion matrix is illustrated on the next slide. According to the matrix, the number of correct predictions by the model considered in this instance is given by $c_{11} + c_{00}$ and the number incorrect predictions is $c_{01} + c_{10}$. The accuracy of the model can be defined as

$$\text{Accuracy} = \frac{\text{Number of correct predications}}{\text{Total number of predictions}} = \frac{c_{11} + c_{00}}{c_{11} + c_{10} + c_{01} + c_{00}}$$

- Similarly, the error rate is defined as

$$\text{Error rate} = \frac{\text{Number of incorrect predications}}{\text{Total number of predictions}} = \frac{c_{01} + c_{10}}{c_{11} + c_{10} + c_{01} + c_{00}}$$

Confusion matrix for a two-class classifier

- A confusion matrix for a typical two-class classifier is shown on below as a tabular listing of the possible outcomes of the classification of the model.

- The matrix elements c_{ij} indicate the agreement or disagreement of the predicated and actual class category.

		Predicted Class	
		Class = 1	Class = 0
Actual	Class = 1	c_{11}	c_{10}
Class	Class = 0	c_{01}	c_{00}

Decision Trees

- A decision tree model is another way to create an algorithm that a machine can implement;
- There is no best way to finding the optimal decision tree. The best-known approach uses what is called the greedy strategy to grow a tree.
- The greedy strategy makes a series of locally optimal decisions that contribute to partitioning the data.
- The best-known method for the greedy strategy is Hunt's algorithm, which is recursive and successfully partitions the training data into purer subsets.
- If D_T is the training data set associated with node T and the corresponding labels are $y = \{y_1, y_2, \ldots y_n\}$, then one can implement Hunt's algorithm with the steps on the next slide.

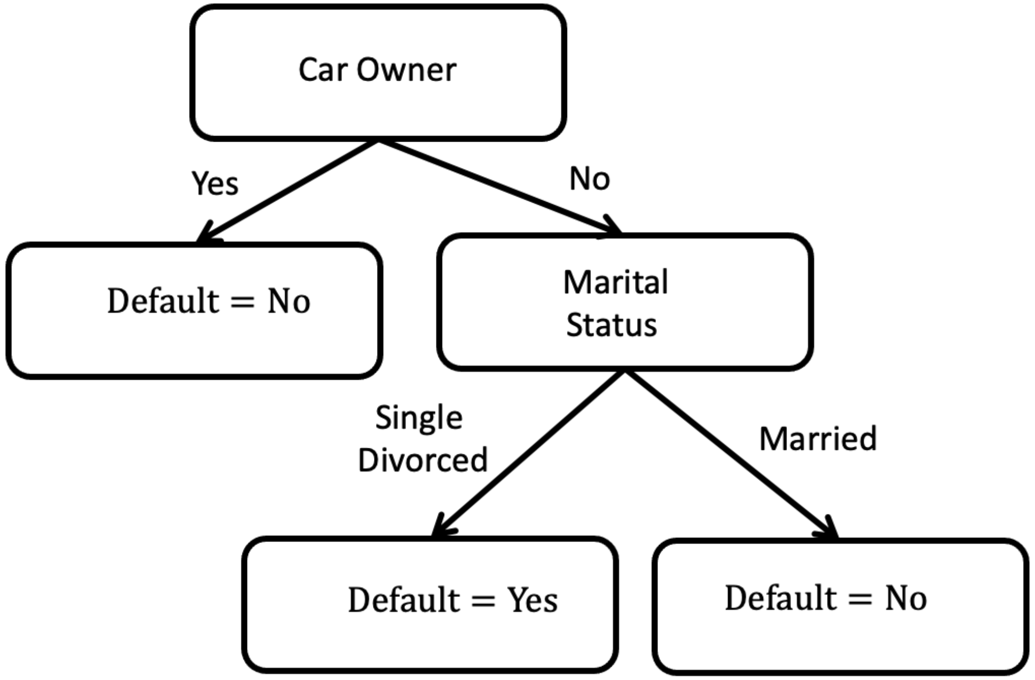
Hunt’s Algorithm for Decision Tree Creation

- If all records (data) in D_T belong to same class y_T, then T is a leaf node.
- If D_T has records (data) that belong to more than one class, then an attribute (feature) test condition is selected to partition the data into smaller subsets; a child node is created for each outcome of the test condition, and the data D_T is distributed to the children based on the outcomes. The process is recursively applied to each child node.
- One can use the bank lender’s problem to illustrate Hunt’s method.
- We assume the banker has accumulated lots of data similar to what is on the next slide; (s)he wants to use it to predict default rate of future customers who wish to borrow money from the bank.
- Algorithms like these can lead to red-lining practices, frowned on because they lead to inequities.

