GR5260
Programming for Quantitative & Computational Finance

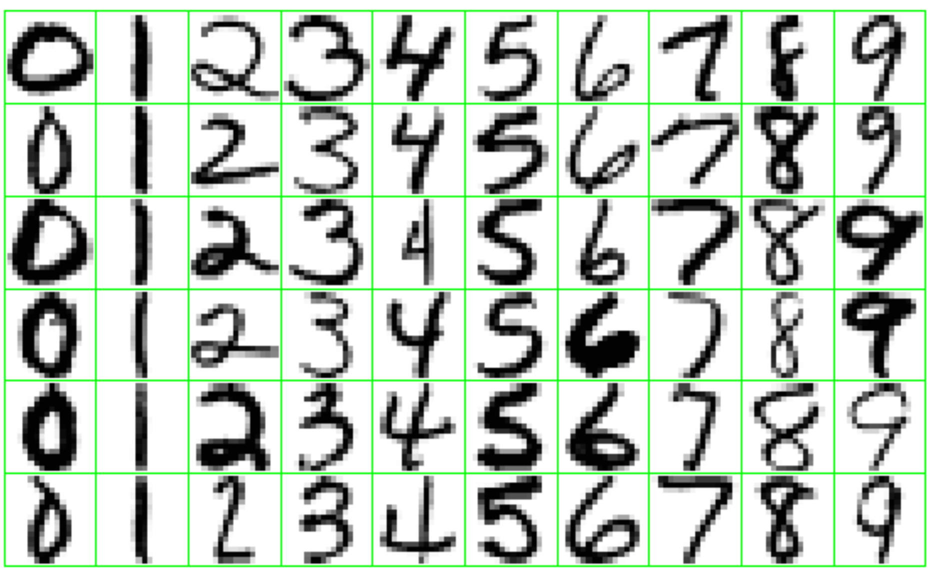
Instructor: Ka Yi Ng

April 7, 2023

Machine Learning Basics

Machine Learning

- Enables machine to learn from its experience in certain tasks
- eg. Pattern recognition
  - Provide digit images for the machine to ‘learn’
  - Ask the machine to predict the digit in a new image



 ?

Supervised Learning

- Training dataset:
  - collection of labeled examples
  - input-output pairs:

$$\left(\boldsymbol{x}^{(\mathbf{1})}, \boldsymbol{y}^{(\mathbf{1})}\right), \dots, \left(\boldsymbol{x}^{(\boldsymbol{m})}, \boldsymbol{y}^{(\boldsymbol{m})}\right)$$

- Task: learn a mapping from input to output
- Input: consists of n ‘features’

$$\boldsymbol{x}^{(\boldsymbol{i})} = \left(x_1^{(i)}, \dots, x_n^{(i)}\right)$$

- Label: $\boldsymbol{y}^{(\boldsymbol{i})}$
  - can be categorical or real-valued
  - $\boldsymbol{y}^{(\boldsymbol{i})} \in \{1,2, \dots, K\}$ categorical -> classification
  - $\boldsymbol{y}^{(\boldsymbol{i})}$ real-valued -> regression

Unsupervised Learning

- Training dataset:
  - collection of unlabeled examples

$$\boldsymbol{x}^{(\boldsymbol{1})}, \dots, \boldsymbol{x}^{(\boldsymbol{m})}$$

- Input: consists of n ‘features’

$$\boldsymbol{x}^{(\boldsymbol{i})} = \left(x_1^{(i)}, \dots, x_n^{(i)}\right)$$

- Task: discover ‘interesting patterns’,

  ‘knowledge discovery’
  - Eg. data clustering,
  - probability density estimation

Ex 1: Digit image recognition

- Classification task
- Input: image represented by n pixels, each pixel value as a feature, $\boldsymbol{x}^{(i)} = \left(x_1^{(i)}, \ldots, x_n^{(i)}\right)$
- Label: actual digit the image represents
- Task: assign a digit to an image
- Performance measure:
  - evaluate how well a ML algorithm learns the given task
  - Test dataset: separate from training dataset
  - Measure = % of test examples correctly classified

Ex 2: Linear regression

- Supervised Learning example
- Task: model a linear relationship between input vector $\boldsymbol{x} = (x_1, \dots, x_n)$ of features and output variable y

$$y = w_0 + w_1 x_1 + \cdots + w_n x_n + \varepsilon$$

where ε is noise term, unobserved r.v.

