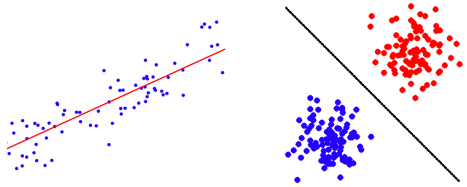
MACQUARIE University

MACHINE LEARNING

COMP8220

03 – Regression and Classification



Xuyun Zhang (Department of Computing)

Lecture Outline

MACQUARIE University

- Linear Regression Introduction
  - Linear Regression Model
  - Overfitting and Model Selection
- Classification
  - Discriminant Functions
  - Perceptron
  - Gradient Descent
  - K-Nearest Neighbors Classifier

Lecture Outline

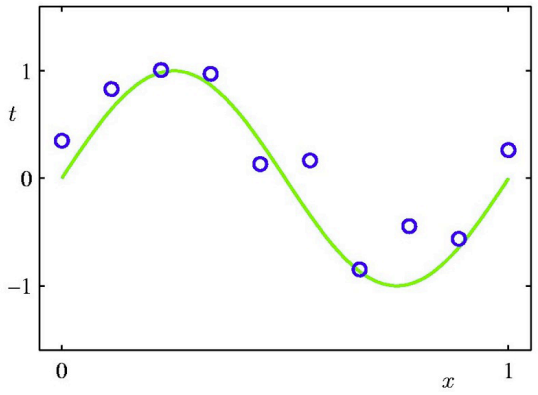
MACQUARIE University

- Linear Regression Introduction
  - Linear Regression Model
  - Overfitting and Model Selection

Regression

MACQUARIE University

- Suppose we have 10 observations
  - $\mathcal{D} = \{\langle x_1, t_1\rangle, \cdots, \langle x_{10}, t_{10}\rangle\}$
  - Ground truth: $t = \sin(2\pi x)$
  - Noise added to t_i



- **Regression** task: estimate $t = y(x)$ from D

Linear Models

MACQUARIE University

- A function is linear if

$$f(\alpha x_1 + \beta x_2) = \alpha f(x_1) + \beta f(x_2)$$

  - α, β are scalars
  - Additivity and homogeneity

- Linear regression model

$$y(\boldsymbol{x}, \boldsymbol{\omega}) = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_D x_D$$

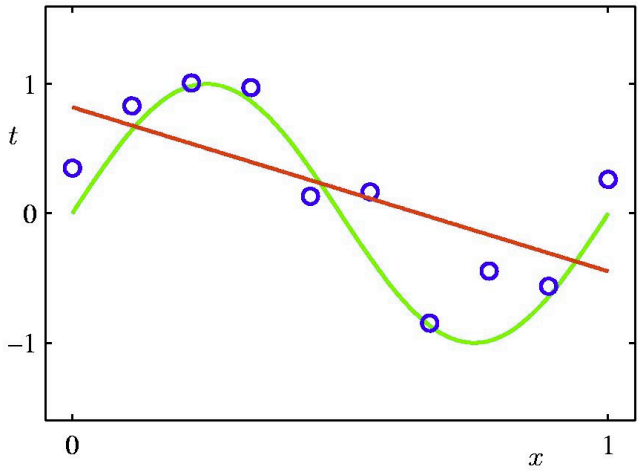
  - Note: D is the # of features, not the # of instances
  - The linearity of $y(\cdot, \omega)$ is with respect to ω
  - Note: we are learning ω

The Simplest From

MACQUARIE University

- Dimensionality $M = 1$

$$y(x, \boldsymbol{\omega}) = \omega_0 + \omega_1 x$$



  - Can only model a "line"
- How to handle non-linearity?

Feature Extraction

MACQUARIE University

- Transform x

$$\boldsymbol{x} \Rightarrow \boldsymbol{\Phi}(\boldsymbol{x}) = (\phi_0(\boldsymbol{x}), \phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \cdots, \phi_M(\boldsymbol{x}))^T$$

  - $\phi_j(x), j \geq 0$ form a family of basic functions
  - Transformation/feature function is often non-linear

- Linear basis function models

$$y(\boldsymbol{x}, \boldsymbol{\omega}) = \sum_{j=0}^{M} \omega_j \phi_j(\boldsymbol{x}) = \boldsymbol{\omega}^T \boldsymbol{\Phi}(\boldsymbol{x})$$