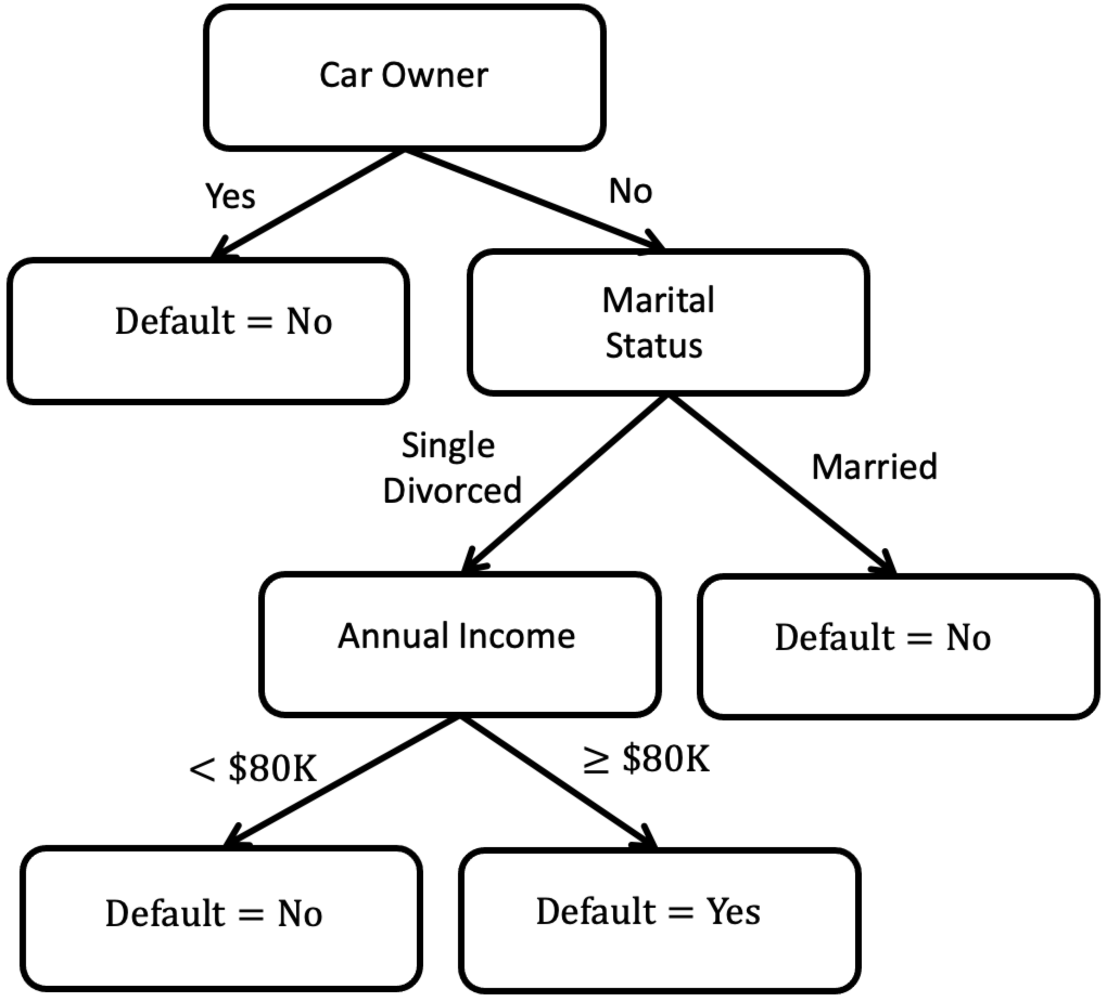
Bank Data from Customers and Creation of a Decision Tree

- By casually inspecting the data, one notices that people who own cars do not default. We therefore use this as root node.

ID	Car Owner	Marital Status	Income	Default
1	Yes	Single	$125K	No
2	No	Married	$100K	No
3	No	Single	$70K	No
4	Yes	Married	$120K	No
5	No	Divorced	$95K	Yes
6	No	Married	$60K	No
7	Yes	Divorced	$220K	No
8	No	Single	$85K	Yes
9	No	Married	$75K	No
10	No	Single	$90K	Yes
	Binary	Categorical	Continuous	Class