- Training dataset: $\left(\boldsymbol{x}^{(\mathbf{1})}, \boldsymbol{y}^{(\mathbf{1})}\right), \dots, \left(\boldsymbol{x}^{(\boldsymbol{m})}, \boldsymbol{y}^{(\boldsymbol{m})}\right)$
- Goal: find $\widehat{\boldsymbol{w}} = (\widehat{w}_0, \widehat{w}_1, \dots, \widehat{w}_n)$ that minimizes the mean square error (MSE)

$$MSE = \frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - w_0 - w_1 x_1^{(i)} - \cdots - w_n x_n^{(i)}\right)^2$$

Ex 2: Linear regression (cont)

- Prediction:
  - Given an input **x** this simple machine predicts the output as $\hat{y} = \hat{w}_0 + \hat{w}_1 x_1 + \cdots + \hat{w}_n x_n$
- Performance:
  - Test dataset $\left(\boldsymbol{x}^{(\mathbf{test\_1})}, \boldsymbol{y}^{(\mathbf{test\_1})}\right), \ldots, \left(\boldsymbol{x}^{(\boldsymbol{test\_r})}, \boldsymbol{y}^{(\boldsymbol{test\_r})}\right)$

$$MSE = \frac{1}{r} \sum_{i=1}^{\boldsymbol{r}} \left(y^{(test\_i)} - \hat{y}^{(test\_i)}\right)^2$$

- Closed form solution for $\hat{\boldsymbol{w}} = (X^T X)^{-1} X^T Y$

$$X = \begin{bmatrix} 1 & x_1^{(1)} \ldots x_n^{(1)} \\ \vdots & \quad\quad\quad \vdots \\ 1 & x_1^{(m)} \ldots x_n^{(m)} \end{bmatrix} \qquad Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

Function estimation

- Estimate the input-output mapping $y = f(\boldsymbol{x}) + \varepsilon$ using a parametrized function $\hat{f}(\boldsymbol{x}; \boldsymbol{w})$
- Prediction: $\hat{y} = \hat{f}(\boldsymbol{x}; \hat{\boldsymbol{w}})$ new input
- Cost (or loss) function $L(\hat{y}, y)$
  - measure the error of the function estimation
- Training dataset: $\left(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}\right), \dots, \left(\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}\right)$
- Goal: find an estimate $\widehat{\boldsymbol{w}}$ for $\boldsymbol{w}$ that minimizes the average cost function $J(\boldsymbol{w})$ over the training set.

$$J(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m} L\left(\boxed{\hat{f}\left(x^{(i)}; \boldsymbol{w}\right)}, y^{(i)}\right)$$

actual label

predicted label

Function estimation (cont)

- Performance:
  - Prediction error on the test dataset $\left(\boldsymbol{x}^{(\mathbf{test\_1})}, \boldsymbol{y}^{(\mathbf{test\_1})}\right), \dots, \left(\boldsymbol{x}^{(\boldsymbol{test\_r})}, \boldsymbol{y}^{(\boldsymbol{test\_r})}\right)$

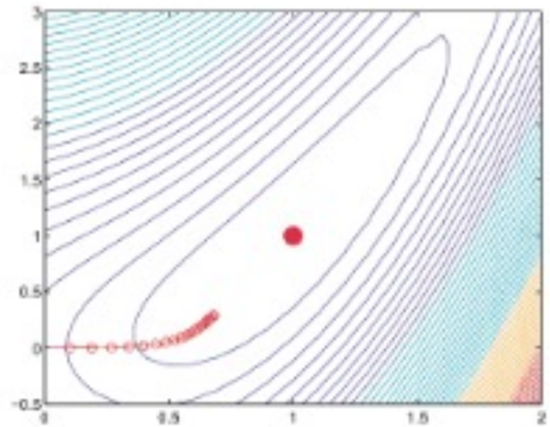
$$\hat{y}^{(test\_i)} = \hat{f}\left(x^{(test\_i)}; \widehat{\boldsymbol{w}}\right)$$

$$PredictionError = \frac{1}{r}\sum_{i=1}^{r} L\left(\hat{y}^{(test\_i)}, y^{(test\_i)}\right)$$

- Key point is to select:
  - Model function $\hat{f}(\boldsymbol{x}; \boldsymbol{w})$ (eg. kernel-based, tree-based)
  - Cost function $L(\hat{y}, y)$ (eg. mean squared error)
  - Optimization algorithm (eg. gradient-based)

Gradient (or Steepest) Descent

- Suppose we want to find $\hat{\boldsymbol{w}}$ that minimizes a real-valued function $f(\boldsymbol{w})$, $\boldsymbol{w} \in \mathbb{R}^n$



