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

Introduction to Machine Learning

Lecture 04

2021.09.14.

Agenda

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

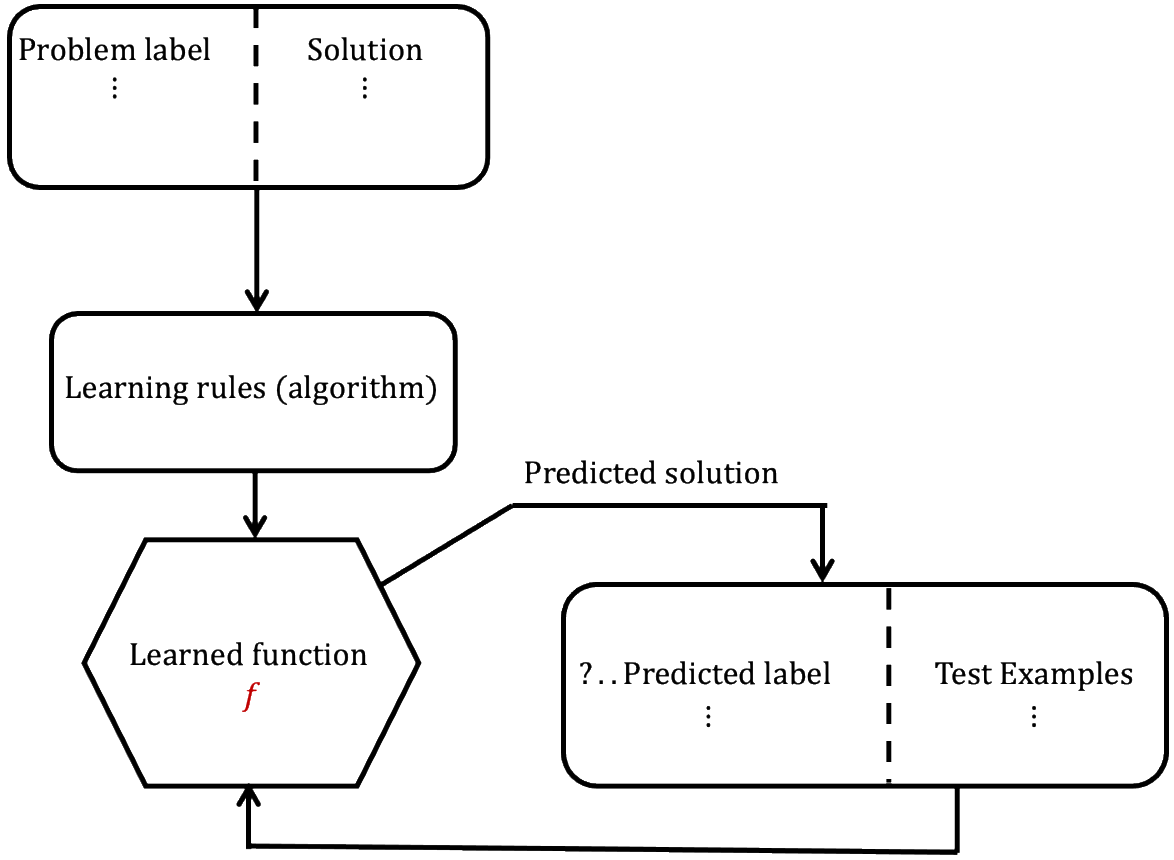
- Regression
  - Linear regression
  - Logistic regression

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. (4.1)

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

$$\mathcal{L}(y, \tilde{y}) = \begin{cases} 0 \text{ if } y = \tilde{y} \\ 1 \text{ oherwise} \end{cases} \quad \text{Eqn. (4.2)}$$

- 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

$$E(x, y) \longrightarrow D\big[\mathcal{L}\big(y, f(x)\big)\big] \quad \text{Eqn. (4.3)}$$

- 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

$$E(x, y) \longrightarrow D\big[\mathcal{L}\big(y, f(x)\big)\big] = \sum_{(x,y)\in D}\big[D(x, y)\mathcal{L}\big(y, f(x)\big)\big] \quad \text{Eqn. (4.4)}$$

- 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

$$E(x, y) \longrightarrow D\big[\mathcal{L}\big(y, f(x)\big)\big] = 1/N \sum_{n=1}^{N}\big[\mathcal{L}\big(y_n, f(x_n)\big)\big] \quad \text{Eqn. (4.5)}$$

Classification

- Classification is determination of a target function f that assigns (maps) an attribute or set of attributes x to predefined class labels y.

