CS 3840
Applied Machine Learning

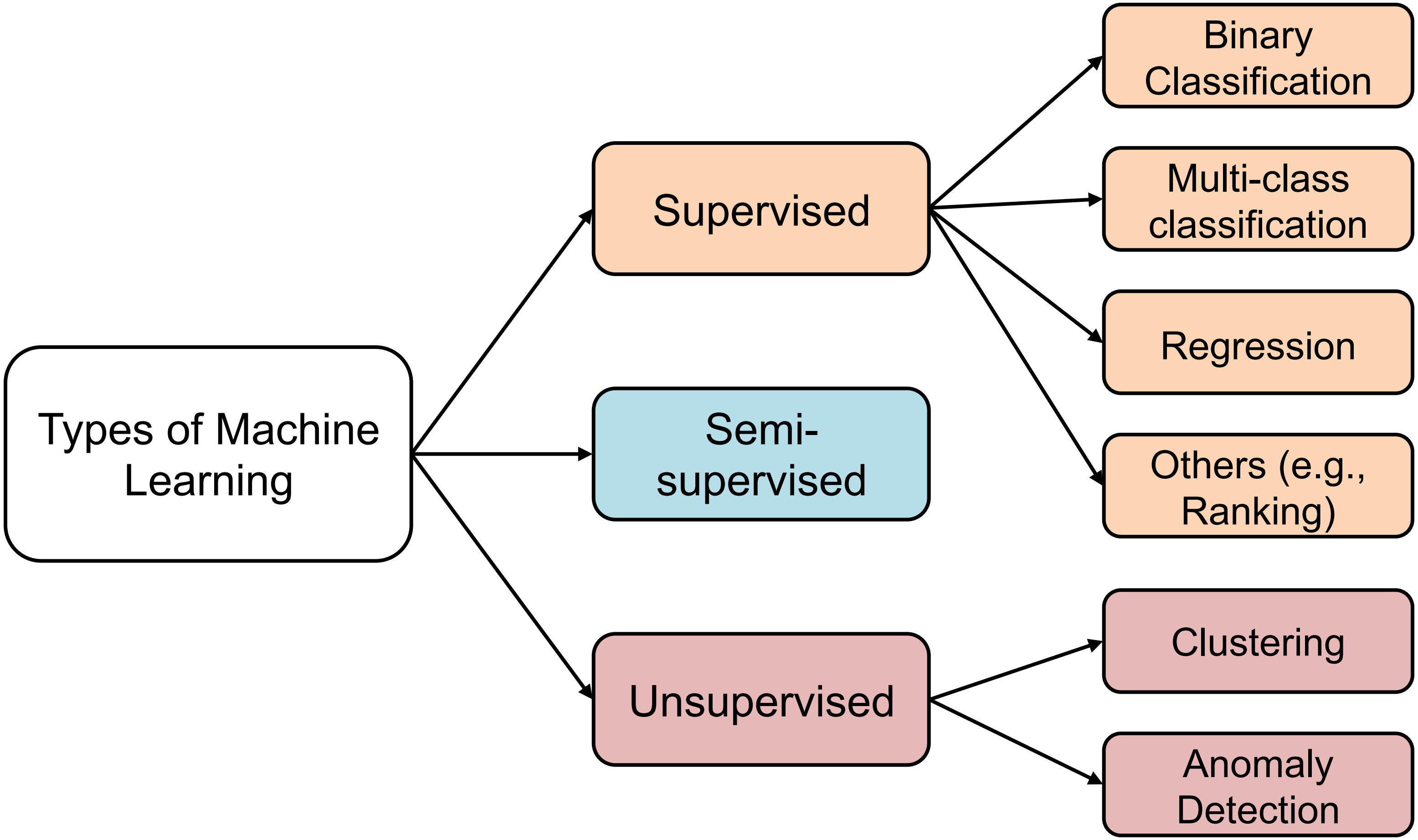
Terms and Principles of Machine Learning