- Algorithm:
  i. start with a guess $\boldsymbol{w_0}$, set $\boldsymbol{k} = \boldsymbol{0}$
  ii. find a direction $\boldsymbol{u_k}$ where $f(\boldsymbol{w_k})$ decreases the fastest
  iii. set $\boldsymbol{w_{k+1}} = \boldsymbol{w_k} + \epsilon \boldsymbol{u_k}$ where ϵ is a small value (learning rate)
  iv. repeat the step ii and iii until some stopping criterion is met (eg. $||\nabla_\theta f(\boldsymbol{w_k})|| <$ some δ)
  v. $\hat{\boldsymbol{w}} = \boldsymbol{w_k}$ where the stopping criterion for $\boldsymbol{w_k}$ is satisfied

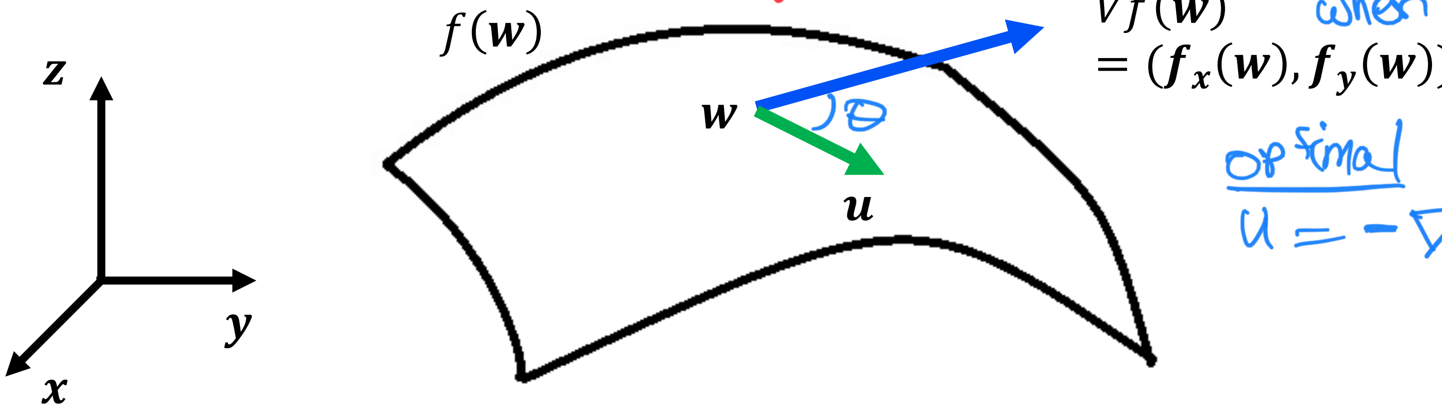
Gradient Descent: Finding $\boldsymbol{u_k}$

- Find a direction $\boldsymbol{u_k}$ where $f(\boldsymbol{w_k})$ <u>decreases</u> the fastest
- Let $\boldsymbol{u}$ be a unit vector. Then the directional derivative of $f(w)$ w.r.t. $\boldsymbol{u}$ is:

$$\frac{\partial}{\partial \cancel{\epsilon}\, t} f(\boldsymbol{w} + \cancel{\epsilon}\, t\,\boldsymbol{u})\Big|_{t=0} = \boxed{\nabla_w f(\boldsymbol{w}) \cdot \boldsymbol{u}}$$

$= \|\nabla_w f\| \cos\theta$

most negative when $\theta = 180^\circ$



<u>optimal</u>

$u = -\nabla f(w)$

- Therefore, $-\nabla f(w)$ is the steepest gradient along which $f(\boldsymbol{w})$ decreases the fastest.

$$\boldsymbol{w_{k+1}} = \boldsymbol{w_k} - \epsilon\, \nabla f(\boldsymbol{w_k})$$

$f(w+tu) = (f_1(w+tu), \ldots,$

$$\frac{\partial f_1}{\partial t} = \sum_{j=1}^{n} \frac{\partial f_1}{\partial w_j} \cdot \underbrace{\frac{\partial w_j + t u_j}{\partial t}}_{u_j}$$

Stochastic Gradient Descent

- Recall: find an estimate $\hat{\boldsymbol{w}}$ for $\boldsymbol{w}$ that minimizes

$$J(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{f}(x^{(i)}; \boldsymbol{w}), y^{(i)})$$