  - $\boldsymbol{\omega} = (\omega_0, \cdots, \omega_M)^T$, $\boldsymbol{\Phi} = (\phi_0, \cdots, \phi_M)^T$

Special Cases

MACQUARIE University

- For $0 < j \leq D$, $\phi_j(x) = x_j$; $\phi_0(x) = 1$

$$y(\boldsymbol{x}, \boldsymbol{\omega}) = \sum_{j=0}^{D} \omega_j x_j$$

  - The original form

- $\phi_j(x) = x^j$, $0 < j \leq M$

$$y(x, \boldsymbol{\omega}) = \sum_{j=0}^{M} \omega_j x^j$$

  - Power series expansions

Loss Function for Regression

MACQUARIE University

- How to learn/choose the parameter vector ω?
  - First, define a loss function $L(y(x), t)$
  - Then, find out $y(x)$ to minimize the loss (expectation)

$$y^*(\boldsymbol{x}) = \arg\min_{y(\boldsymbol{x})} \mathbb{E}[L] = \int\int L(y(\boldsymbol{x}), t)p(\boldsymbol{x}, t)d\boldsymbol{x}dt$$

  - E.g., general squared loss function:

$$L(y(\boldsymbol{x}), t) = \frac{1}{2}||y(\boldsymbol{x}) - t||^2$$

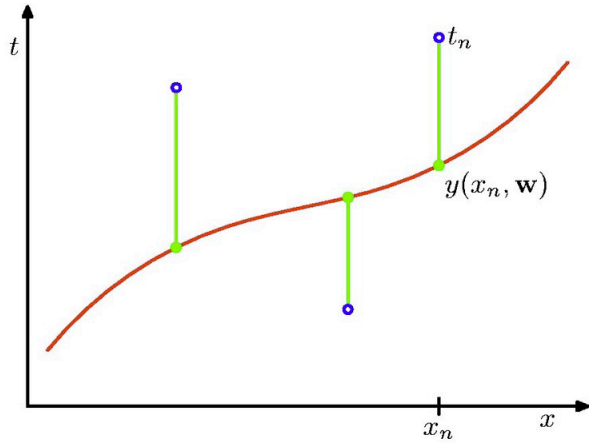
  - In practice, we often use the loss function defined on training data

Least Squares

MACQUARIE University

- Sum of squared error:

$$E(\omega) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \omega) - t_n\}^2$$