Decision Tree Addition after First Pass



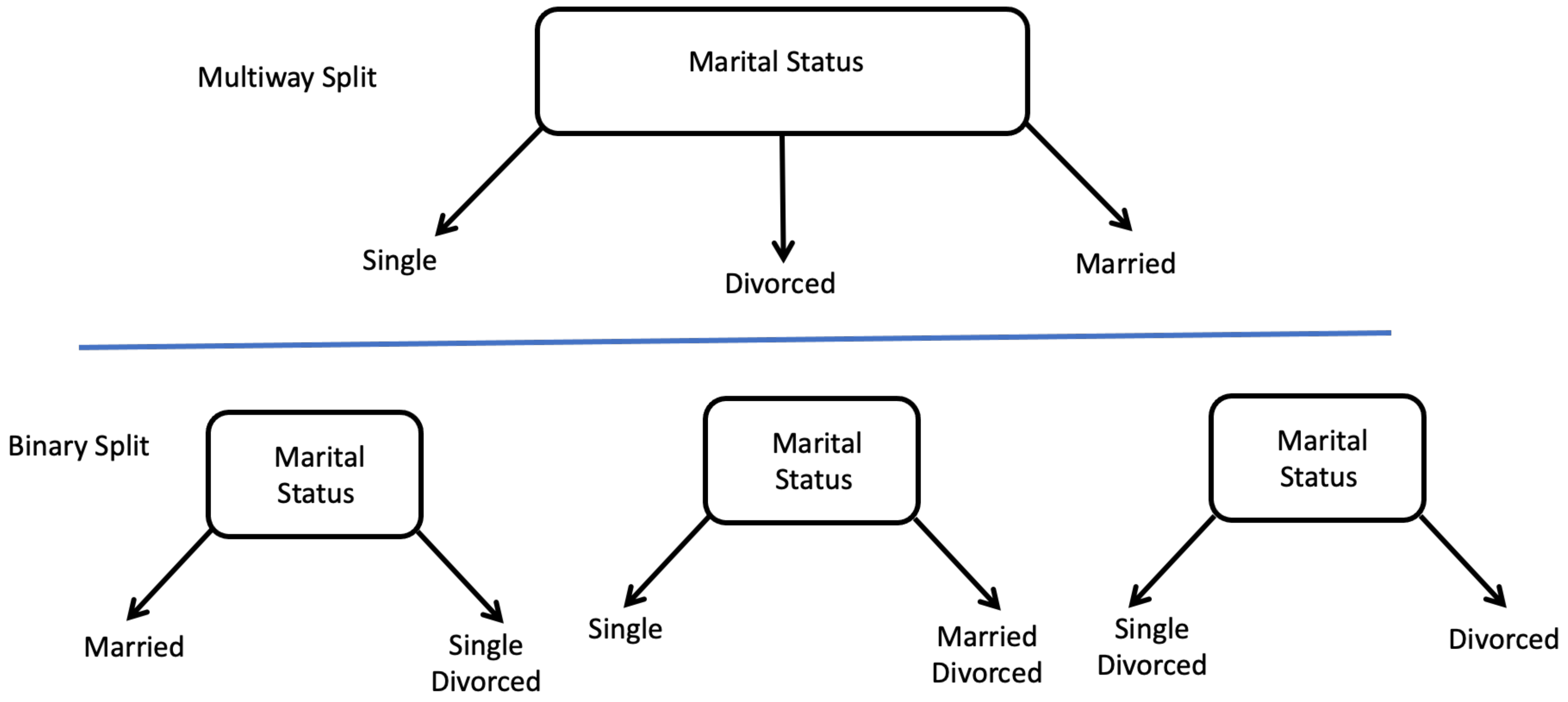
- The data also suggests that we consider income for single, divorced people. Resulting tree would be as shown on left.
- Hunt’s algorithm works if every combination of attributes is present in training data and each combination has a unique label. If a child node in step 2 is empty, then it is declared a node.
- In step 2, if data has identical attribute values, then it is not possible to split node any further, then the node is declared a leaf.

Problems with Decision Tree Induction

- It is typically difficult to decide on how to split the training data when using the decision tree model.
- A second issue, once one has made the decision to split the data, is how to stop the splitting procedure.
- One way to handle the issue of splitting the data is by selecting a test condition for dividing the data into smaller subsets; an objective measure must be provided for evaluating the "goodness" for each test condition.
- Stopping the process can be done by continuing to expand the mode until all data have identical attributes or the data belongs to the same class.

Bank Data Splitting Decision Tree Options

- There is no a sharp rule on how to split the data or when to stop the decision tree from creating another child node. The illustration below shows several equally valid ways.



Linear Regression as a Machine Learning Algorithm

- Linear regression is one of the simplest algorithms that can be used to demonstrate the idea of a machine learning algorithm for predicting a range of continuous values. A prototypical function for linear regression is

$$y = wx + b \text{ Eqn. (3.6).}$$

- The variables w and b are parameters of the model that must be "learned" to produce the most accurate predictions of y for input x;

- We are given the sales and advertising budgets for several companies in the table (matrix) to the right. The columns (features) represent expenditures on advertising and number of units sold in a year.

