Basics of Machine Learning

By

Dr. M. Shahidur Rahman

Professor, DoCSE, SUST

Outline

- Different types of Machine Learning
- Learning process
- Performance and evaluation metrics
- Model validation
- ML algorithms and Gradient Descent

Machine Learning

- Machine learning, the application and science of algorithms that make sense of data and gives computers the ability to Learn from Data.
- Machine learning is composed of three areas: supervised learning, unsupervised learning, and reinforcement learning.
- Supervised learning relies on learning from a dataset with labels for each of the examples.
- Two types of supervised learning: classification and regression
  - **Classification** maps an input into a fixed set of categories, for example, classifying an image as either a cat or dog.
  - **Regression** problems map an input to a real number value. An example of this is to predicting the cost of your utility bill or the stock market price.

Machine Learning…

- Unsupervised learning determines categories from data where there are no labels present.
- These tasks can take the form of clustering, grouping similar items together, defining how closely a pair of items is related.
  - For example, to recommend a movie based on a person's viewing habits, we could cluster users based on what they have watched and enjoyed, and evaluate whose viewing habits most match the person to whom we are recommending the movie.

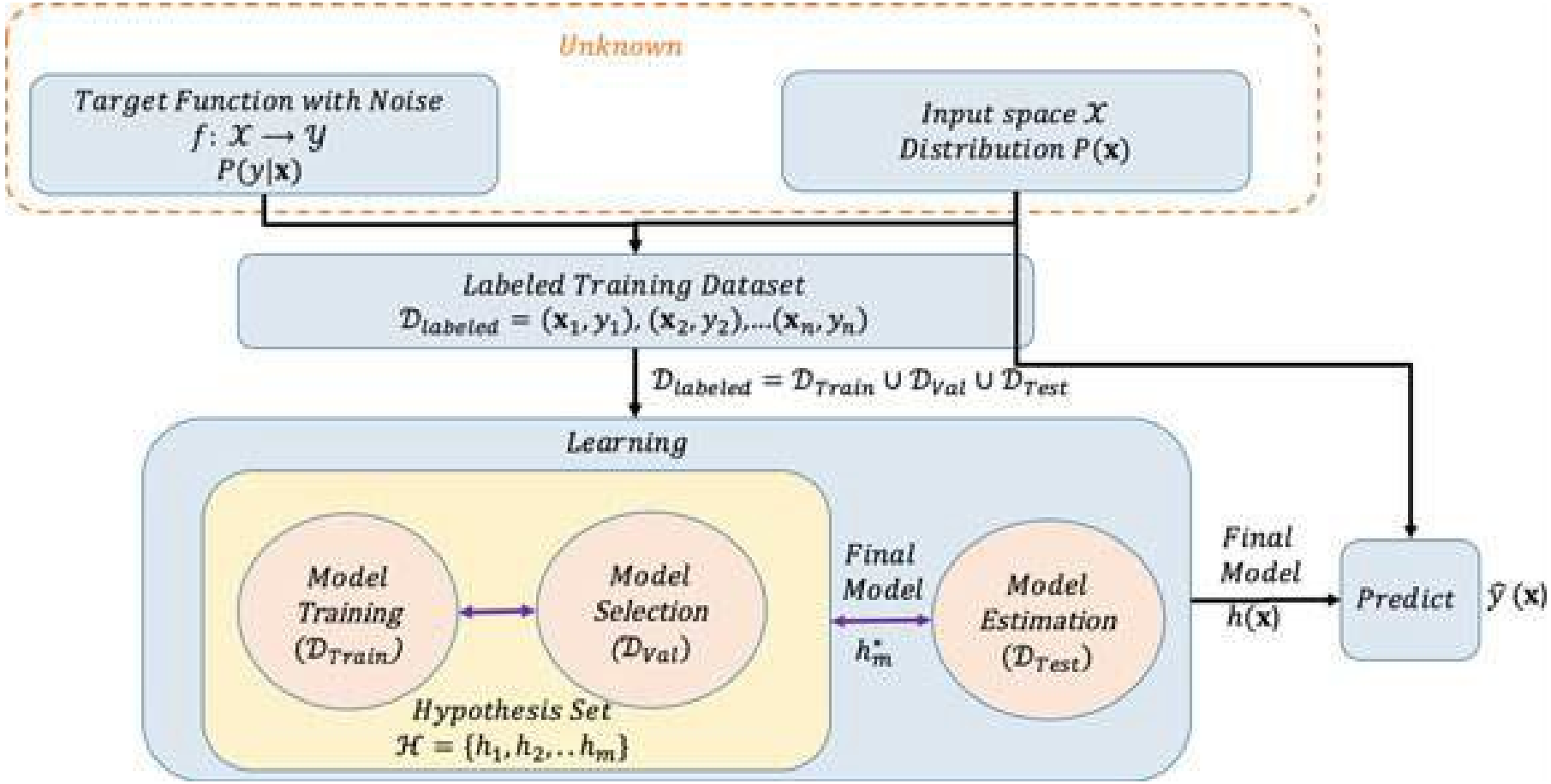
Machine Learning…

- When it is not possible to label or annotate the entire dataset due to either cost or lack of expertise or other constraints, learning jointly from the labeled and unlabeled data is called semi-supervised learning.
- Instead of expert labeling of data, if the machine provides insight into which data should be labeled, the process is called active learning.

Machine Learning…

- Transfer learning is to help the model adapt to situations it has not previously encountered.
  - This form of learning relies on tuning a general model to a new domain.
- Learning from many tasks to jointly improve the performance across all the tasks is called multitask learning.
  - These techniques are becoming the focus in both deep learning and NLP/speech.
- Reinforcement learning focuses on maximizing a reward given an action or set of actions taken. The algorithms are trained to encourage certain behavior and discourage others.
  - work well on games like chess or go, where the reward may be winning the game

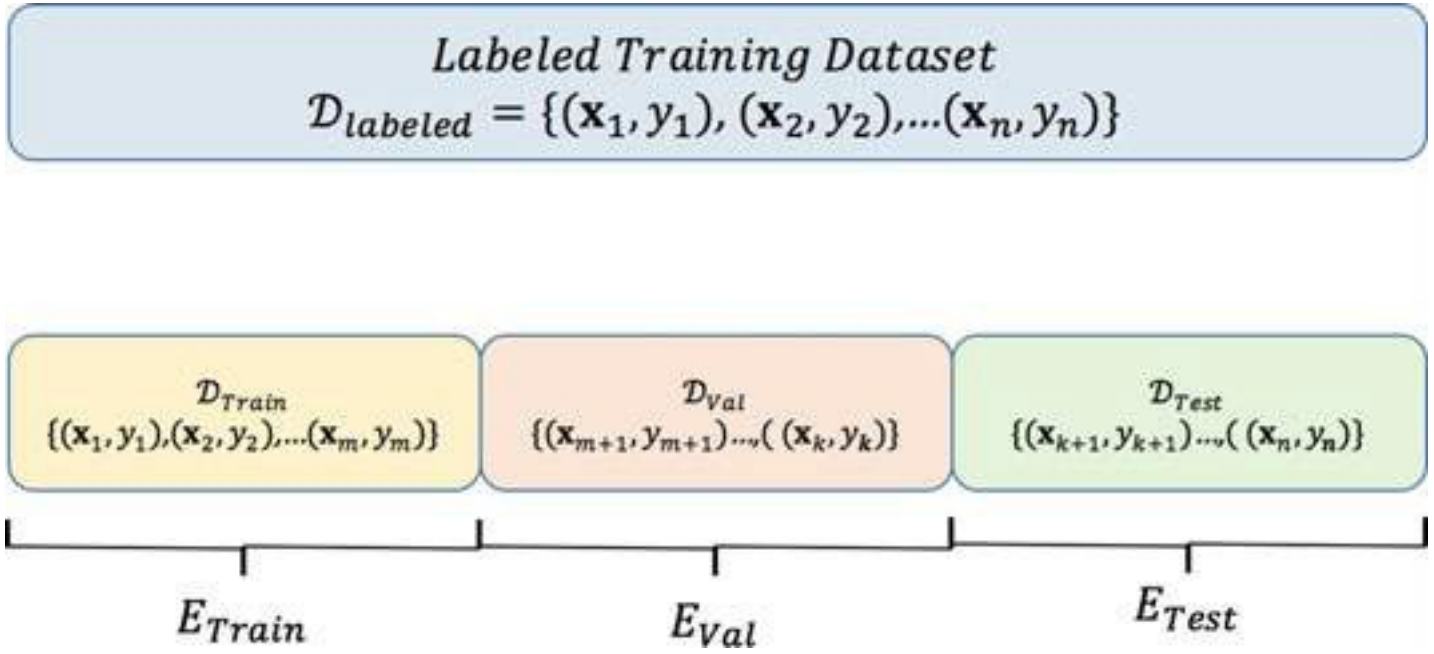
Supervised Learning: Framework and Formal Definitions