- Then, the specific learning problem

$$\omega^* = \arg\min_{\omega} E(\omega)$$

Optimization Problem

MACQUARIE University

- Learning problem → optimization problem

$$\omega^* = \arg\min_{\omega} E(\omega),$$

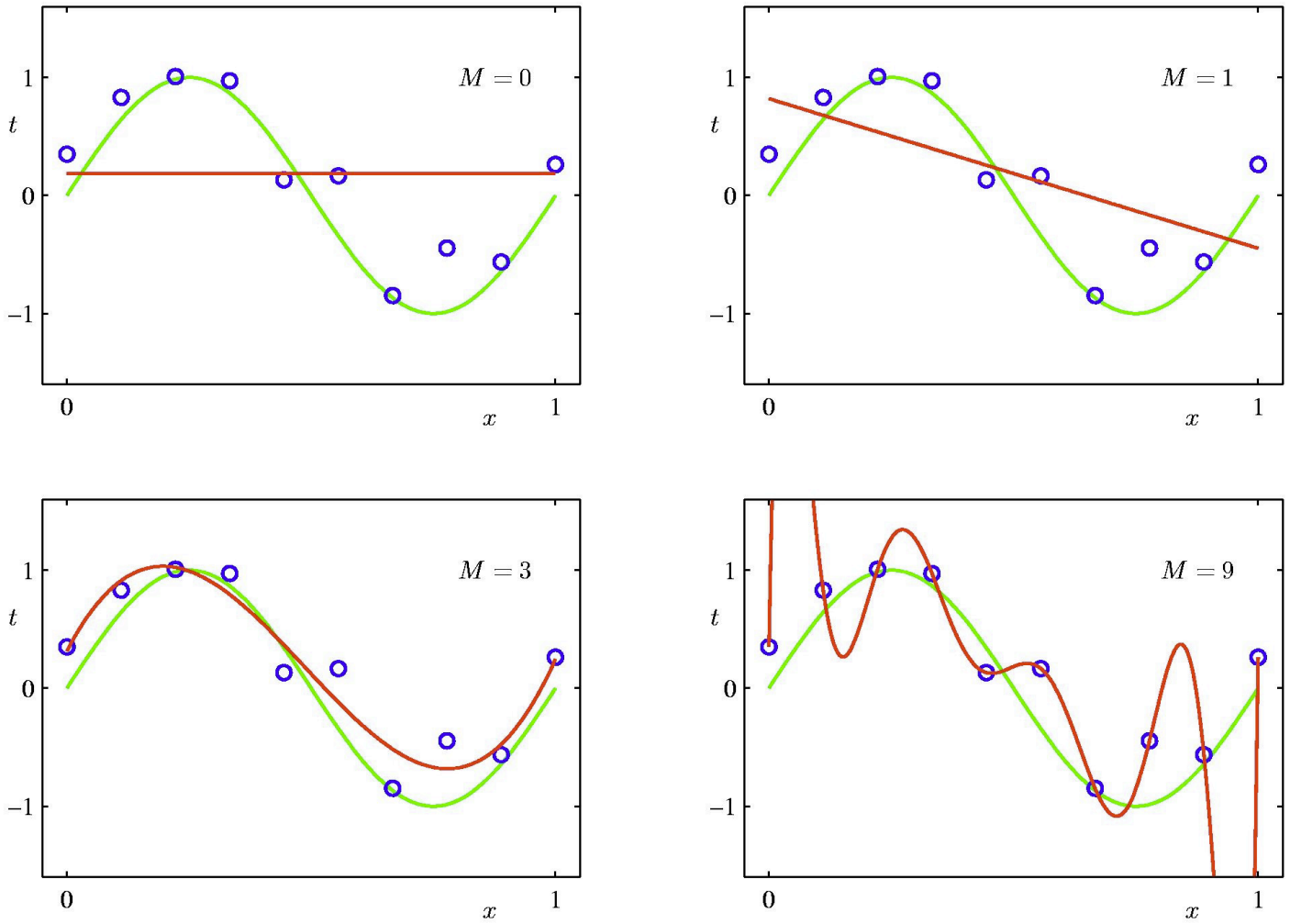
$$E(\omega) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \omega) - t_n\}^2$$

- How to solve? $\dfrac{\partial E(\omega)}{\partial \omega} = 0$
  - Close-form solution can be obtained
  - For the basis function model

$$\omega_{ML} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\boldsymbol{t}$$

Different Degrees of Polynomials

MACQUARIE University



Observations

MACQUARIE University

- $M = 0$ or $M = 1$
  - Only passing a couple of points
  - Models are too simple
- $M = 9$
  - Passing every points but oscillating wildly
  - **Overfitting** problem: Lack of generalisation capability
- $M = 3$
  - Much better solution
  - Good trade-off has been made

Training/Test Error

MACQUARIE University

- Training/empirical error of the trained model $\hat{f}$ on a training dataset with size N, e.g.,

$$E_{emp}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

  - Learning process usually tries to minimize this error

- Test error on a test data set with size N', e.g.,

$$E_{test}(\hat{f}) = \frac{1}{N'}\sum_{i=1}^{N'} L(y_i, \hat{f}(x_i))$$

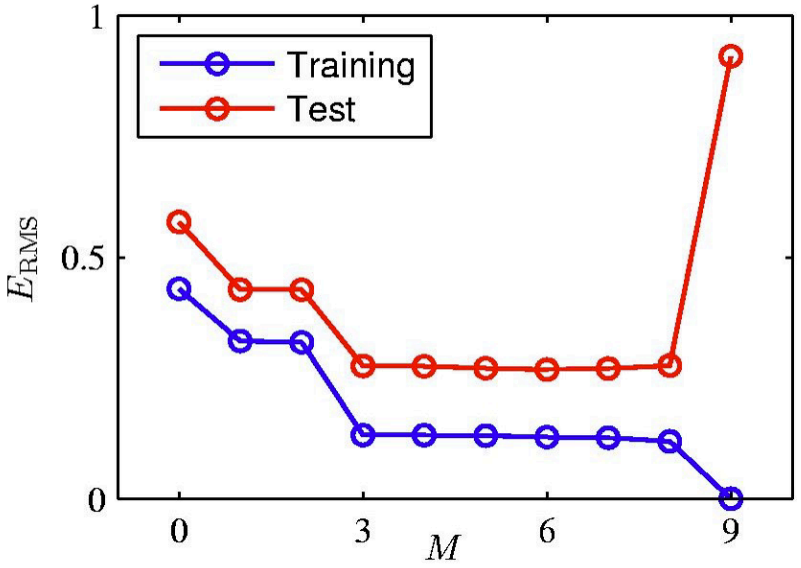
  - Test error indicates the generalisation capability of a learned model