- Our goal is to develop a function that predicts units sold; note that rows (observations) represent the names of companies.

Company	Advertising ($\times\$ 10^3$)	Sales Volume ($\times 10^6$)
Apple	20	25.1
Dell	39	10.2
HP	45	11.2
Asus	15	3.5

Example: Predicting Sales Units from Advertising Amount

- Assume a nominal predictive model of the form: y= $w * Advertising + Bias$.
- The variable coefficient w is called the “weight” in machine learning; “Advertising” is an independent variable called the “feature” in machine learning. ”Bias” is the intercept and is the offset.
- The goal for our model and the resulting algorithm is to learn the values of the variable “*w*” and the “Bias” during training.
- We care most about *accuracy*, and measure it by a *cost function*, which permits optimization of the weights.
- In linear regression, the best cost function is the mean square error (MSE) or the L_2 norm.

Linear regression in machine learning

- Linear regression is one of the simplest machine learning algorithms. In conventional linear algebra, a linear function takes input, and manipulates it in some fashion to generate an output. For input x_i, the function may generate output y_i. Thus, $y_i = f(x_i)$. If $f(x_i)$ is of the form

$$f(x_i) = wx_i + b \qquad \text{Eqn. (3.7)},$$

where w and b are parameters (constants) of the function, the function is said to be linear.

- Such functions can have multiple input variables (features), x_i, corresponding to outputs, y_i. The task of machine learning is as follows: given the input and output data in the table below, find a function (model) that can predict outputs for any future input data that the machine has not seen, but is similar in kind to the data it was trained on.

x_{in}	1	2	3	4	5
y_{out}	5	7	9	11	13

The machine learning problem

- The usual linear algebra problem is to calculate y_i, given x_i, with a known equation. In machine learning, the equation is not known; it must be found from the data, while making sure one uses the most optimal parameters (constants) in the equation. If one assumes a linear model, then the job of training is to find the most optimal parameters (constants) w and b in Eqn. (3.7).

- The problem then can be framed as that of minimizing the deviation (error) of predicted output by the assumed model from the true value of the output. We write this as

$$C(w,b) = \frac{1}{N}\sum_{i=1}^{N}[y_i - (wx_i + b)]^2 \qquad \text{Eqn. (3.8)}$$

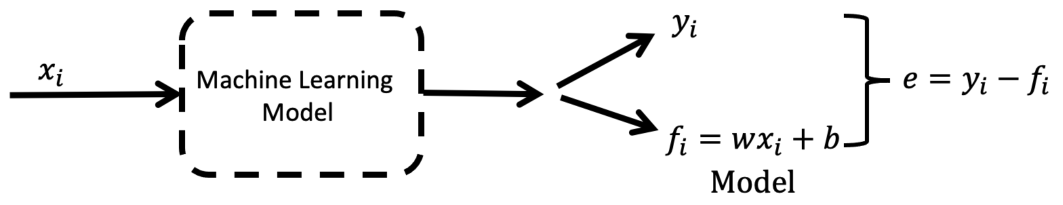
- A one-time error, $e = y_i - (wx_i + b)$, for a data pair (x_i, y_i) is called a loss function, but the average of the squared sum is called the cost function.

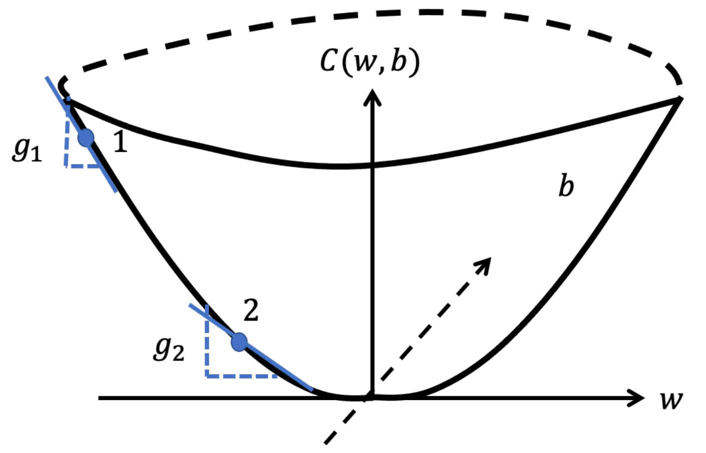
- The most optimal values of w and b should lead to a minimum of the cost function, which is the goal.

Minimization of cost function

- The error in Eqn. (11.2) is squared, which means it is a convex function. Convex functions always have a minimum with respect to relevant parameters. With two parameters to optimize, the method of gradient descent turns out to be a reasonable approach to take for locating the minimum, iteratively.