Lingwei Chen

2022-01-19

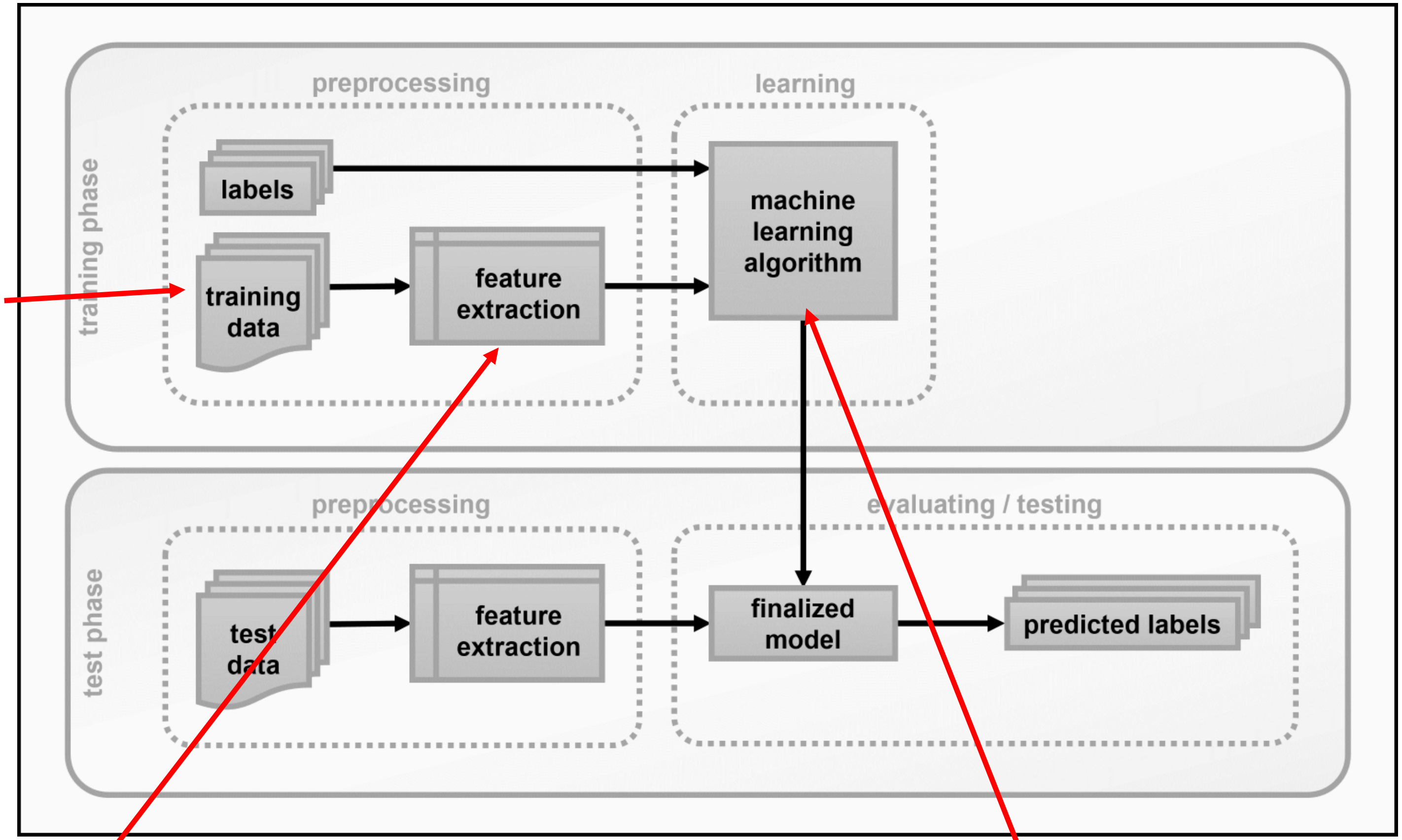
WRIGHT STATE UNIVERSITY

Review


Types of Machine Learning
Supervised
Semi-supervised
Unsupervised
Binary Classification
Multi-class classification
Regression
Others (e.g., Ranking)
Clustering
Anomaly Detection

Review

Workflow of supervised machine learning



Data collection – collect representative data for learning

Feature representation – represent the input data x in terms of something that the computer can understand

Optimization – search for the optimal settings/parameters that give the model with the best evaluation performance

Roadmap

- Linear classification
- Linear regression
- Optimization using Gradient Descent

Linear Classification

Classification Problem

- Example task – predict y, whether a string x is an email address
- Question – what features of x might be relevant for predicting y?
- Feature extraction/representation – given input x, output a set of feature name and feature value pairs

abc@gmail.com

Length > 10	1
FracOfAlpha	0.85
Contains_@	1
EndsWith_.com	1
EndsWith_.org	0

Feature Vector Notation

Mathematically, feature vector doesn't need feature names but values

Length > 10	1
FracOfAlpha	0.85
Contains_@	1
EndsWith_.com	1
EndsWith_.org	0

$$\rightarrow \begin{bmatrix} 1 \\ 0.85 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

- Feature vector – for an input x, its feature vector is

$$\phi(x) = [\phi_1(x), \ldots, \phi_d(x)], \ (\phi(x) \in R^d)$$

Feature vector can be considered as a point in d-dimensional space

Weight Vector

- Weight vector – for each feature j, have real number w_j which represents contribution of feature j to the prediction

$$w = [w_1, \ldots, w_d], \ (w \in R^d)$$

Weight vector is of the same dimension as feature vector

Length > 10	-1.2
FracOfAlpha	0.6
Contains_@	3
EndsWith_.com	2.2
EndsWith_.org	1.4

Weight vector

Linear Classification

Length > 10	-1.2
FracOfAlpha	0.6
Contains_@	3
EndsWith_.com	2.2
EndsWith_.org	1.4

Weight vector

Length > 10	1
FracOfAlpha	0.85
Contains_@	1
EndsWith_.com	1
EndsWith_.org	0

Feature vector

- Prediction score – weighted combination of feature values

$$\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}) = w_1\phi_1(x) + \cdots + w_d\phi_d(x) = \sum_{j=1}^{d} w_j\phi_j(x)$$

- Example

$$-1.2\times1 + 0.6\times0.85 + 3\times1 + 2.2\times1 + 1.4\times0 = 4.51$$

Linear Classification

- Weight vector – $w \in R^d$
- Feature vector – $\phi(x) \in R^d$

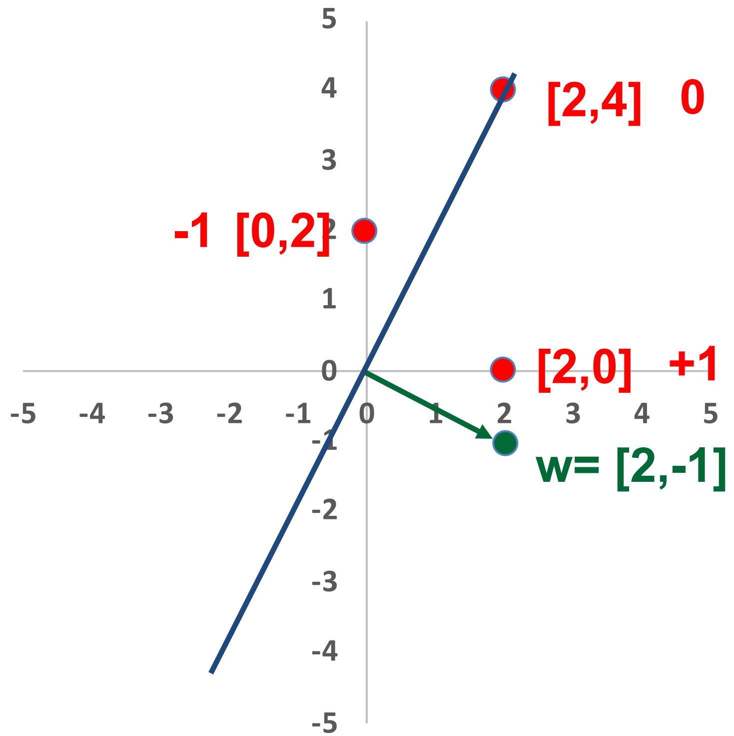
- Linear classification model for binary classification