- It is a tool for distinguishing between objects of different classes, or a tool that can also be used to predict unknown records.

- Classification methods are best for predicting data sets with binary or nominal categories.

- Classifiers are created using
  - Rule-based methods;
  - Support vector machines;
  - Bayes statistics;
  - Tree-based approaches or
  - Neural networks

Evaluation of Classifiers

- One evaluates a classifier model on the counts of test data it correctly and incorrectly predicts.
- The accuracy of a classifier is quantified as

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

- The error rate is defined similarly as

$$\text{Error Rate} = \frac{\text{Number of incorrect predictions}}{\text{Total number of predictions}}.$$

- Classification models can usually serve as explanatory tools for objects of different classes.

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.

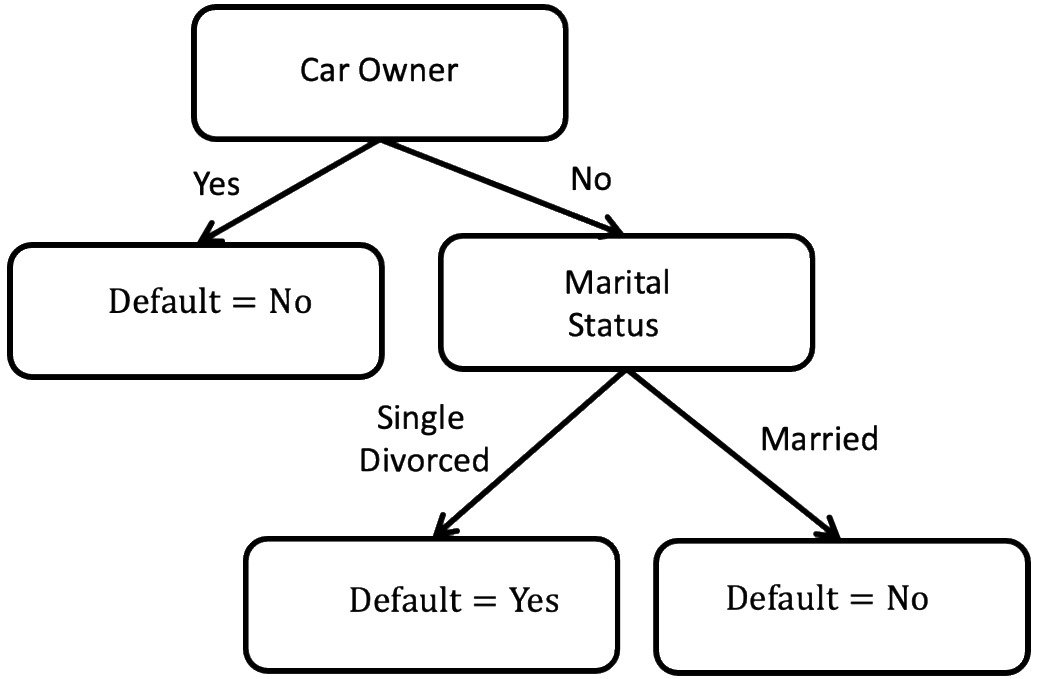
Hunt's Algorithm for Decision Tree Creation

- 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.