- In each iteration use a smaller set of training data
- Algorithm:

i. Start with a guess $\boldsymbol{w_0}$. Fix a learning rate ϵ. Set $\boldsymbol{k} = \boldsymbol{0}$

ii. Select a minibatch of l examples from the full training dataset

iii. Estimate gradient using $A(\boldsymbol{w}) = \frac{1}{l}\sum_{i=1}^{l} L(\hat{f}(x^{(s\_i)}; \boldsymbol{w}), y^{(s\_i)})$

iv. Set $\boldsymbol{w_{k+1}} = \boldsymbol{w_k} - \epsilon \nabla A(\boldsymbol{w_k})$

v. Repeat the step ii - iv until some stopping criterion is met (eg. $||\nabla A(\boldsymbol{w_k})|| <$ some δ)

vi. $\hat{\boldsymbol{w}} = \boldsymbol{w_k}$ where the stopping criterion for $\boldsymbol{w_k}$ is satisfied

Stochastic Gradient Descent

- Minibatch selection:
  - Initially shuffle the points in the training dataset
  - Use this ordering to pick the minibatch in each iteration
  - Example: let's say $\boldsymbol{d_i} = \left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right)$
  - shuffled: $\boldsymbol{d_{15}}, \boldsymbol{d_4}, \boldsymbol{d_{300}}, \boldsymbol{d_{101}}, \boldsymbol{d_{72}}, \boldsymbol{d_{26}}, \ldots$
  - Iteration 1: first l examples, Iteration 2: next l examples, etc.

- Gradient Descent: $\boldsymbol{w_{k+1}} = \boldsymbol{w_k} - \epsilon \nabla J(\boldsymbol{w})$
- Stochastic Gradient Descent: $\boldsymbol{w_{k+1}} = \boldsymbol{w_k} - \epsilon \nabla A(\boldsymbol{w})$
- The gradient estimate may not reach near zero

Early Stopping

- Stop the iteration when the predictive power is sufficiently good
- Validation set: set aside a subset of training dataset (eg. 20%) $\left(\boldsymbol{x}^{(\mathbf{val\_1})}, \boldsymbol{y}^{(\mathbf{val\_1})}\right), \ldots, \left(\boldsymbol{x}^{(\boldsymbol{val\_p})}, \boldsymbol{y}^{(\boldsymbol{val\_p})}\right)$
- In each iteration of estimating $\boldsymbol{w_k}$, compute the validation error

$$VE(\boldsymbol{w_k}) = \frac{1}{p}\sum_{i=1}^{p} L(\hat{f}\left(x^{(val\_i)}; \boldsymbol{w_k}\right), y^{(val\_i)})$$

- If the validation error is sufficiently small, stop the iteration loop
- Or if the validation error doesn't improve in several consecutive iterations, stop the iteration loop (Early Stopping)

Stochastic Gradient Descent

- Modified algorithm:
  i. Set aside a validation set $\left(\boldsymbol{x}^{(\mathbf{val\_1})}, \boldsymbol{y}^{(\mathbf{val\_1})}\right), \dots, \left(\boldsymbol{x}^{(\boldsymbol{val\_p})}, \boldsymbol{y}^{(\boldsymbol{val\_p})}\right)$
  ii. Randomly shuffle points in the training dataset (remaining ones)
  iii. Start with a guess $\boldsymbol{w_0}$. Fix a learning rate ϵ. Set $\boldsymbol{k} = \mathbf{0}$
  iv. Compute validation error $VE(\boldsymbol{w_k})$
  v. For each iteration k, do the following:
     - If validation error $\leq \delta$ or validation error hasn't improved (stopping criteria), then break the loop
     - Take the kth minibatch $\boldsymbol{D_k}$ of l examples from the shuffled training dataset
     - Estimate gradient using the minibatch $\boldsymbol{D_k}$
     - $A(\boldsymbol{w}) = \frac{1}{l}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\boldsymbol{D_k}} L(\hat{f}(\boldsymbol{x};\boldsymbol{w}), \boldsymbol{y})$
     - Set $\boldsymbol{w_{k+1}} = \boldsymbol{w_k} - \epsilon\nabla A(\boldsymbol{w_k})$
  vi. $\widehat{\boldsymbol{w}} = \boldsymbol{w_k}$ the last calculated value after iteration stops

Epoch vs Iterations

- Given $N = 10000$ training data points.
- In each iteration, if we use a minibatch of 50 examples, then all training data points will be used in 200 iterations.
- One epoch: iterate through all training data points in the model parameter optimization procedure
- Multiple epochs are used in the optimization
- Eg. 10 epochs → 2000 iterations