$$y = f_w(x) = sign\left(w \cdot \phi(x)\right) = \begin{cases} +1 & if\ w \cdot \phi(x) > 0 \\ -1 & if\ w \cdot \phi(x) < 0 \\ ? & if\ w \cdot \phi(x) = 0 \end{cases}$$

Geometric Intuition – An Example

- Weight vector: $w = [2, -1]$
- Feature vectors for three input samples:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \{[2,0], [0,2], [2,4]\}$$

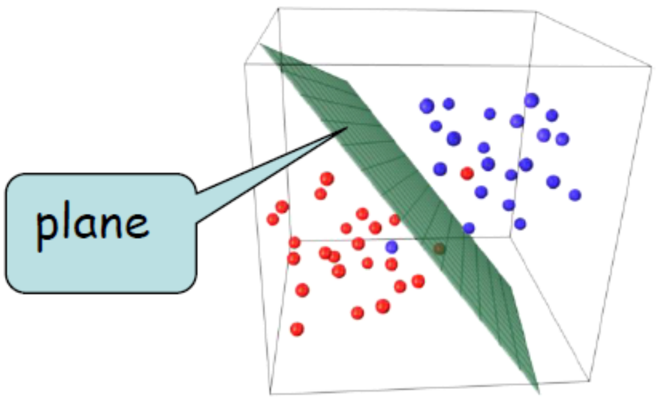


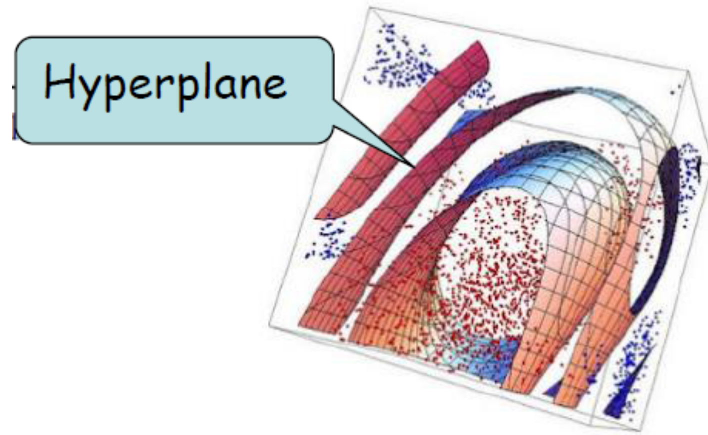
$$y = f_w(x) = sign\big(w \cdot \phi(x)\big)$$

- The binary classification model f_w defines a decision boundary with weight vector w
- The decision boundary for linear classification over two-dimensional feature vectors is a straight line

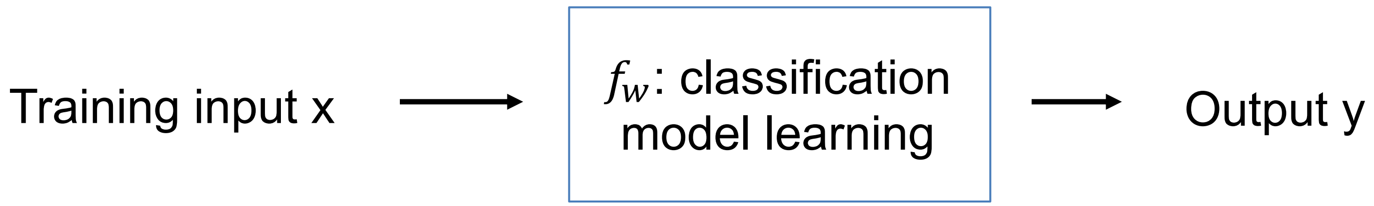
Decision Boundary

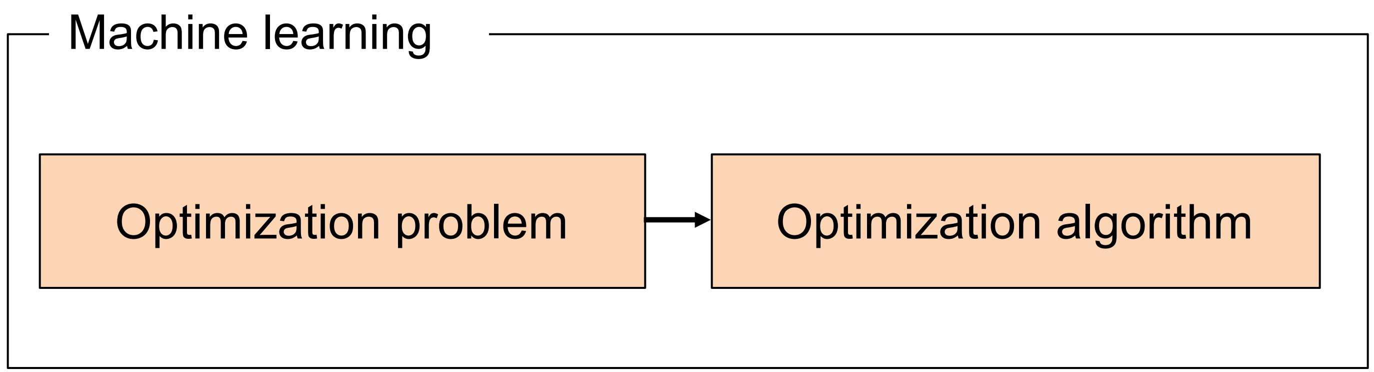
- The decision boundary – is not just for the linear classification model, but whatever classification models, the decision boundary is the separation between regions of where the classification is positive versus negative