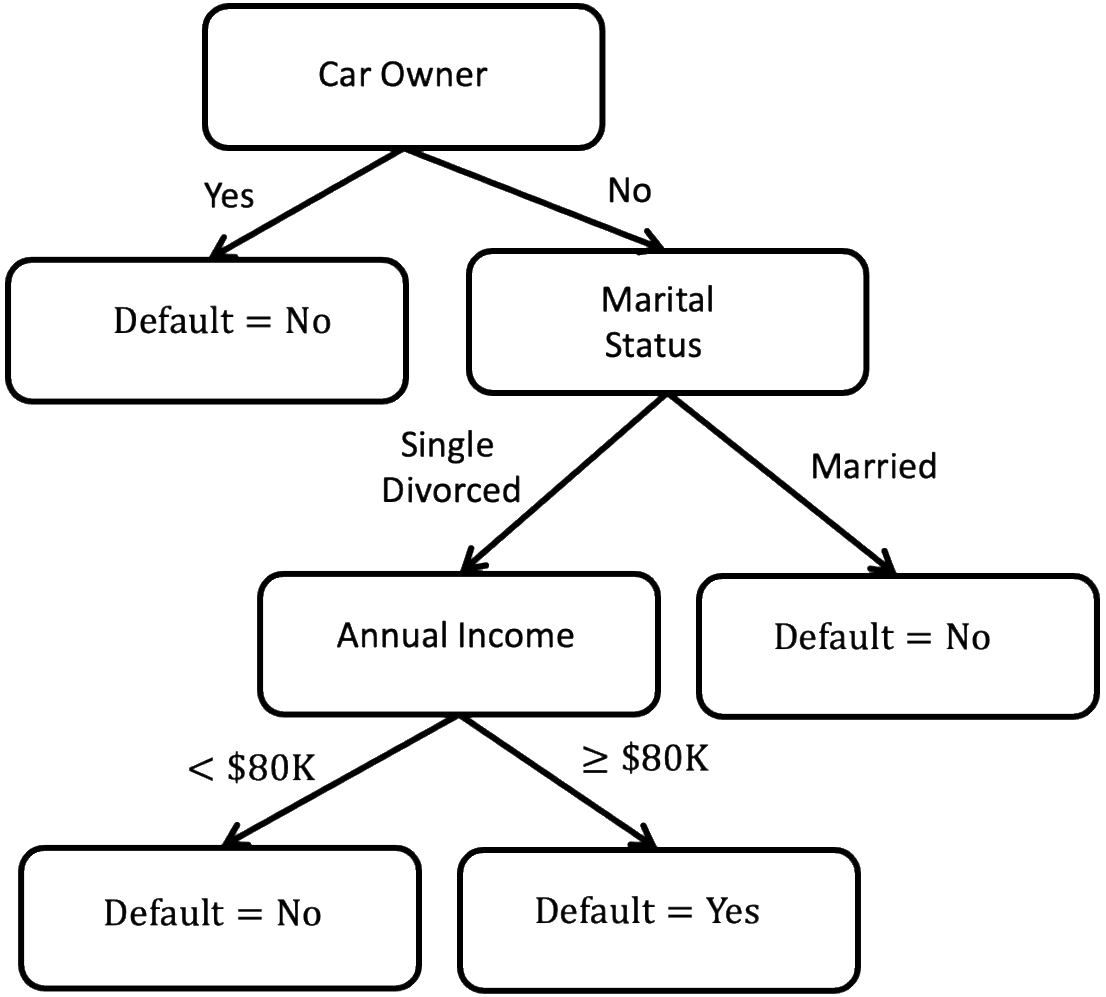
Bank Data from Customers and Creation of a Decision Tree