Overfitting Issue

MACQUARIE University

- Test errors with respect to M
  - Test data containing 100 points
  - Root-Mean-Square (RMS) error

$$E_{RMS} = \sqrt{2E(\omega^*)/N}$$



Model Selection

MACQUARIE University

- Observations on M
  - Different models generated by varying M
  - M controls the learning capability of the model
  - Unlike model parameter ω, M can NOT be learned from data (**hyperparameter**)
- Model selection
  - Models generated by different learning algorithms
  - Models generated by different algorithm parameter (model hyperparameter) configurations
- Model selection is a non-trial job!

Regularization

MACQUARIE University

- Another way to the overfitting issue
  - Penalizing large coefficients in error function
  - E.g., quadratic regularizer

$$\tilde{E}(\boldsymbol{\omega}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{\omega}) - t_n\}^2 + \frac{\lambda}{2}||\boldsymbol{\omega}||^2,$$

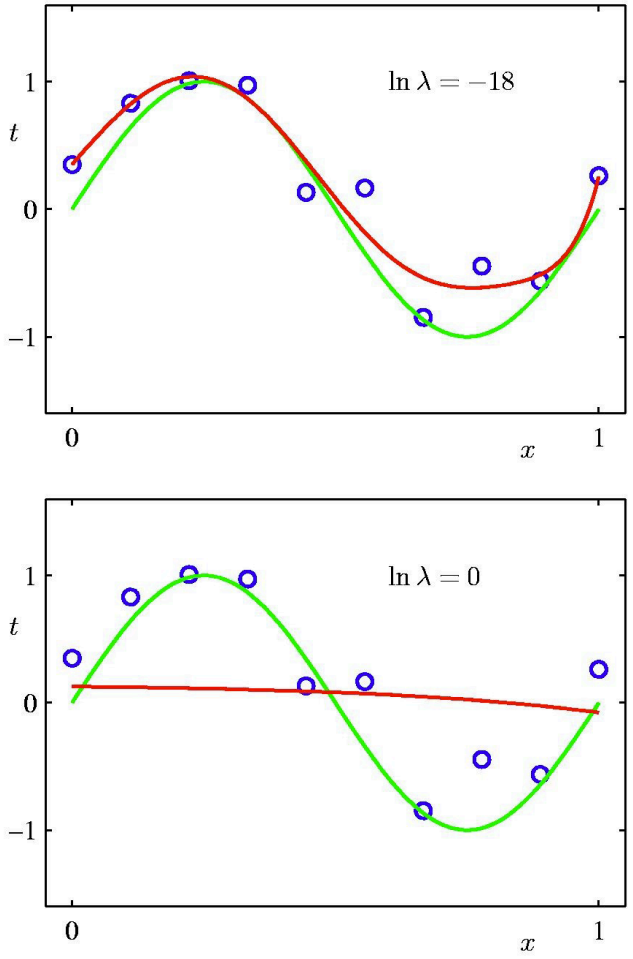
$$||\boldsymbol{\omega}||^2 = \boldsymbol{\omega}^T\boldsymbol{\omega} = \omega_0^2 + \omega_1^2 + \cdots + \omega_M^2$$

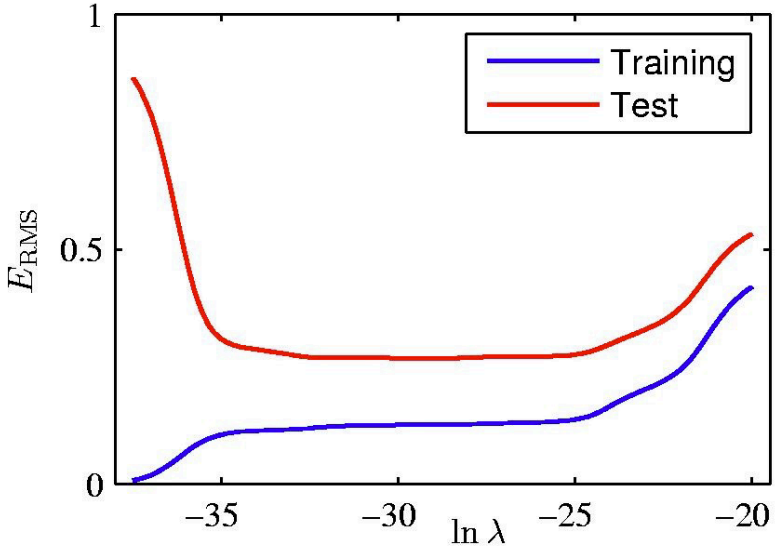
  - λ governs the trade-offs between the two terms
  - A.k.a. ridge regression
  - Close-form solution can be obtained
  - Lasso regression: the regularizer is $||\omega||_{L_1}$