Optimization





Loss Function

- Loss function – a loss function Loss(x, y, w) quantifies how unhappy you would be if you used w to make a prediction on x when the correct output is y. It is the objective function we want to minimize
- Loss function can be considered as the distance between the prediction and true label, which measure the performance of a classification model
- High loss is bad, and low loss is good.
- The optimization problem of a classification model seeks to minimize the loss function

Margin

- True label: y
- Classification model: $f_w(x) = sign\big(w \cdot \phi(x)\big)$
- The prediction score: $w \cdot \phi(x)$

- Margin – the margin on an example (x, y) is $\big(w \cdot \phi(x)\big)y$, specifying how correct the classification model is

- Example: w = [2, -1], $\phi(x)$ = [2, 0], y = -1

$$\big(w \cdot \phi(x)\big)y = 4\times(-1) = -4$$

- If the prediction score and the correct label have the same sign, and then the margin is positive (the prediction is correct); otherwise, the margin will be negative (the prediction is wrong)

Question

When is the binary classification model making mistake on a given example?

- Prediction score less than 0
- Margin less than 0
- Prediction score greater than 0
- Margin greater than 0

Zero-one Loss

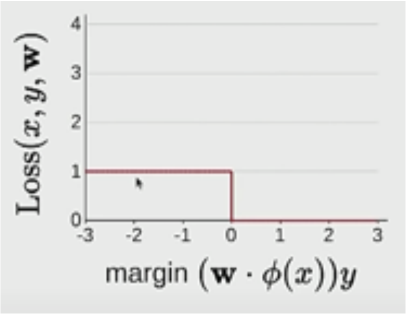
- True label: y
- Example: w = [2, -1], $\phi(x)$ = [2, 0], y = -1
- Classification model: $f_w(x) = sign\big(w \cdot \phi(x)\big)$
- Zero-one loss

$$Loss_{0-1}(x, y, w) = 1[f_w(x) \neq y]$$

$$Loss_{0-1}(x, y, w) = 1[\big(w \cdot \phi(x)\big)y \leq 0]$$

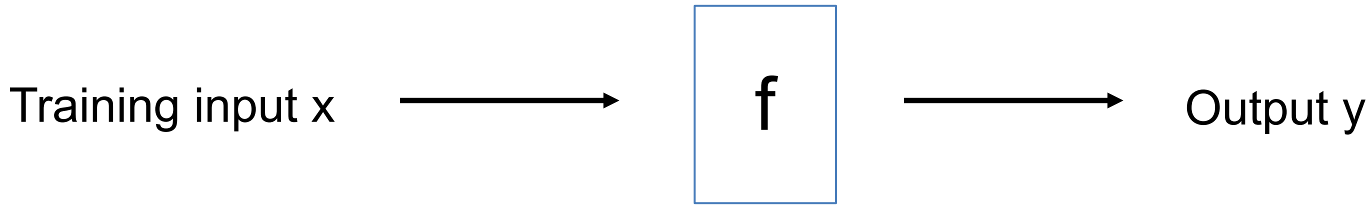
- Optimization on loss function – given a set of training examples, minimize the loss function to obtain the optimal weight vector

- Loss function is generally minimized using gradient descent, but $Loss_{0-1}(x, y, w)$ is not differentiable, and thus we cannot optimize directly on zero-one loss function



Linear Regression

Regression Problem



- Regression – $y \in R$

  The model is trained to predict input x into a real number (numeric value) in the continuous value space

- Example task –given x, predict y which is the house price

- Feature vector – $\phi(x) = [\phi_1(x), \dots, \phi_d(x)]$, $(\phi(x) \in R^d)$

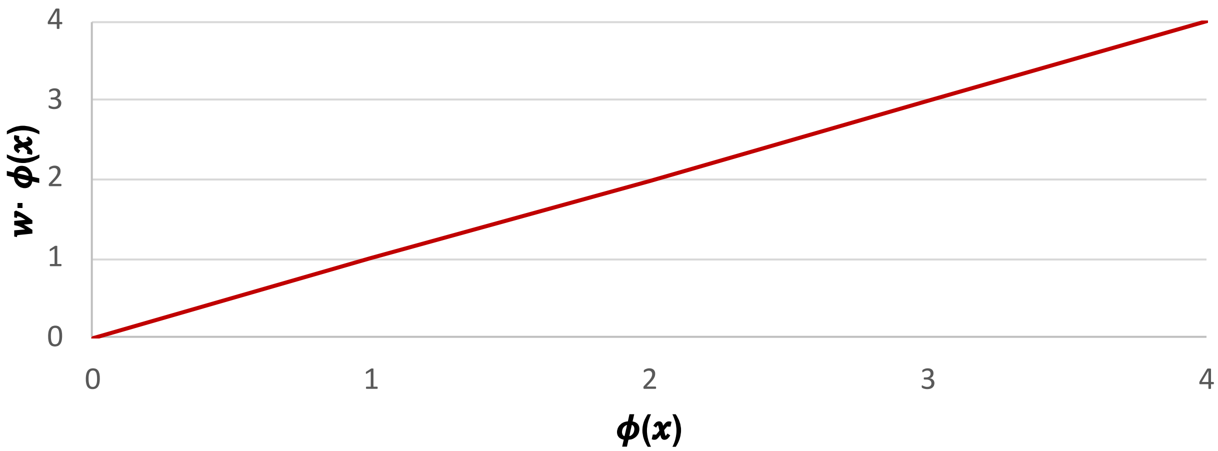
- Weight vector – $w = [w_1, \dots, w_d]$, $(w \in R^d)$