- 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



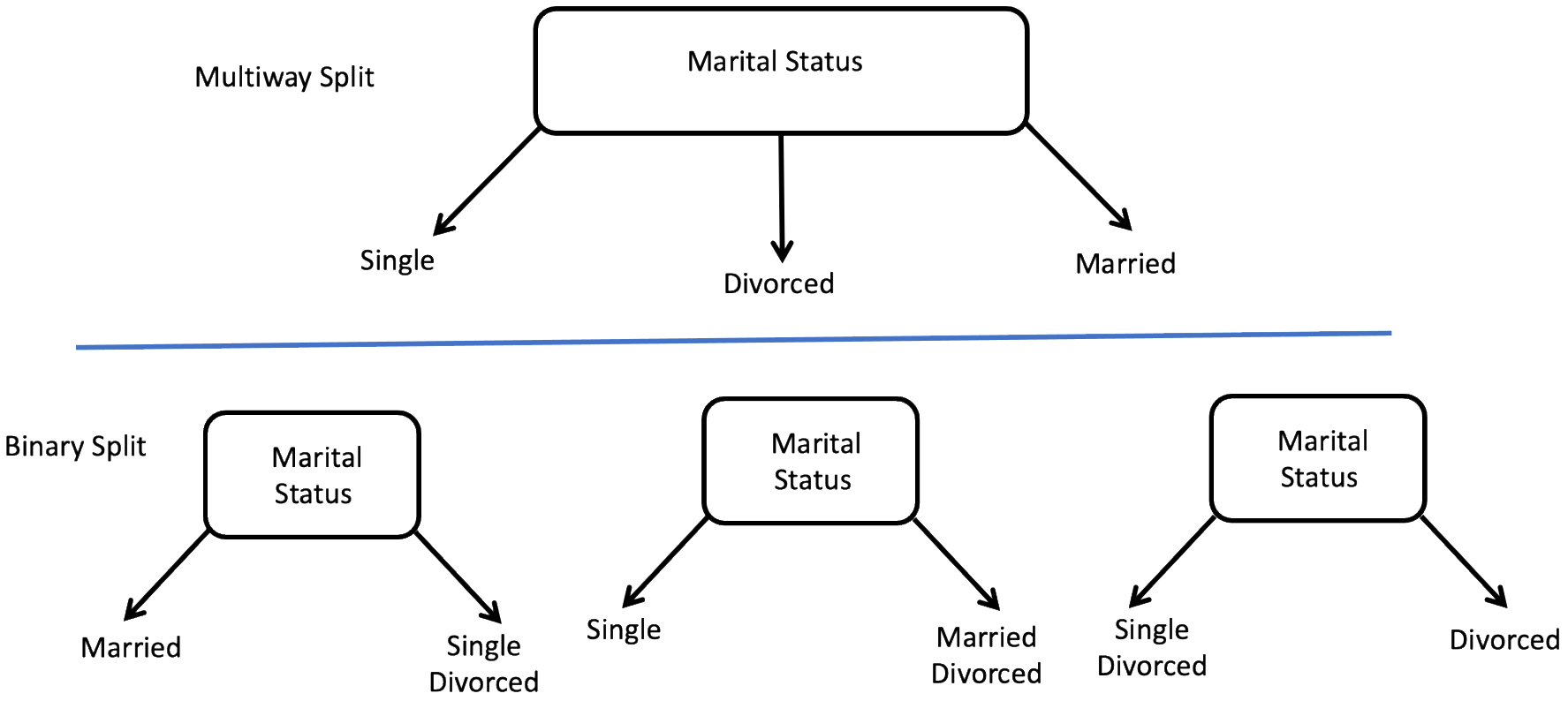
- The data also suggests that we consider income for single, divorced people. Resulting tree would be as shown on left.
- Hunt's algorithm works if every combination of attributes is present in training data and each combination has a unique label. If a child node in step 2 is empty, then it is declared 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 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

$$y = wx + b \text{ Eqn. } (4.6).$$

- 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 amounts 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.

Cost Function for Linear Regression

- The predictive model in the previous slide can be written as

$$y = wx + b, \quad \text{Eqn. (4.7)}$$

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

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

$$MSE = f(w,b) = \frac{1}{N}\sum_{i=1}^{N}[y_i - (wx_i + b)]^2 \quad \text{Eqn. (4.8)}$$