- An accurate plot of the cost function, $C(w, b)$, would be a three-dimensional plot. We illustrate the qualitative form of the graphic on the right. From this, it is evident that to find the minimum, one must search for a coordinate point (w, b) that minimizes $C(w, b)$. This is best done through gradient descent.





Gradient descent

- On the paraboloid cost function graphic in the previous slide, one must determine which way to go on the "hill" (up or down) toward the minimum, and how big a step to take when we decide on which direction to move. Once one determines the gradient at position 1 on the paraboloid graphic of the previous slide, one can take fairly large steps down the gradient. At position 2, one computes a slope that is much smaller than at position 1, there is therefore a need to reduce the step size so that one does not miss the minimum and wind up on the other side of the valley.
- From our cost function of

$$C(w,b) = \frac{1}{N}\sum_{i=1}^{N}[y_i - (wx_i + b)]^2 \qquad \text{Eqn. (3.9),}$$

- The gradient is found in the manner indicated below, where the chain rule of differentiation has been invoked.

$$C'(w,b) = \begin{bmatrix} \frac{\partial C}{\partial w} \\ \\ \frac{\partial C}{\partial b} \end{bmatrix} = \begin{bmatrix} -\frac{2}{N}\sum_{i=1}^{N} x_i\big(y_i - (wx_i + b)\big) \\ \\ -\frac{2}{N}\sum_{i=1}^{n}\big(y_i - (wx_i + b)\big) \end{bmatrix} \qquad \text{Eqn. (3.10)}$$

Gradient descend and learning rate

- In gradient descent, one must pick a step-size by which to move. This is the learning rate. Beginning with randomly chosen values for the parameters w and b, the goal is to adjust them such that the values that reduce the cost function are found. The sign of the gradient indicates the direction in which we must update to reduce the cost function. One must move in a direction opposite to that of the gradient.

- For each iteration of the parameters, one has

$$\left.\begin{aligned} w &= w - \alpha . \frac{\partial C}{\partial w} \\ b &= b - \alpha . \frac{\partial C}{\partial b} \end{aligned}\right\} \quad \text{Eqn. } (3.11),$$

where α is the learning rate (step size).

- For each iteration, one must recalculate the cost function to check that it is decreasing. One can stop the gradient descent process when the cost function is at its minimum. Note that in Eqn. (3.9), one must find the gradients of the cost function by the chain rule of differentiation.

Cost Function for Linear Regression

- The predictive model we have been discussing can be written as

$$y = wx + b, \quad \text{Eqn. (3.12)}$$

- Where y is the sales, x the advertising amount, and b the bias.

- The mean square error (MSE) can therefore be computed from

$$MSE = f(w,b) = \frac{1}{N}\sum_{i=1}^{N}[y_i - (wx_i + b)]^2 \quad \text{Eqn. (3.13)}$$

- We can optimize our choices for w and b by taking the derivative of the MSE function above to get

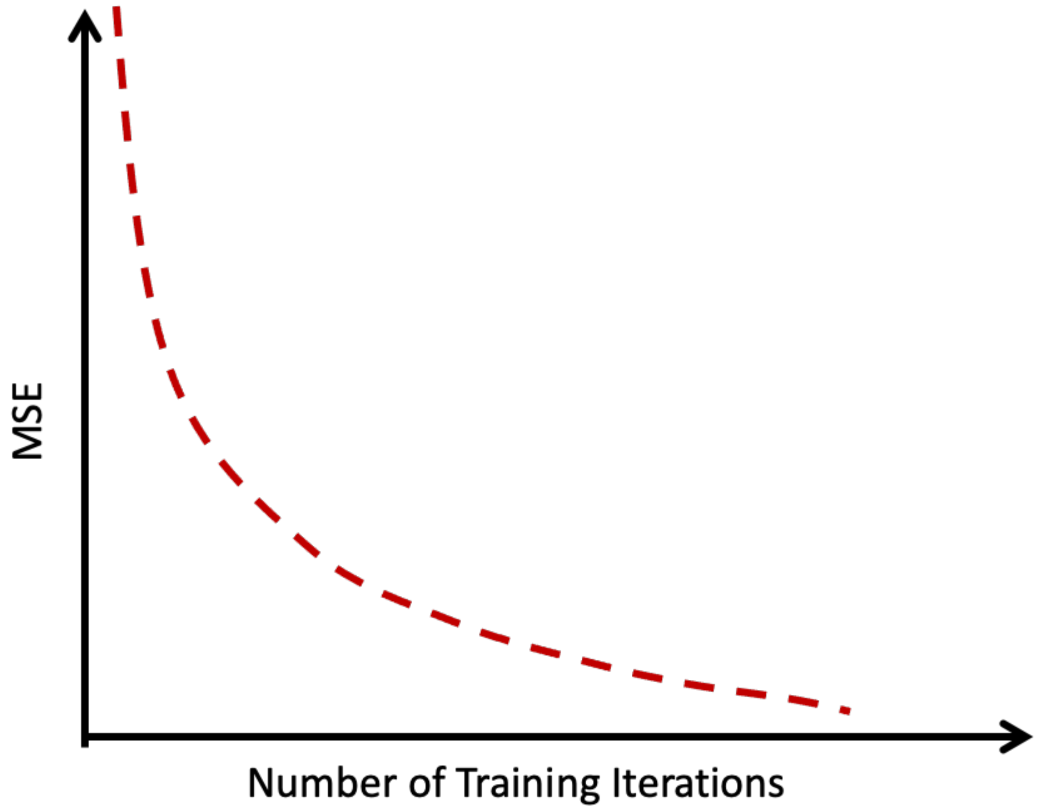
$$f'(w,b) = \begin{bmatrix} \frac{\partial f}{\partial w} \\ \frac{\partial f}{\partial b} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum_i -2x_i\big(y_i - (wx_i + b)\big) \\ \frac{1}{N}\sum_i -2\big(y_i - (wx_i + b)\big) \end{bmatrix} \quad \text{Eqn. (3.14)}$$