Linear Regression

- Weight vector – $w \in R^d$
- Feature vector – $\phi(x) \in R^d$
- Prediction score – $w \cdot \phi(x) = w_1\phi_1(x) + \cdots + w_d\phi_d(x)$

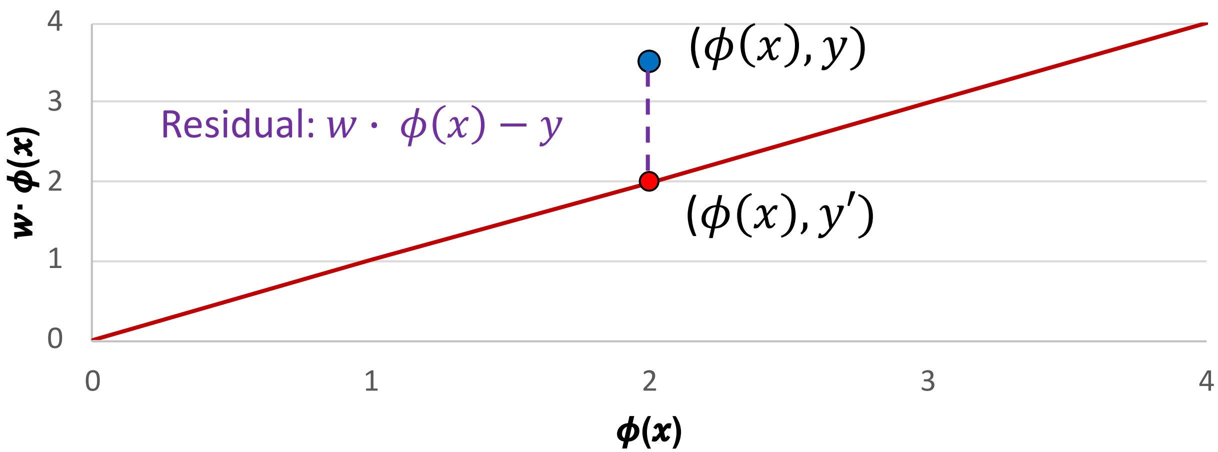
- Linear regression model

$$y = f_w(x) = w \cdot \phi(x)$$



Residual

- Input: $\phi(x)$
- True label value: y
- Predicted label value using linear regression: $y' = w \cdot \phi(x)$



- Residual – the residual is $w \cdot \phi(x) - y$, the difference between the true value and predicted value, which is the amount by which prediction $f_w(x) = w \cdot \phi(x)$ overshoots the target y

Squared Loss

$$y = f_w(x) = w \cdot \phi(x)$$

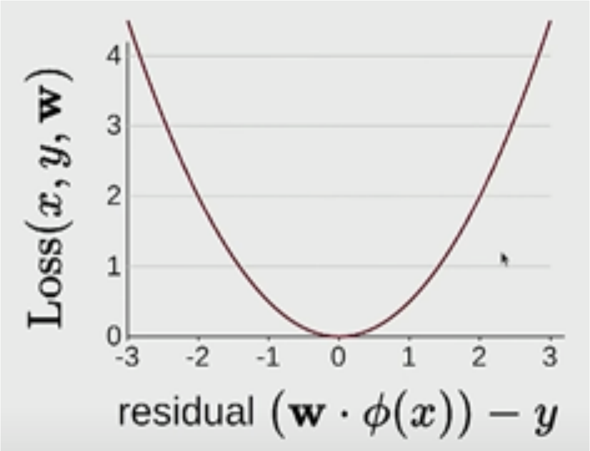
- Squared loss – the squared residual

$$Loss_{squared}(x, y, w) = (w \cdot \phi(x) - y)^2$$

- Example

$$w = [2, -1], \qquad \phi(x) = [2,0], \qquad y = -1$$

$$Loss_{squared}(x, y, w) = \left(4 - (-1)\right)^2 = 25$$



- Quadratic
- Convex curve
- $Loss_{squared}(x, y, w) = 0$ when residual is 0
- As residual grows in either direction, the loss grows quadratically

Absolute Deviation Loss

$$y = f_w(x) = w \cdot \phi(x)$$

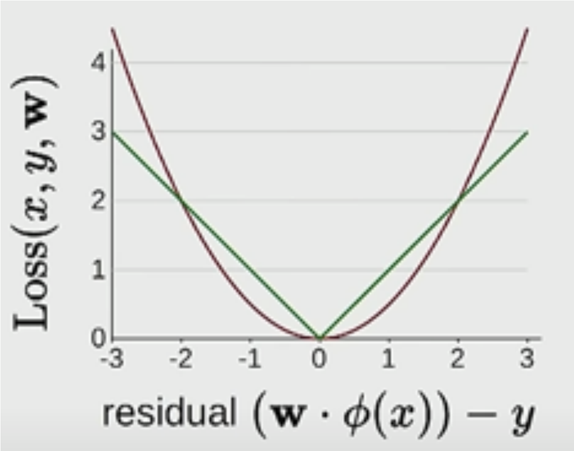
- Absolute deviation loss – the absolute value of the residual