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

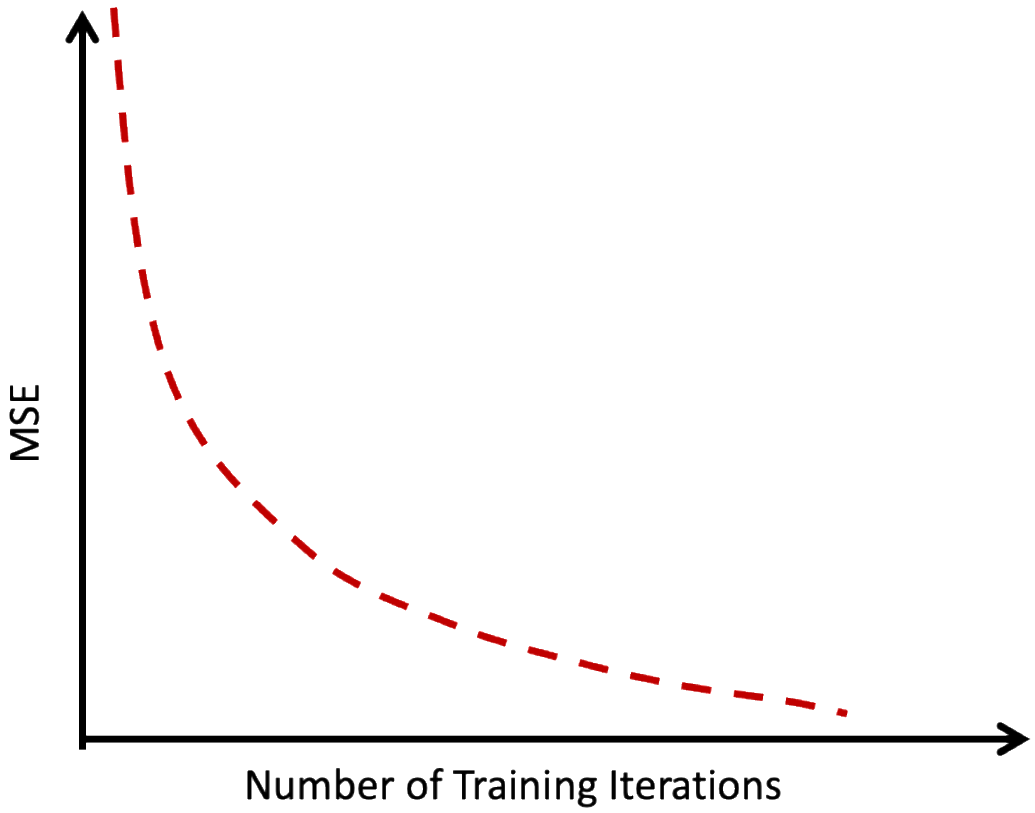
$$f'(w,b) = \begin{bmatrix} \frac{\partial f}{\partial w} \\ \frac{\partial f}{\partial b} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum_i -2x_i\left(y_i - (wx_i + b)\right) \\ \frac{1}{N}\sum_i -2\left(y_i - (wx_i + b)\right) \end{bmatrix} \quad \text{Eqn. (4.9)}$$

- 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

$$y = w_1(radio) + w_2(TV) + w_3(web) + \text{b} \quad \text{Eqn. (4.10).}$$

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

$$f = MSE = \frac{1}{2N}\sum_{i=1}^{N}[y_i - (w_1x_{1i} + w_2x_{2i} + w_3x_{3i})]^2 \quad \text{Eqn. (4.11).}$$

- 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. (4.11) is a vector of partial derivatives given by

$$\begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \frac{\partial f}{\partial w_3} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\left[-x_{1i}\left(y_i - (w_1x_{1i} + w_2x_{2i} + w_3x_{2i})\right)\right] \\ \frac{1}{N}\left[-x_{2i}\left(y_i - (w_1x_{1i} + w_2x_{2i} + w_3x_{3i})\right)\right] \\ \frac{1}{N}\left[-x_{3i}\left(y_i - (w_1x_{1i} + w_2x_{2i} + w_3x_{3i})\right)\right] \end{bmatrix} \quad \text{Eqn. (4.12)}$$

Regression Algorithm for Machine learning

- The typical task in machine learning where regression is a viable approach is stated as follows:

- Given the dataset on the right, determine the function (model) that predicts outputs for any future input data that the machine has not seen yet but is similar in kind to what it was trained on.