Effects of λ

MACQUARIE University

- Performance of different λ when $M = 9$





Scikit-learn (sklearn)

MACQUARIE University

- Data preparation
- Training/test data split
  - x_train, y_train, x_test, y_test
- Train the model
  - Build a model: model = xxxModel.xxxModel()
  - Train a model: model.fit(x_train, y_train)
  - Validate: cross_validate(xxxmodel, x_train, y_train, k, scoring)
- Test the learning model
  - Test the model: y_pred = xxxModel.predict(x_test)
  - Can use y_pred and y_test to calculate the error, e.g., mean_squared_error(y_test, y_pred)

Lecture Outline

MACQUARIE University

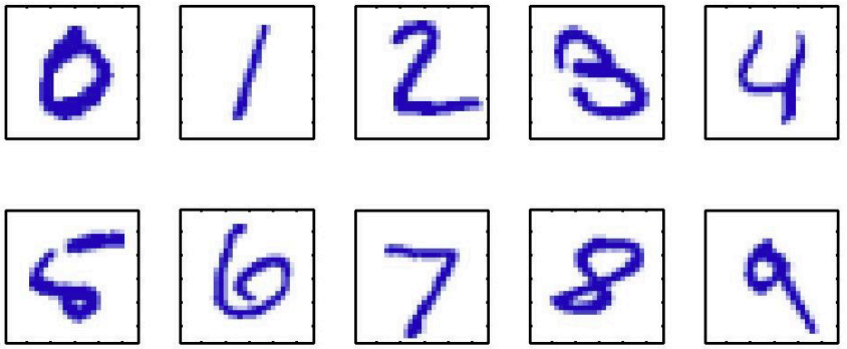
- Linear Regression Introduction
  - Linear Regression Model
  - Overfitting and Model Selection
- Classification
  - Discriminant Functions
  - Perceptron
  - Gradient Descent
  - K-Nearest Neighbors Classifier

Classification Problem

MACQUARIE University

- Given training set $\mathcal{D} = \{\langle \boldsymbol{x}_1, t_1 \rangle, \langle \boldsymbol{x}_2, t_2 \rangle, \ldots, \langle \boldsymbol{x}_N, t_N \rangle\}$
- $t_i \in \{C_1, \ldots, C_K\}$ is discrete: Classification
- $\boldsymbol{x}_i$ can be continuous or discrete
- Learning problem: to construct y: $\mathcal{X} \rightarrow \{C_1, \ldots, C_K\}$
- E.g., MINST data classification
  - t_i is the digital 0 to 9
  - $\boldsymbol{x}_i$ is an image

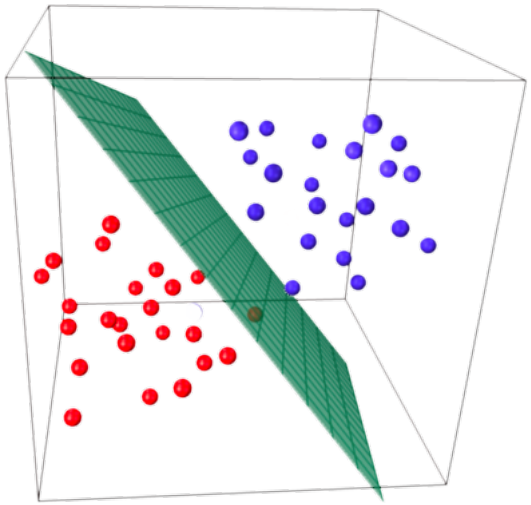
    x_i is usually mapped to a vector via a feature extraction function $\emptyset(x_i)$ explicitly or implicitly