$$Loss_{absdev}(x, y, w) = |w \cdot \phi(x) - y|$$

- Example

$$w = [-2, -1], \qquad \phi(x) = [2,0], \qquad y = -1$$

$$Loss_{absdev}(x, y, w) = |(-4) - (-1)| = 3$$



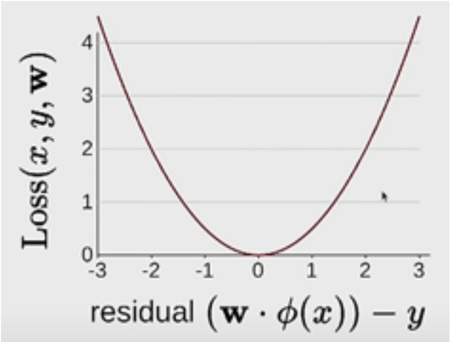
- $Loss_{squared}(x, y, w) = 0$ when residual is 0
- As residual grows in either direction, the loss grows linearly
- $Loss_{absdev}(x, y, w)$ - not smooth, harder to optimize
- $Loss_{squared}(x, y, w)$ - blows up to the outliers (large values)

Optimization using Gradient Descent

Loss Minimization

$$Loss(x, y, w) = (w \cdot \phi(x) - y)^2$$

- When we have only one input example – $Loss(x, y, w)$ is very easy to minimize



- Loss minimization for machine learning – find one weight vector to fit all the training data, and balance the errors across all of them
- Minimize the train loss over the training data – need to set w to make global trade-offs (not every example can be happy)

$$TrainLoss(w) = \frac{1}{D_{train}} \sum_{(x,y \in D_{train})} Loss(x, y, w)$$

$$\min_{w \in R^d} TrainLoss(w)$$

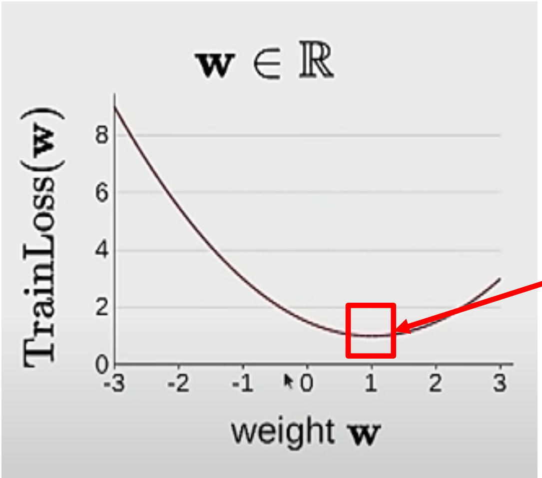
Optimization Problem

Machine learning

Optimization problem → Optimization algorithm

$$TrainLoss(w) = \frac{1}{D_{train}} \sum_{(x,y \in D_{train})} Loss(x, y, w)$$

$$\min_{w \in R^d} TrainLoss(w)$$



Find a weight vector to minimize the train loss

How to Optimize – Gradient Descent

- Gradient descent – a very generic optimization algorithm capable of finding optimal weights that minimize the train loss to a wide range of machine learning problems. The general idea of gradient descent is to tweak weights iteratively in order to minimize the train loss

- Lost in the mountains in a dense fog
- A good strategy to get down to the bottom of the valley quickly is to go downhill in the direction of the steepest slope step by step
- Once the slope is zero, you have reached to the bottom

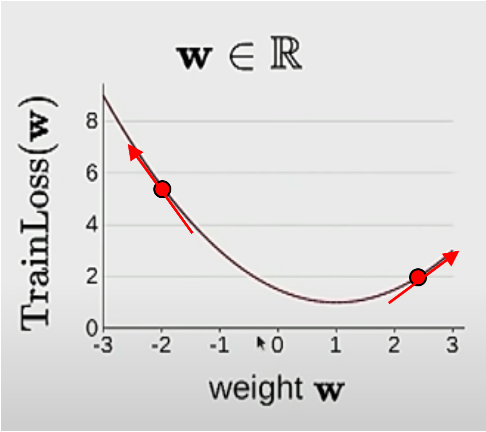
Gradient Descent

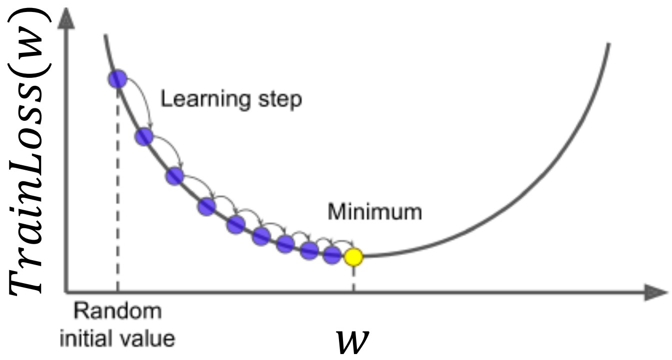
- Gradient – the gradient of the train loss with respect to a weight vector w is the direction that increase the loss the most, which is denoted as

$$\nabla_w TrainLoss(w) = \frac{\partial TrainLoss(w)}{\partial w}$$

- Gradient descent – starts by initializing the weight vector w with random values, computer the gradient $\nabla_w TrainLoss(w)$, and change the weight vector in the opposite direction of gradient (gradient descending direction) one step at a time:

$$For\ t = 1, \dots, T:$$
$$w \leftarrow w - \eta \nabla_w TrainLoss(w)$$

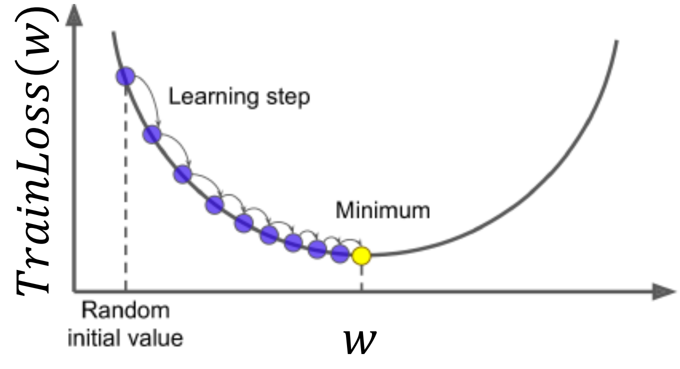




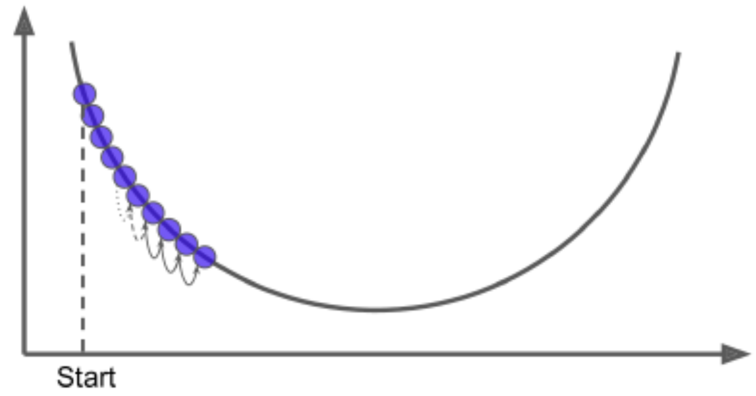
Gradient Descent – Learning Rate

- Learning rate (η) – an important parameter in gradient descent that decides the size of steps (how large we want to update weight vector, or how fast we want to move w to make progress)

$$w \leftarrow w - \boldsymbol{\eta} \nabla_w TrainLoss(w)$$

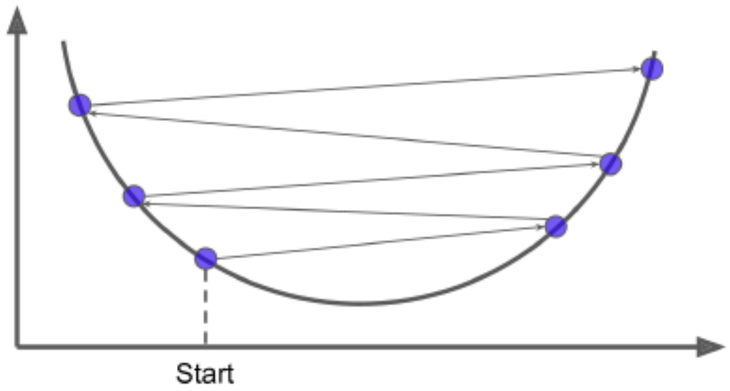


- If the learning rate is too small, then the gradient descent algorithm will have to go through many iterations to converge to the minimum

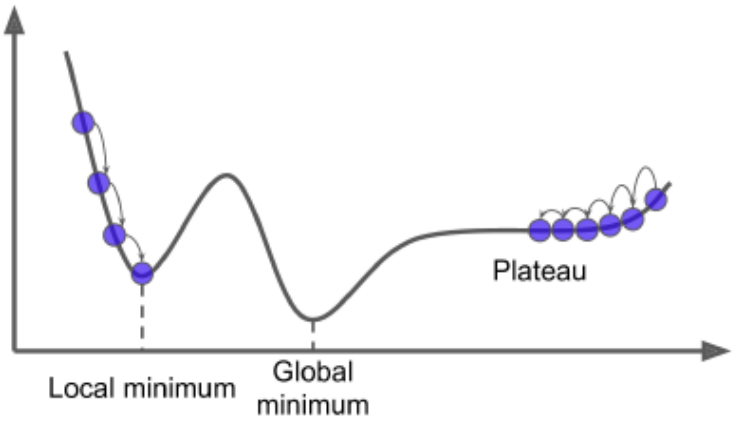


Gradient Descent – Learning Rate

- If the learning rate is too large, then we might jump across the minimum and end up on the other side, probably even higher up the train loss. This might make the gradient descent algorithm diverge, failing to find a good weight vector to minimize the loss



- Not all the train loss functions look like nice regular bowls: the loss function may have local minimum, or plateau. If the weight vector is initialized on the left, the learning rate choice may make the algorithm converge to the local minimum; if the weight vector starts on the right, it will take a very long time to across the plateau



Gradient Descent – Learning Rate

$$w \leftarrow w - \boldsymbol{\eta} \nabla_w TrainLoss(w)$$

- Question – what should the learning rate (step size) η be?