- The sign of the derivative (gradient) tells us which direction we should update to reduce the cost function. We move in direction opposite to that of the gradient. Size of the update is controlled by the learning rate.

Training of the Linear Regression model

- To train the model, we iteratively loop through the dataset, each time updating the weight "w" and the bias "b" in the direction indicated by the sign of the slope of the cost function. Training is accomplished when the error (cost) function is at its minimum or when the training iterations fail to reduce the cost function.

- At the beginning of the training, the weight w and bias b are initialized to some random values (default values). The hyper parameters, which in this case are the learning rate and the number of iterations, must also be set at the beginning of training.
- One way to track progress is to plot the MSE as a function of training iterations (see sketch on right).



Multivariate Regression

- If the computer companies we discussed earlier, advertised in several places: radio, TV, and web, then the model must be extended to all relevant variables; the sales function is now written as

$$y = w_1(radio) + w_2(TV) + w_3(web) + \mathrm{b} \quad \text{Eqn. (3.15).}$$

- For convenience, we set the bias term to $b = 0$. The cost function is therefore

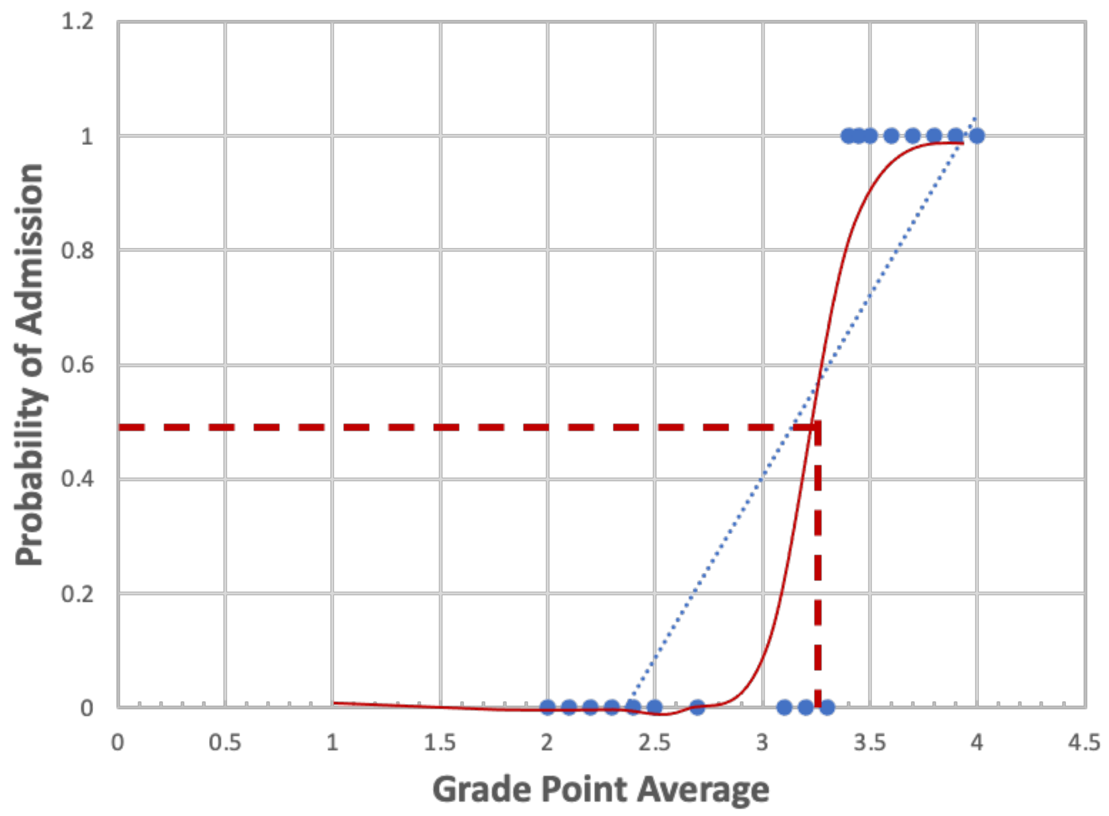
$$f = MSE = \frac{1}{2N}\sum_{i=1}^{N}[y_i - (w_1x_{1i} + w_2x_{2i} + w_3x_{3i})]^2 \quad \text{Eqn. (3.16).}$$