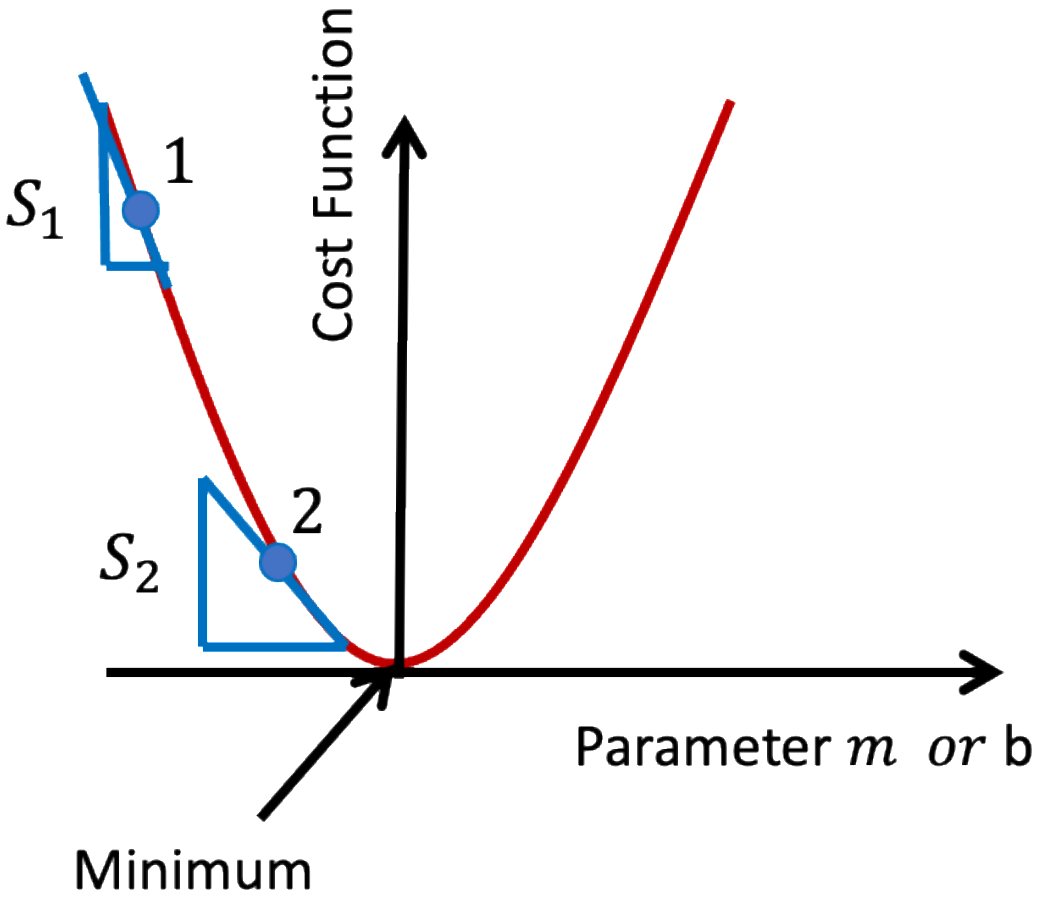
x_{in}	y_{out}
1	5
2	7
3	9
4	11
5	13

- In machine learning, we do not know the model, we must let the machine find a model with the most optimal parameters.
- For an assumed linear model, the cost function must then be

$$f(m,b) = MSE = \frac{1}{N}\sum_{i-1}^{N}[y_i - (mx_i + b)]^2 \text{ Eqn. (4.13).}$$

- The quantity in brackets is the error and is a function of m and b. A one-time error for a data pair (x_i, y_i) is called the loss function but the average sum is called the cost function.

Analysis of a Cost Function



- Notice that the error in Eqn. (4.13) is squared, which means it is a convex function; convex functions have a minimum with respect to relevant parameters.
- The method of gradient descent is the best approach for determining the minimum of the cost function iteratively by a machine.
- The error (cost) function can be plotted (sketched in this case) as illustrated on the left.
- When cost is plotted as a function of the parameters m and b, one obtains a 3D surface.

Implementation of Gradient Descent

- To implement gradient descent in our simple regression problem, we need
  - To determine which way to go on the hill (up or down) toward the minimum; and
  - Determine how big a step to take when we decide which direction to take.
- In the previous slide, if we are at position 1, and determine that the slope is of a certain value and direction, we can take fairly large steps down the slope.
  - Once we get to position 2, we realize the slope is smaller than it was at 1, and so we must take smaller steps so that we don't overshoot the minimum (bottom);
- Mathematically, our cost was defined by (4.13) as

$$f(m,b) = \frac{1}{N}\sum_{i=1}^{N}[y_i - (mx_i + b)]^2 \quad \text{Eqn. (4.13)}$$