The Learning Process

Machine learning is the process that seeks to answer three questions:

1. How to train model parameters from labeled data?
2. How to select hyperparameters for a model given labeled data?
3. How to estimate the out-of-sample error from labeled data?



The Learning Process…

- The training set, D_{Train}, is used to train a given model or hypothesis and learn the model parameters that minimize the training error E_{Train}.
- The validation set, D_{Val} , is used to select the best parameters or models that minimize the validationerror E_{Val} , which serves as a proxy to the out-of-sample error.
- Finally, the test set, D_{Test}, is used to estimate the *unbiased error* of the model trained with the best parameters over D_{Val} and with learned parameters over D_{Train}.

Model Performance and Evaluation Metrics

- In classification domain, the simplest visualization of the success of a model is normally described using the **confusion matrix.**

		Predicted Class	
	Actual Class	*Class Positive*	*Class Negative*
Actual Class	*Class Positive*	*True Positive (TP)*	*False Negative (FN)*
	Class Negative	*False Positive (FP)*	*True Negative (TN)*

Evaluation Metrics

- Accuracy: $$Accuracy = \frac{TN + TP}{(TP + FN + FP + TN)}$$
- True positive rate (TPR) or recall or hit rate or sensitivity: $$TPR = \frac{TP}{(TP + FN)}$$
- Precision or positive predictive value: $$Precision = \frac{TP}{(TP + FP)}$$
- F1 Score: $$F1 = 2\frac{Precision \times Recall}{(Precision + Recall)}$$
- Specificity: $$Specificity = \frac{TN}{(TN + FP)}$$

Evaluation Metrics…

- Miss rate or false negative rate: $FNR = \frac{FN}{(TP + FN)}$
- False Positive Rate (FPR): $FPR = \frac{FP}{FP + TN}$
- Matthews correlation coefficient (MCC):

$$MCC = 2\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$$

Evaluation Metrics…

- *Accuracy* and *classification error* are informative measures of success when the data is balanced in terms of the classes
- When the data is *imbalanced*, i.e., one class is represented in larger proportion over the other class in the dataset, these measures become biased towards the majority class and give a wrong estimate of success.
- In such cases, base measures, such as *true positive rate (TPR)*, *false positive rate (FPR)*, *true negative rate (TNR)*, and *false negative rate (FNR)*, become useful.
- Metrics such as *F1 score* and *Matthews correlation coefficient (MCC)* combine the base measures to give an overall measure of success.

Evaluation Metrics…

- The curve that plots TPR and FPR for a classifier at various thresholds is known as the *receiver-operating characteristic (ROC)* curve.
- Precision and recall can be plotted at different thresholds, giving the *precision-recall curve (PRC)*
- The areas under each curve are respectively known as auROC and auPRC and are popular metrics of performance.
- In particular, auPRC is generally considered to be an informative metric in the presence of imbalanced classes.

Regression Evaluation Metrics

- Average prediction error: $\bar{y} = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)}{n}$
- Mean absolute error (MAE): $MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$
- Root mean squared error (RMSE): $RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$
- Relative squared error (RSE) is used when two errors are measured in different units:

$$RSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (\bar{y}_i - y_i)^2}$$

Regression Evaluation Metrics…

- Coefficient of determination (R^2) summarizes the explanatory power of the regression model

$$SSE_{residual} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$SSE_{total} = \sum_{i=1}^{n} (y_i - \bar{y}_i)^2$$