Key steps

- Data preparation:
  - Data cleaning
  - Feature selection
    - Correlation between each feature and the target label
    - Regression
  - Feature engineering (eg. categories→values)
  - Designate a test dataset, separate from training
  - Features scaling:
    - Make features in the same order of magnitude
- Fit a model using training dataset
- Evaluate the trained model using test dataset
- Select best model from many trained models

Feature engineering

- Categorical features
  - Categories → ordinals:
    eg. ‘Low’ → 1, ‘Medium’ → 2, ‘High’ → 3, etc.
  - Categories → one-hot vectors:
    eg. ‘Aaa’ → (1,0,0,0,0,0,0,0,0), ‘Aa’ → (0,1,0,0,0,0,0,0,0)
- Binary features: {0,1} or {-1,1}
- Numerical features:
  - In some cases, may want to convert them to categorical values
  - Eg. family income → categories
- Create new features from the features available
  - Eg. historical prices → daily returns
  - Eg. Principal Component Analysis: dimension reduction

Feature scaling

- Recall: model training as optimizing a cost function
- To address the different order of magnitudes in different features. eg. trade volume, price volatility, rate of return
- For each feature: find some function h_k: $x_k \rightarrow z_k = \boxed{h_k(x_k)}$
- Transformed training dataset: $\left(\boldsymbol{z}^{(\mathbf{1})}, \boldsymbol{y}^{(\mathbf{1})}\right), \ldots, \left(\boldsymbol{z}^{(\boldsymbol{m})}, \boldsymbol{y}^{(\boldsymbol{m})}\right)$ where $\boldsymbol{z}^{(\boldsymbol{i})} = \left(z_1^{(i)}, \ldots, z_n^{(i)}\right)$
- Transformed test data: $\left(\boldsymbol{x}^{(\boldsymbol{test\_i})}, \boldsymbol{y}^{(\boldsymbol{test\_i})}\right) \rightarrow \left(\boldsymbol{z}^{(\boldsymbol{test\_i})}, \boldsymbol{y}^{(\boldsymbol{test\_i})}\right)$
- Use the transformed datasets for training and evaluation
- Training set: $\left\{\left(\boldsymbol{x}^{(\boldsymbol{i})}, \boldsymbol{y}^{(\mathbf{i})}\right)\right\}_{i=1,..,m} \rightarrow \hat{f}(\boldsymbol{x}; \boldsymbol{w})$

$\hat{f}$

- Transformed: $\left\{\left(\boldsymbol{z}^{(\boldsymbol{i})}, \boldsymbol{y}^{(\mathbf{i})}\right)\right\}_{i=1,..,m} \rightarrow \boxed{\hat{g}_w\,(\boldsymbol{z})} = \boxed{\hat{g}_w\,(\boldsymbol{h}(\boldsymbol{x}))}$

Feature scaling

- Two common transformations:
  - Standardization (z-score):
  - $h_k(x) = \boxed{\frac{x-\mu_k}{\sigma_k}}$ where μ_k and σ_k are resp. the mean and standard dev of $x_k^{(1)}, x_k^{(2)}, \ldots, x_k^{(m)}$ from training set
  - MinMax scaling:

    [0, 1]
  - $h_k(x) = a + \frac{b-a}{M_k - m_k}(x - m_k) \in [a, b]$ where m_k and M_k are resp. the min and max of $x_k^{(1)}, x_k^{(2)}, \ldots, x_k^{(m)}$ from training set