- We have divided by 2N so that when we take the derivative, the 2 from the differentiation cancels the 2 from the $2N$.
- The gradient of Eqn. (3.16) is a vector of partial derivatives given by

$$\begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \frac{\partial f}{\partial w_3} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\left[-x_{1i}\left(y_i - (w_1x_{1i} + w_2x_{2i} + w_3x_{2i})\right)\right] \\ \frac{1}{N}\left[-x_{2i}\left(y_i - (w_1x_{1i} + w_2x_{2i} + w_3x_{3i})\right)\right] \\ \frac{1}{N}\left[-x_{3i}\left(y_i - (w_1x_{1i} + w_2x_{2i} + w_3x_{3i})\right)\right] \end{bmatrix} \quad \text{Eqn. (3.17)}$$

Regression on data sets with binary outcomes

- Undergraduate admissions offices at some universities have a tool that does a rough-cut admission.
- Using only GPA and standardized test scores (SAT scores), automated admission tools can narrow down the applicant pool so that the human admissions committees can focus on reading references and essays for only a few students.
- Input to the automated admission tool is continuous but the output is binary. Clearly the linear regression method we have studied will not work here. However, transforming the input data so that the output is binary is an option. This is binary classification and a type of approach called logistic regression is the preferred method.



Logistic Regression: Conditional Probabilities

- Sometimes the models we want should give discrete and not continuous outputs. Examples of these kind include questions like:
  - Will a person repay a loan or not (yes/no), given certain things you know about them?
  - Will it snow tomorrow, given that today is sunny and yesterday was sunny?
  - Will I get an "A" in this class given that I have done all the HW and passed all the quizzes with scores of over 80%?
- The task we have at hand is to analyze data that pertains to questions like those above. The output is clearly binary (yes/no). These types of situations are important in machine learning.
- We need to have a conditional distribution of the output, given the input variables: in another words, we seek $P(Y|X)$.
- To link the input variable to the probability, we introduce a quantity called odds: this is the ratio of the probability of an event happening to the probability that it will not happen;

$$\text{Odds} = \frac{p}{1-p} \quad \text{Eqn. (3.18).}$$

Odds Ratio and its Relationship to Logit

- Probabilities are distributed between 0 and 1 but input variables are generally not. We cannot link the odds ratio directly to a linear combination of independent variables because it does not make sense.

- The best strategy is to consider the natural logarithm of the odds ratio and link that to the linear combination of input variables, thus

$$ln\left(\frac{p(x)}{1-p(x)}\right) = \sum_{i=0}^{k} a_i x_i \quad \text{Eqn. (3.19);}$$

- The simplest case of this linear combination could be

$$a_0 + a_1 x_1 = \sum_{i=0}^{1} a_i x_i \quad \text{Eqn. (3.20)}$$

- We have set $x_0 = 1$ to permit adding a bias term a_0. The left-hand side of Eqn. (3.19) is called the logit of $p(x)$, which is where the term logistic regression comes from.

- Eqn. (3.19) can be rewritten (using the relationship between exponentials and natural logarithms) as

$$\frac{p(x)}{1-p(x)} = \exp\left(\sum_{i=0}^{k} a_i x_i\right) = \prod_{i=0}^{k} \exp(a_i x_i) \quad \text{Eqn. (3.21).}$$

Logistic Regression

- Eqn. (3.21) suggests that logistic models are multiplicative in their inputs; the value of $\exp(a_i)$ tells us how the odds of the output being true increase or decrease as x_i increases by one unit.

- For example: if $a_i = 0.693$, then $\exp(0.693) = 2$. If x_i is a numerical variable such as someone's weight in pounds, then every increase in weight by one pound, doubles the odds of that person being overweight (as the output), if other things remain the same.