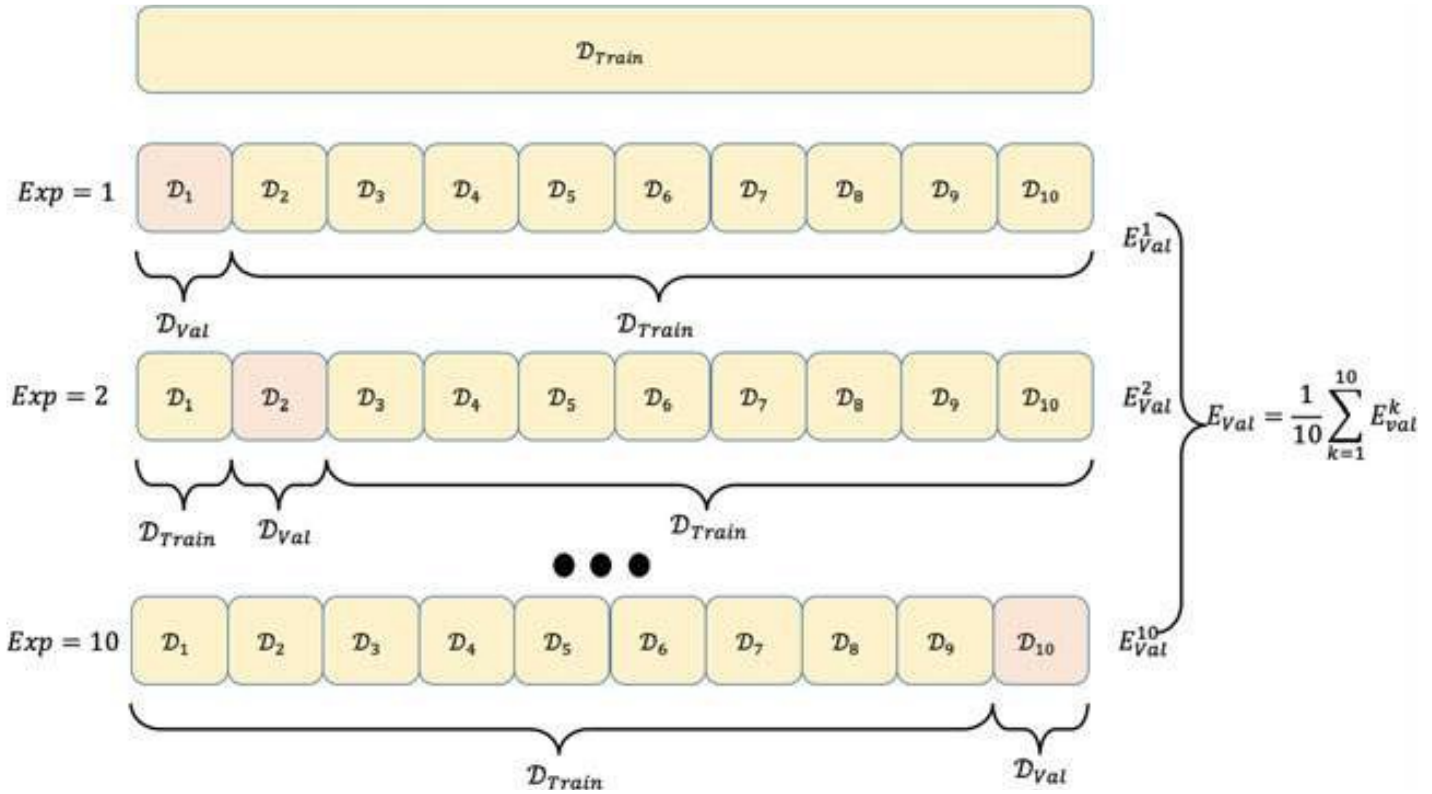
$$R^2 = 1 - \frac{SSE_{residual}}{SSE_{total}}$$

Model Validation

- Validation techniques are meant to answer the question of how to select a model(s) with the right hyperparameter values.
- When there are many hypotheses in the hypothesis set, then each unique hypothesis is trained on the training set D_{train} and then evaluated on the validation set D_{val} ; the model(s) with the best performance metrics is then chosen.

Model Validation…

- The validation process needs a large number of labeled data points for creating the training set and the validation set.
- Collecting a large labeled set is usually difficult
- In such cases, instead of physically separating the training set and validation set, ***k*-fold cross-validation** is used.