    respectively

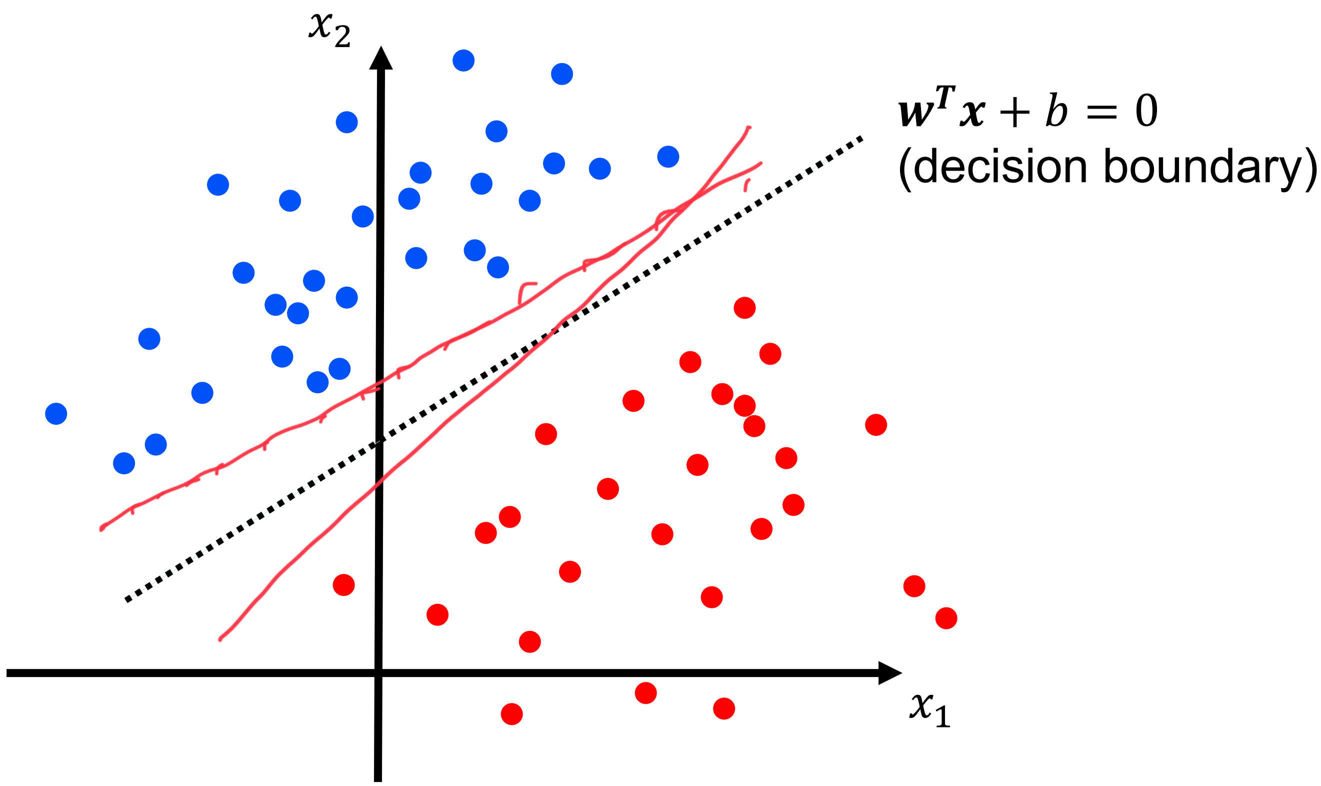
Model selection

- Machine learning algorithms:
  - Model function $\hat{f}(\boldsymbol{x}; \boldsymbol{w})$
  - Cost function $L(\hat{y}, y)$
- Classical ML models:
  - Logistic regression models
  - Support Vector Machine (SVM)
  - Decision trees, random forest, ensemble methods
  - K-Nearest Neighbor (KNN)
  - Neural networks

Support Vector Machines

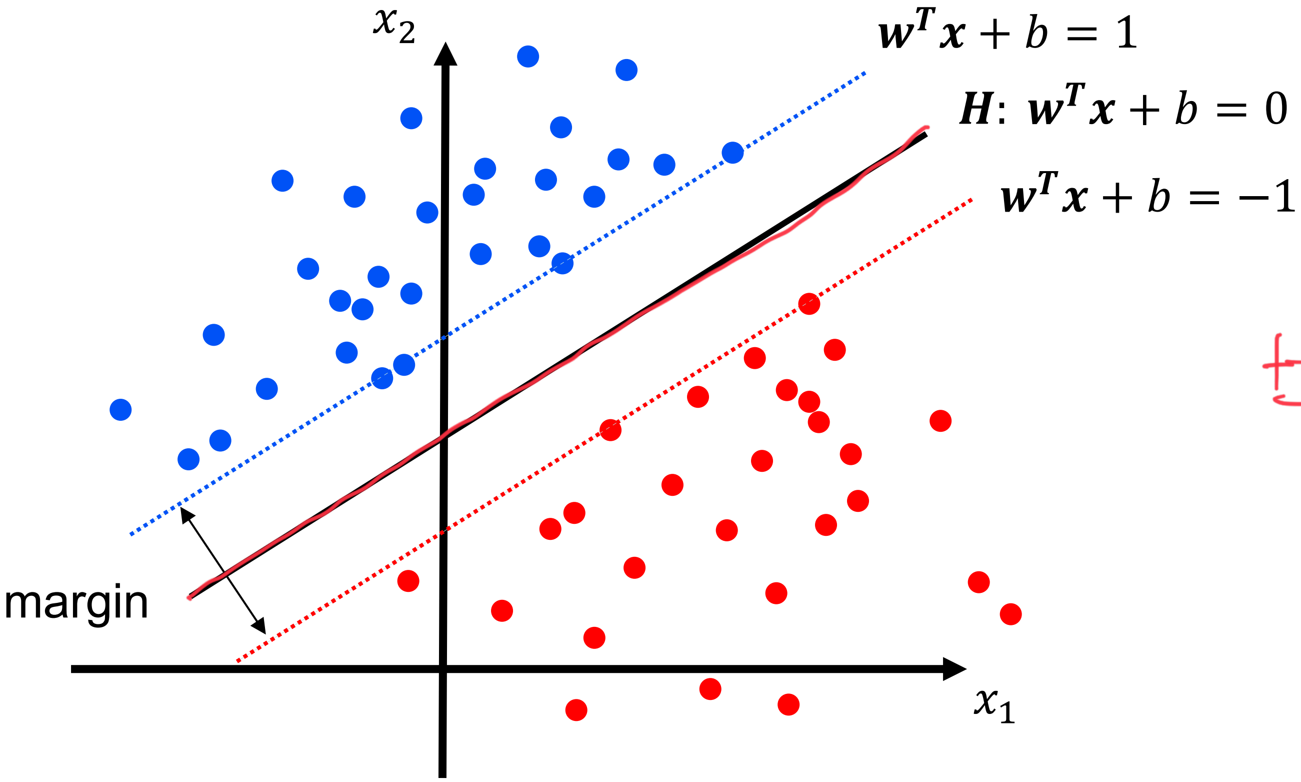
- Linear vs non-linear method
- Hard vs soft margin
- Binary classification
- Regression
- Multi-class classification
- Detect outliers and anomalies