- One can invert Eqn. (3.21) as follows:

$$p(y) = \frac{\exp(y)}{1+\exp(y)}, \text{ where } y = \sum_{i=0}^{k} a_i x_i \quad \text{Eqn. (3.22).}$$

- Finally, we have a function linking the real number line to the probability interval between 0 and 1, [0,1].

- By the chain rule of differentiation, the derivative of $p(y)$ in Eqn. (3.22) is

$$p'(y) = p(y)(1 - p(y)) \qquad \text{Eqn. (3.24).}$$

Solution of a Logistic Regression Problem

- If required, the gradient of $p(.)$ with respect to the coefficients a_i can be obtained as

$$\frac{\partial p}{\partial a} = p(y)(1 - p(y))\frac{\partial y}{\partial a} \quad \text{Eqn. (3.25).}$$

- A solution to a logistic regression problem is therefore the set of parameters a_i that maximizes the likelihood of the data x_i.

- One can express the solution as a product of the predicted probabilities of the k individual observations, thus

$$\mathcal{L}(x|p) = \prod_{i=1|x_i=1}^{k} p(x_i) \prod_{i=1|x_i=0}^{k} (1 - p(x_i)) \quad \text{Eqn. (3.26).}$$

Likelihood function for logistic regression

- Input to the logistic model is a vector of features, x_i, and the output is a class y_i. The probability of a class has probability p for $y_i = 1$ and $1 - p$ for $y_i = 0$. We define the likelihood as

$$\mathcal{L}(b_o, b) = \prod_{i=1}^{k} p(x)^{y_i}(1 - p(x))^{1-y_i} \qquad \text{Eqn. (3.27);}$$

- The log-likelihood is then obtained by taking the natural log of Eqn. (3.27) to give

$$\begin{aligned} \ell(b_0, b) &= \sum_{i=1}^{k} y_i \log p(x_i) + (1 - y_i)\log(1 - p(x_i)) \\ &= \sum_{i=1}^{k} y_i[\log p(x_i) - \log(1 - p(x_i))] + \log(1 - p(x_i)) \qquad \text{Eqn. (3.28)} \\ &= \sum_{i=1}^{k} y_i \log\left(\frac{p(x_i)}{1 - p(x_i)}\right) + \log(1 - p(x_i)) \\ &= \end{aligned}$$

Maximizing the likelihood

- The log-likelihood equation from the the previous slide is

$$\ell(b) = \sum_{i=1}^{k} y_i(b_0 + b_i x_i) - \sum_{i=1}^{k} \log(1 + \exp(b_0 + b_i x_i)) \quad \text{Eqn. (3.29).}$$

- To maximize Eqn. (3.29), we need to differentiate with respect to some b_i and set it to zero, thus

$$\frac{\partial \ell}{\partial b_i} = \sum_{i=1}^{k} y_i x_i - \sum_{i=1}^{k} \frac{\exp(b_0 + b_i x_i)}{1 + \exp(b_i + b_i x_i)} x_i = \sum_{i=1}^{k} (y_i - p(x_i: b_0, b_i))x_i \qquad \text{Eqn. (3.30).}$$

- Eqn. (3.30) can only be solved numerically by iteration to find the values of b_0, b_i that make the log-likelihood a maximum.

Decision boundary

- Once the coefficients, b_0, b_i of the probability function of the logistic have been determined through a learning process, given a data set, one is ready to predict a new output for a given input, x_i through

$$p(x) = \frac{\exp(b_0 + b_i x_i)}{1 + \exp(b_0 + b_i x_i)} = \frac{1}{1 + \exp[-(b_0 + b_1 x_i)]} \quad \text{Eqn. (3.31).}$$

- The associated class for input x_i, is given according to

$$f(x) = \begin{cases} 0 & p(x) \leq 0.5 \\ 1 & p(x) > 0.5 \end{cases} \quad \text{Eqn. (3.32).}$$

- One could also focus attention on the logit $p(x) = \sum_{i=0}^{k} b_i x_i$ Eqn. (3.33), which leads to

$$f(x) = \begin{cases} 0 & b_i x_i \leq 0 \\ 1 & b_i x_i > 0 \end{cases}$$

- The decision boundary is then: $b_0 + b_1 x_1 + \cdots b_k x_k = 0$, which is a point for $k = 1$, and a line for $k = 2$, and a $(k-1)$ dimensional subspace.

Summary

- Reviewed the concept of learning
  - Learning in machines
  - Classification

- Introduced regression in machine learning
  - Linear models in learning
  - Logistic regression: mapping continuous input variables to probability