- Strategy 1 – take large learning rate (step size) first (e.g., $\eta = 0.1$) , and decrease the step size every n steps (e.g., every 20 steps, $\eta = \eta/10$)
- Strategy 2 – decreasing learning rate (step size) every step, such that $\eta = 1/\sqrt{\#updates\ made\ so\ far}$

Loss Minimization for Linear Regression

$$Loss(x, y, w) = (w \cdot \phi(x) - y)^2$$

$$TrainLoss(w) = \frac{1}{D_{train}} \sum_{(x,y \in D_{train})} Loss(x, y, w)$$

- We call the linear regression using squared loss as Least Squares Regression, which minimizes the average on the squares of residuals over the individual training examples

$$TrainLoss(w) = \frac{1}{D_{train}} \sum_{(x,y \in D_{train})} (w \cdot \phi(x) - y)^2$$

- Gradient:

$$\nabla_w TrainLoss(w) = \frac{\partial TrainLoss(w)}{\partial w}$$

$$= \frac{1}{D_{train}} \sum_{(x,y \in D_{train})} 2(w \cdot \phi(x) - y)\phi(x)$$

- Gradient descent: $w \leftarrow w - \eta \nabla_w TrainLoss(w)$

Optimization on Linear Classification

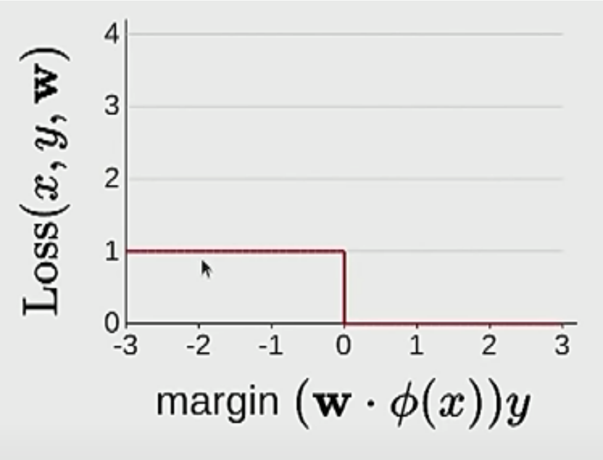
Gradient Descent Does not Work on Zero-one Loss

- Zero-one loss $Loss_{0-1}(x, y, w) = 1[(w \cdot \phi(x))y \leq 0]$
- Train loss using zero-one loss

$$TrainLoss(w) = \frac{1}{D_{train}} \sum_{(x,y \in D_{train})} 1[(w \cdot \phi(x))y \leq 0]$$

- Gradient of train loss

$$\nabla_w TrainLoss(w) = \frac{\partial TrainLoss(w)}{\partial w}$$



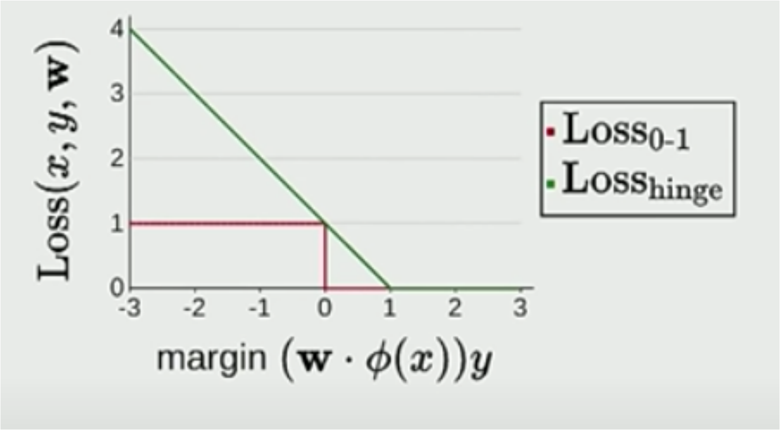
- $Loss_{0-1}(x, y, w)$ is not differentiable, and thus we cannot calculate the gradients for all weight vectors
- For the differentiable part, the gradient of loss function is zero. Weight vector won't be updated

$$w \leftarrow w - \eta \nabla_w TrainLoss(w)$$

Hinge Loss

- Hinge loss

$$Loss_{hinge}(x,y,w) = \max\{1 - \big(w \cdot \ \phi(x)\big)y, 0\}$$

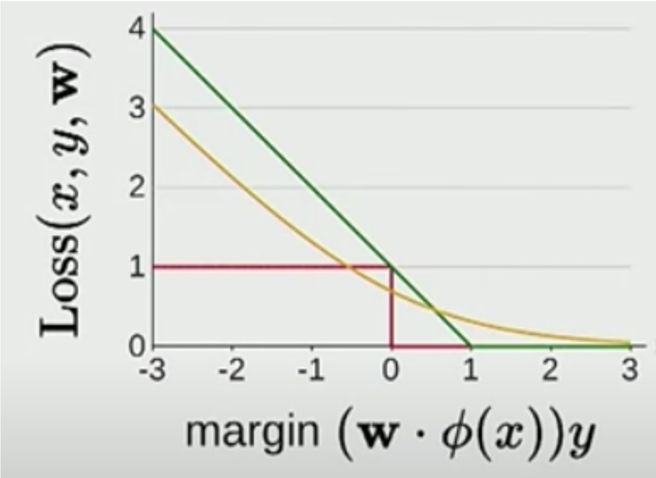


- Hinge loss is differentiable, and has a property of convexity (when you run the gradient descent, the loss will converge to the global minimum)
- Hinge loss is used as the loss function for classification model: Support Vector Machine

Logistic Loss

- Logistic loss

$$Loss_{logistic}(x, y, w) = \log(1 + e^{-(w \cdot \phi(x))y})$$



- Logistic loss is differentiable, and convex as well

- Logistic loss is used as the loss function for classification model: Logistic Regression

Summary So Far

Linear Classification and Regression

Prediction score $w \cdot \phi(x)$

	Classification	Regression
Prediction $y = f_w(x)$	$sign(score)$	$score$
Related to true label y	margin: $score\ y$	residual: $score - y$
Loss function $Loss$	zero-one hinge logistic	squared absolute deviation
Algorithm	gradient descent	gradient descent

Next Class

(2022-01-24)

Stochastic Gradient Descent and Feature Representation

WRIGHT STATE UNIVERSITY