Binary Classification: linear



- Linear Classifier: use a hyperplane to separate the two classes of points

Binary Classification: linear



Hard margin

- Maximum margin classifier
- Goal: find the hyperplane H which has the greatest distance from the nearest data points

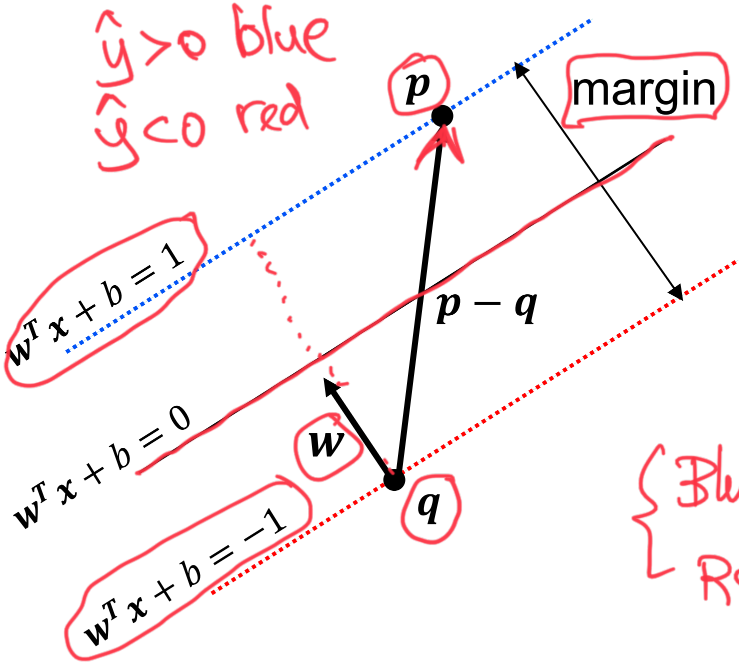
Binary Classification: Linear

red blue

- Training set: $(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \ldots, (\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}), \boldsymbol{y}^{(i)} \in \{-1, 1\}$
- Decision function: $\boldsymbol{w}^T\boldsymbol{x} + b = w_1x_1 + \cdots + w_nx_n + b$
- Prediction: $\hat{y} = sign(\boldsymbol{w}^T\boldsymbol{x} + b)$

new point

$\hat{y} > 0$ blue
$\hat{y} < 0$ red



$$\textbf{margin} = \frac{(\boldsymbol{p}-\boldsymbol{q})\cdot\boldsymbol{w}}{\|\boldsymbol{w}\|} = \frac{\boldsymbol{p}\cdot\boldsymbol{w} - \boldsymbol{q}\cdot\boldsymbol{w}}{\|\boldsymbol{w}\|} = \frac{\mathbf{1}-b-(-\mathbf{1}-b)}{\|\boldsymbol{w}\|} = \boxed{\frac{\mathbf{2}}{\|\boldsymbol{w}\|}}$$

min $\|w\|$

minimize $\frac{1}{2}\|\boldsymbol{w}\|^2$ subject to:

Blue: $\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b \geq 1$ for i with $\boldsymbol{y}^{(i)} = 1$

Red: $\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b \leq -1$ for i with $\boldsymbol{y}^{(i)} = -1$

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \text{ subject to } \boldsymbol{y}^{(i)}(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b) \geq 1 \text{ for } 1 \leq i \leq m$$

Binary Classification: Linear

$$\boxed{\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \text{ subject to } \boldsymbol{y}^{(i)}(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})} + b) \geq 1 \text{ for } 1 \leq i \leq m}$$

<u>Karush-Kuhn-Tucker Theorem</u> (Extended Lagrange Multiplier):

Given an optimization problem (A),

$$\min_{\boldsymbol{x}} \underline{f(\boldsymbol{x})} \text{ subject to } g_i(\boldsymbol{x}) \leq 0 \text{ for } 1 \leq i \leq m$$

where $f(\boldsymbol{x})$ and $g_i(\boldsymbol{x})$ are convex functions

Then $\hat{\boldsymbol{x}}$ is a solution to (A) $\Leftrightarrow$ $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\alpha}})$ is a solution to

$$\max_{\boldsymbol{\alpha} \geq \boldsymbol{0}} \min_{\boldsymbol{x}} \left[\underline{f(\boldsymbol{x})} + \underline{\textstyle\sum_{i=1}^{m} \alpha_i g_i(\boldsymbol{x})}\right]$$

< 0

In particular, $\hat{\alpha}_i g_i(\hat{\boldsymbol{x}}) = 0$ for all i

$$\boxed{\text{Set } f(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2,\ g_i(\boldsymbol{w}) = 1 - \boldsymbol{y}^{(i)}\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})} + b\right)}$$

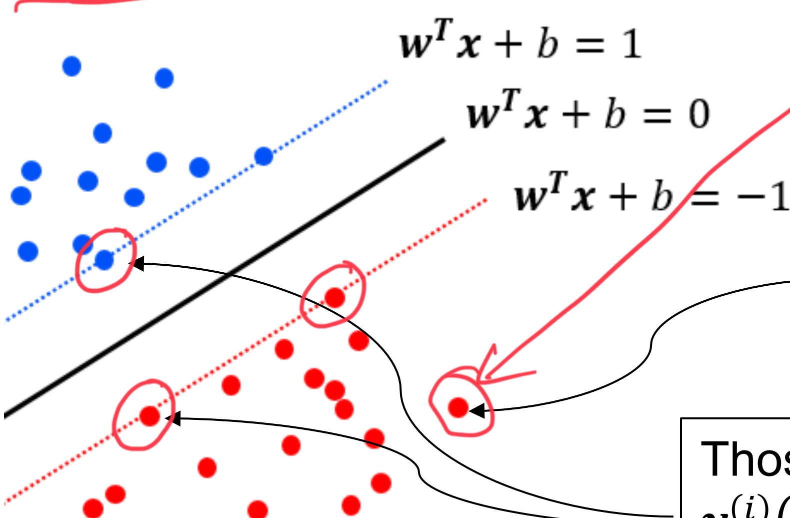
Binary Classification: Linear

$$\min_{\boldsymbol{w},b} \frac{\mathbf{1}}{\mathbf{2}} \|\boldsymbol{w}\|^2 \text{ subject to } \boldsymbol{y}^{(i)}(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b) \geq 1 \text{ for } 1 \leq i \leq m$$

$$\max_{\boldsymbol{\alpha} \geq \mathbf{0}} \min_{\boldsymbol{w},b} \frac{\mathbf{1}}{\mathbf{2}} \|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \alpha_i \{1 - \boldsymbol{y}^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right)\}$$

Thm $\hat{\alpha}_i g_i(\hat{\boldsymbol{x}}) = 0 \rightarrow \hat{\alpha}_i \{1 - \boldsymbol{y}^{(i)}\left(\hat{\boldsymbol{w}}^T\boldsymbol{x}^{(i)} + b\right) = 0$

$w^T x + b < -1$



For $\boldsymbol{x}^{(i)}$ with $\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b < -1$ or > 1, $\boldsymbol{y}^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) > 1$, $\hat{\alpha}_i$ must be 0

Those with $\hat{\alpha}_i > 0$ must have $\boldsymbol{y}^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) = 1$
They are called **support vectors**