- The slope vector was determined to be

$$\begin{bmatrix} \frac{\partial f}{\partial m} \\ \frac{\partial f}{\partial b} \end{bmatrix} = \begin{bmatrix} \frac{-2}{N}\sum_{i=1}^{N} x_i\left(y_i - (mx_i + b)\right) \\ \frac{-2}{N}\sum_{i=1}^{N}\left(y_i - (mx_i + b)\right) \end{bmatrix} \quad \text{Eqn. (4.9)}$$

Implementation of Gradient Descent

- The step sizes we take constitute the "learning rate"; for initial random values of m and b, after taking a given step size, one must modify the values of m and b. Thus

$$\left.\begin{aligned} m &= m - \alpha \frac{\partial f}{\partial m} \\ b &= m - \alpha \frac{\partial f}{\partial b} \end{aligned}\right\} \text{Eqn. (4.14), where } \alpha \text{ is the learning rate (step size).}$$

- For each iteration we make, we recalculate the cost function to make sure it is decreasing. We stop the gradient descent process when the cost function is at its lowest.

- To determine the derivatives $\partial f/\partial m$ and $\partial f/\partial b$, we must use the chain rule of differentiation; to see this explicitly, we rewrite the cost function as

$$f(m,b) = \frac{1}{N}\sum_{i=1}^{N}[y_i - y(m,b)]^2 \qquad \text{Eqn. (4.15)}$$

error

Implementing the Gradient Descent

- Cost function in Eqn. (4.15) is now an explicit function of the error, which is a function of m and b.

- The derivative vector of the cost function now becomes

$$\begin{bmatrix} \frac{\partial f}{\partial m} \\ \frac{\partial f}{\partial b} \end{bmatrix} = \begin{bmatrix} -2.\left(\frac{error}{N}\right)x_i \\ -2.\left(\frac{error}{N}\right).1 \end{bmatrix} = \begin{bmatrix} -2. < errorr > .x_i \\ -2. < error > .1 \end{bmatrix} \quad \text{Eqn. (4.16);}$$

- We have ignored the summation for clarity and have now written the error as average error $< errorr >$ because of the division by the number of instances of the data, N.

- The chain rule comes from the fact that the cost function can be written as a nested function, $f = e(y(x))$, whose derivative $f' = (\partial e/\partial y)(\partial y/\partial x)$.

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;

$$\text{Odds} = \frac{p}{1-p} \quad \text{Eqn. (4.17).}$$

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.

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

$$ln\left(\frac{p(x)}{1-p(x)}\right) = \sum_{i=0}^{k} a_i x_i \quad \text{Eqn. (4.18);}$$

- The simplest case of this linear combination could be

$$a_0 + a_1 x_1 = \sum_{i=0}^{1} a_i x_i \quad \text{Eqn. (4.19)}$$

- We have set $x_0 = 1$ to permit adding a bias term a_0. The left-hand side of Eqn. (4.19) is called the logit of $p(x)$, which is where the term logistic regression comes from.

- Eqn. (4.19) can be rewritten (using the relationship between exponentials and natural logarithms) as

$$\frac{p(x)}{1-p(x)} = \exp\left(\sum_{i=0}^{k} a_i x_i\right) = \prod_{i=0}^{k} \exp(a_i x_i) \quad \text{Eqn. (4.20).}$$

Logistic Regression

- Eqn. (4.19) 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. (4.20) as follows:

$$p(y) = \frac{\exp(y)}{1+\exp(y)}, \text{ where } y = \sum_{i=0}^{k} a_i x_i \quad \text{Eqn. (4.21).}$$

- 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. (4.21) is

$$p'(y) = p(y)(1 - p(y)) \quad \text{Eqn. (4.22).}$$

Solution of a Logistic Regression Problem

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

$$\frac{\partial p}{\partial a} = p(y)(1 - p(y))\frac{\partial y}{\partial a} \quad \text{Eqn. (4.23).}$$

- 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

$$\mathcal{L}(x|p) = \prod_{i=1|x_i=1}^{k} p(x_i) \prod_{i=1|x_i=0}^{k} (1 - p(x_i)) \quad \text{Eqn. (4.24).}$$

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