Linear Regression- A Regressive ML Model

- The dataset assumes the labels to be numeric value or a real number
- The hypothesis *h* is a linear combination of input **x** and a weight parameters **w** (that we intend to learn through training).
- In a *d*-dimensional input (**x** = [x_1, x_2 . . .x_d]), we introduce another dimension called the bias term, x_0, with value 1.

$$h(\mathbf{x}) = \sum_{i=0}^{d} w_i x_i$$

- The process of learning is represented as minimizing the squared error between the hypothesis function $h(\mathbf{x}_n)$ and the target real values y_n, as:

$$E_{train}(h(\mathbf{x}, \mathbf{w})) = \frac{1}{N} \sum_{i=0}^{d} (\mathbf{w}^{\mathrm{T}} \mathbf{x}_n - y_n)^2$$

Linear Regression…

- Since the data **x** is given, we will write the equation in terms of weights **w**:

$$E_{train}(\mathbf{w}) = \frac{1}{N}(\mathbf{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - 2\mathbf{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \mathbf{y}^{T}\mathbf{y})$$

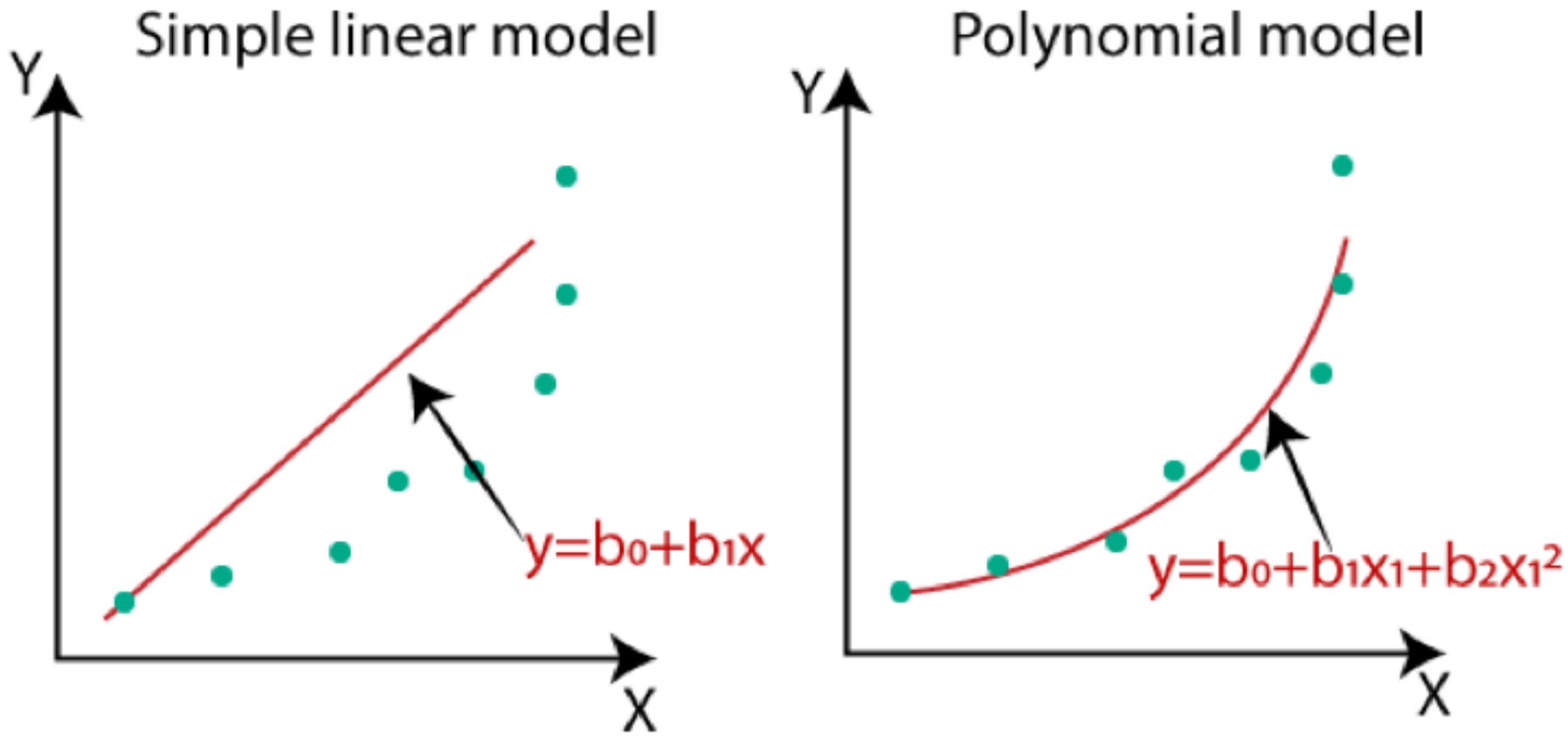
- To minimize E_{train}, we assume that the loss function $E_{train}(\mathbf{w})$ is differentiable. So, to obtain a solution, we take the gradient of the loss function with respect to **w**, and set it to the zero vector **0**:

$$\nabla E_{train}(\mathbf{w}) = \frac{2}{N}(\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - \mathbf{X}^{\mathrm{T}}\mathbf{y}) = \mathbf{0}$$

$$\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} = \mathbf{X}^{\mathrm{T}}\mathbf{y}$$

$$\mathbf{w}_{opt} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

Linear Regression…



Logistic Regression: A Classification Model

- Logistic regression can be seen as a transformation on the linear combination $\theta(\mathbf{x}^T\mathbf{w})$ that allows a classifier to return **a probability score**

$$h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$$

- A *logistic* (or a *sigmoid*) *function* $\mathbf{w}^T\mathbf{x}$ is generally used for the transformation:

$$h(\mathbf{x}) = \frac{\exp(\mathbf{w}^T\mathbf{x})}{1+\exp(\mathbf{w}^T\mathbf{x})}$$

- For a binary classification, where $y \in \{-1,+1\}$, the hypothesis can be seen as a likelihood of predicting $y = +1$, i.e., $P(y = +1 \mid \mathbf{x})$. The log-likelihood of the hypothesis can be written as:

Logistic Regression…

$$\log h(\mathbf{x}) = \sum_{i=0}^{n} \log P(y_i|\mathbf{x}_i)$$

$$\log \mathcal{L}(h(\mathbf{x})) = \sum_{i=0}^{n} \begin{cases} \log h(\mathbf{x}_i) \; if \; y_i = +1 \\ (1 - \log h(\mathbf{x}_i)) \; if \; y_i = -1 \end{cases}$$

$$\log \mathcal{L}(h(\mathbf{x})) = \sum_{i=0}^{n} (y_i \log h(\mathbf{x}_i) + (1 - y_i)(1 - \log h(\mathbf{x}_i)))$$

- The above equation is referred to as **cross-entropy** error. This cross-entropy error cannot be solved in closed form.
- Instead, an iterative algorithm known as gradient descent can be employed.

Gradient Descent

- The goal is to find weights **w** that minimize E_{train}, and that at the minimum, the gradient of E_{train} is 0.
- The negative of the gradient is followed in an iterative process until the gradient is (close to) zero. The gradient is a vector containing partial derivatives over each dimension

$$\mathbf{g} = \nabla E_{train}(\mathbf{w}) = \left[\frac{\partial E_{train}}{\partial w_0}, \frac{\partial E_{train}}{\partial w_1} \cdots \frac{\partial E_{train}}{\partial w_n} \right]$$

Gradient Descent Algorithm

Data: Training Dataset $\mathcal{D}_{train} = (\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..(\mathbf{x_n}, y_n)$ such that $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in [+1, -1]$, Loss Function $E_{train}(\mathbf{w})$, Step size η and MaxIterations T