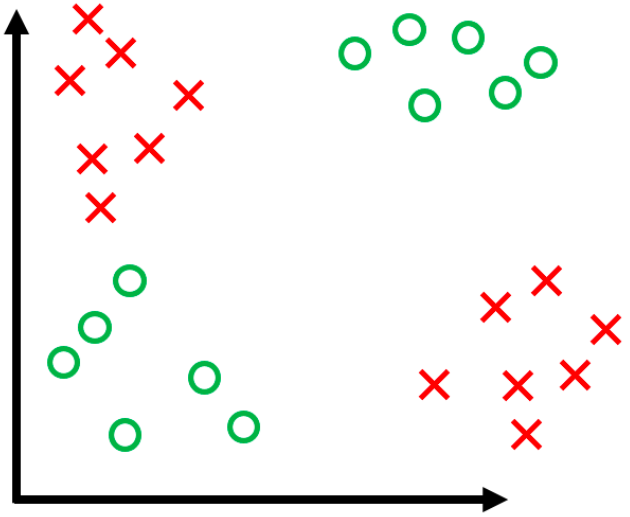


Linear Separability

MACQUARIE University

- Linearly separable: Classes of labelled data sets can be separated exactly by linear decision surfaces (boundaries)





- A classic linearly inseparable example: XOR data

2-class Discriminant Functions

MACQUARIE University

- The linear decision surface should be a hyperplane

$$\boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$$

  - M-dimensional feature vector $\boldsymbol{x}$
  - Classes: C_1 and C_2, or $t_i \in \{0, 1\}$
  - $\boldsymbol{w}$: weight vector (column-based)
  - w_0 bias or intercept

- The discriminant function (the predictive model)

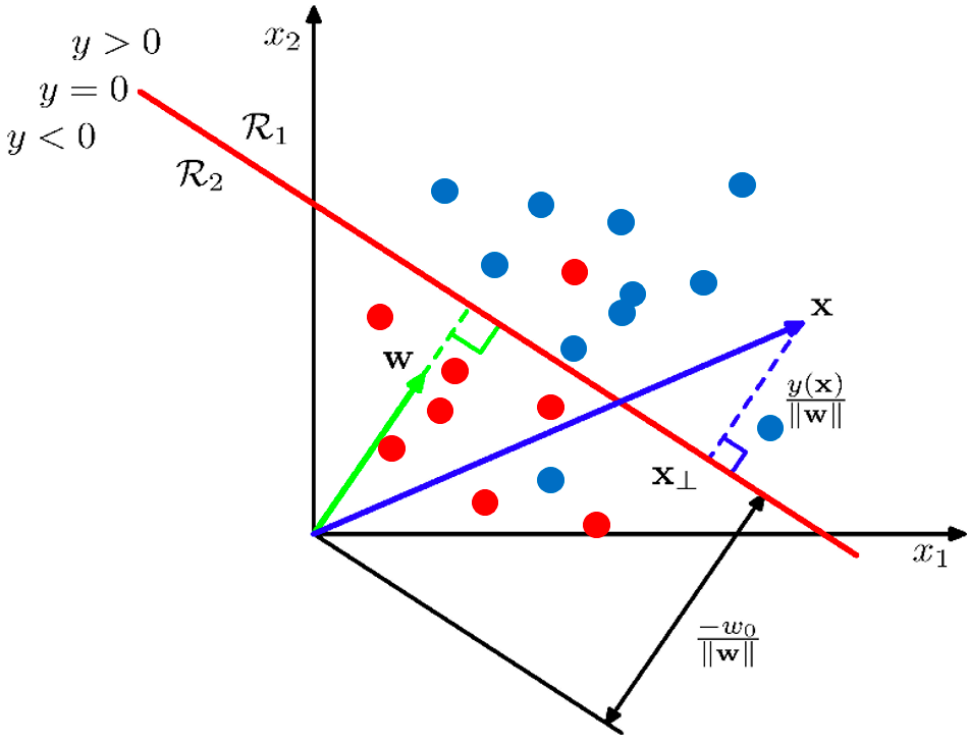
$$y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$$

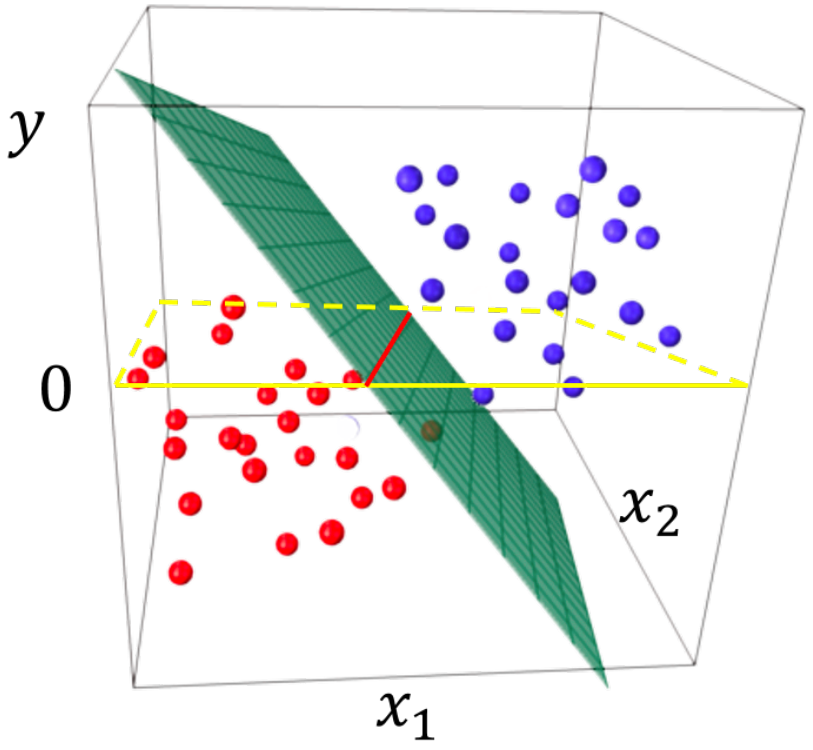
- Decision rule

$$Label(\boldsymbol{x}) \leftarrow \begin{cases} C_1, & y(\boldsymbol{x}) \geq 0 \\ C_2, & y(\boldsymbol{x}) < 0 \end{cases}$$

Geometry Perspective

- E.g., 2-dimensional input space





Geometry Perspective

MACQUARIE University

- $\boldsymbol{w}$ determines the orientation of the decision surface
  - Assume two points $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ on surface
  - $\because y(\boldsymbol{x}_A) = y(\boldsymbol{x}_B) = 0, \boldsymbol{w}^T(\boldsymbol{x}_A - \boldsymbol{x}_B) = 0$
- Normal distance from the origin to the decision surface

$$\frac{\boldsymbol{w}^T \boldsymbol{x}_A}{\|\boldsymbol{w}\|} = -\frac{w_0}{\|\boldsymbol{w}\|}$$

  - w_0 determines the location of the decision surface
- Relationship between $y(\boldsymbol{x})$ and the signed distance of $\boldsymbol{x}$ to the decision surface (denoted as r)

$$r = \frac{y(\boldsymbol{x})}{\|\boldsymbol{w}\|}$$

  - $\because \boldsymbol{x} = \boldsymbol{x}_\perp + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$; multiply the equation by $\boldsymbol{w}$ and add w_0

Multiple-class Cases

MACQUARIE University

- $t_i \in \{C_1, \ldots, C_K\}$
- K-class discriminant functions

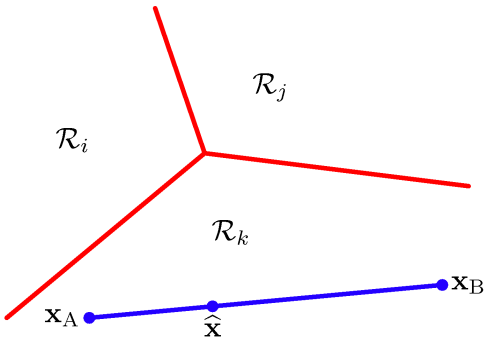
$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0}$$

  - $1 \leq k \leq K$

- Decision rule

$$Label(\boldsymbol{x}) \leftarrow \arg\max_k \{y_k(\boldsymbol{x})\}$$

- Decision regions are always connected and convex



Least Squares for Classification

MACQUARIE University

- 1-of-K binary coding scheme (a.k.a. one-hot coding)

$$C_k \Leftrightarrow \left(0, \ldots, 0, \underbrace{1}_{k^{th}\ position}, 0, \ldots, 0\right)^T$$

- Matrix denotation for the model

$$\boldsymbol{y}(\boldsymbol{x}) = \widetilde{\boldsymbol{W}}^T \tilde{\boldsymbol{x}}$$

  - $\tilde{\boldsymbol{x}} \equiv \langle 1, x_1, \ldots, x_M \rangle$
  - $\widetilde{\boldsymbol{W}}$ includes both coefficients and intercepts

- Given dataset $\mathcal{D}$, we can minimzse sum-of-squares error to learn the model parameters $\widetilde{\boldsymbol{W}}$