Result: Weight vector $\mathbf{w} \in \mathbb{R}^{d+1}$

begin

$\mathbf{w}_0 \leftarrow \text{init}(\mathbf{w})$

for $t \in 0..T-1$ **do**

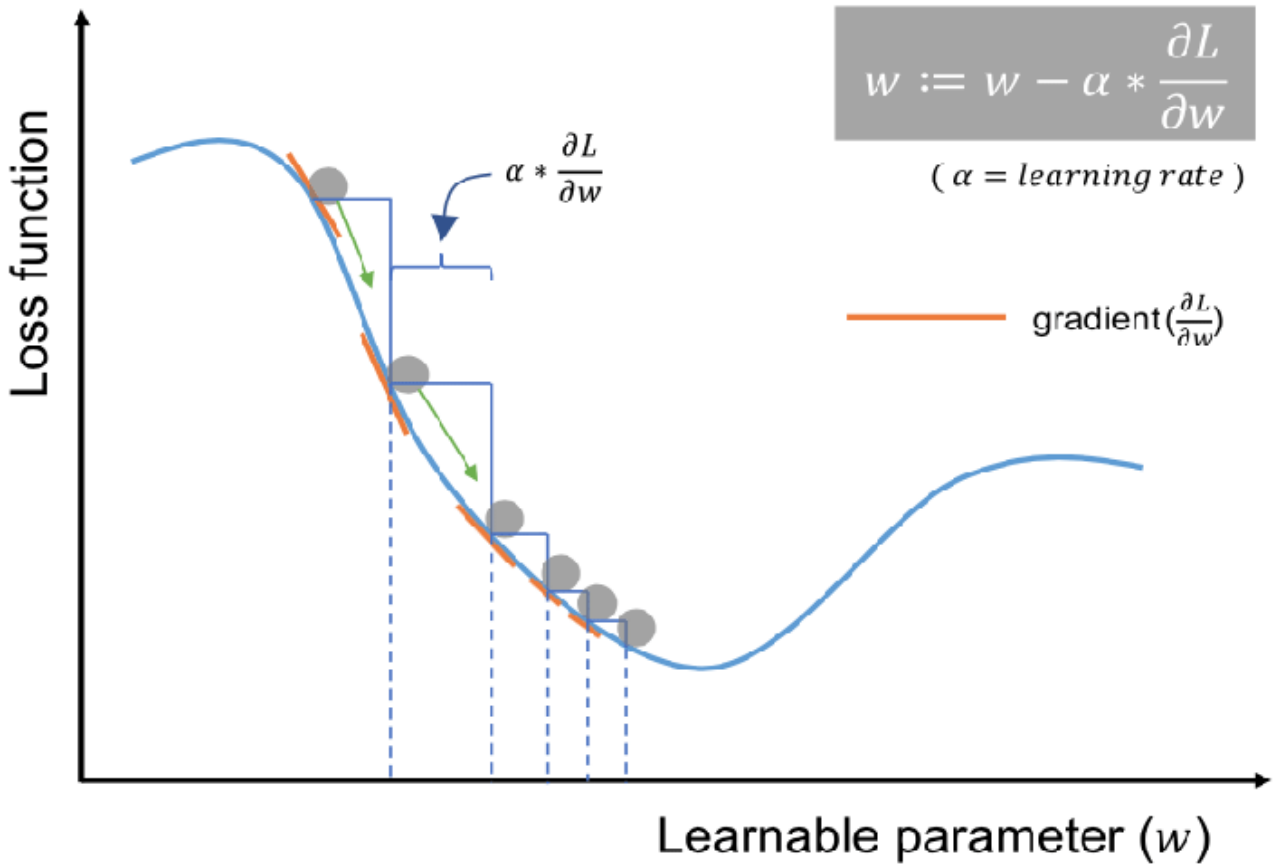
$\mathbf{g}_t \leftarrow \nabla E_{train}(\mathbf{w}_t)$

$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \hat{\mathbf{g}}_t$

return $\mathbf{w}$

- The weights **w** can be initialized to the **0** vector or set to random values.
- A small step size η is made in the direction of $-\hat{g}$, and the weights are updated accordingly, leading to an optimal point.
- Selecting a small step size is important, otherwise the algorithm oscillates and does not reach the optimum point.

How Gradient Descent Works



Stochastic Gradient Descent (SDG)

- One of the disadvantages of gradient descent is the use of the entire training dataset when computing the gradient.
- This has an implication on the memory and computation speed, which increase as the number and dimension of training examples increase.
- Stochastic gradient descent picks a data point uniformly at random from the training dataset (hence the name stochastic).
- with a large number of iterations and a small step size, SDG generally reaches the same optimum as the batch gradient descent algorithm