Least Squares for Classification

MACQUARIE University

$$E_D(\widetilde{\boldsymbol{W}}) = \frac{1}{2}\mathrm{Tr}\{\boldsymbol{E}^T\boldsymbol{E}\}$$

$$\boldsymbol{E} = \widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{W}} - \boldsymbol{T} = \begin{bmatrix} \boldsymbol{e}_1^T \\ \vdots \\ \boldsymbol{e}_N^T \end{bmatrix}, \qquad \boldsymbol{e}_i = \widetilde{\boldsymbol{w}}^T\boldsymbol{x}_i - \boldsymbol{t}_i$$

$$\boldsymbol{E}^T\boldsymbol{E} = \begin{bmatrix} \boldsymbol{e}_1^T \\ \vdots \\ \boldsymbol{e}_N^T \end{bmatrix}^T \begin{bmatrix} \boldsymbol{e}_1^T \\ \vdots \\ \boldsymbol{e}_N^T \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^N e_{i1}^2 & \sum_{i=1}^N e_{i1}e_{i2} & \cdots \\ \sum_{i=1}^N e_{i2}e_{i1} & \ddots & \vdots \\ \vdots & \cdots & \sum_{i=1}^N e_{ik}^2 \end{bmatrix}$$

Least Squares for Classification

MACQUARIE University

- Set the derivative w.r.t. $\widetilde{\boldsymbol{W}}$ to $\mathbf{0}$ to obtain

$$\widetilde{\boldsymbol{W}} = \left(\widetilde{\boldsymbol{X}}^T\widetilde{\boldsymbol{X}}\right)^{-1}\widetilde{\boldsymbol{X}}^T\boldsymbol{T} = \widetilde{\boldsymbol{X}}^{\dagger}\boldsymbol{T}$$

  - $\widetilde{\boldsymbol{X}}^{\dagger}$ is the pseudo-inverse of the matrix of $\widetilde{\boldsymbol{X}}$

- The discriminant function

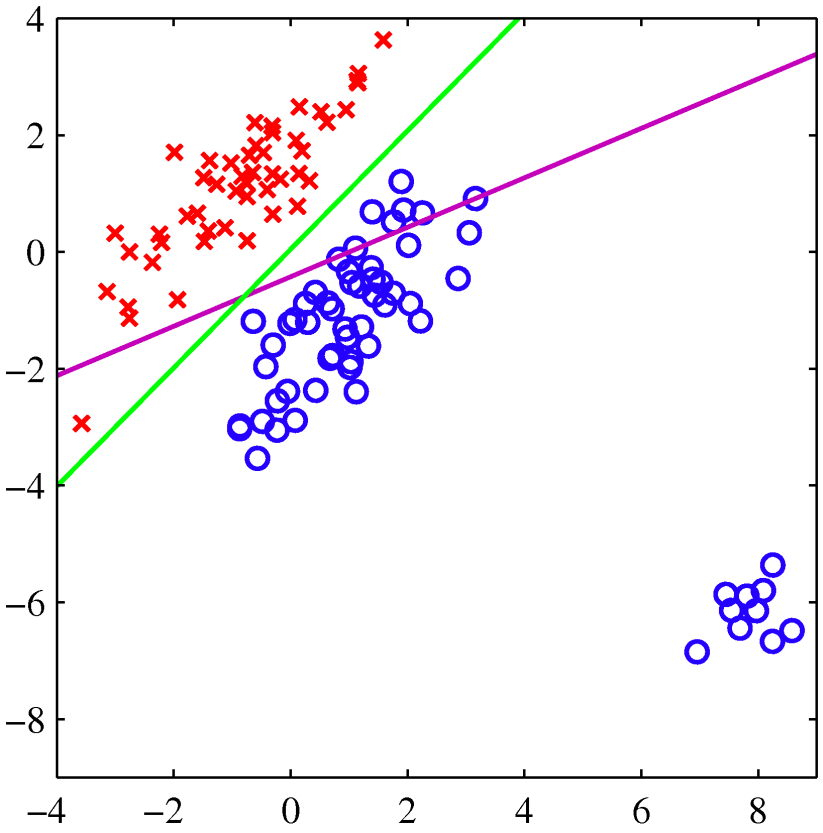
$$\boldsymbol{y}(\boldsymbol{x}) = \widetilde{\boldsymbol{W}}^T\widetilde{\boldsymbol{x}} = \boldsymbol{T}^T\left(\widetilde{\boldsymbol{X}}^{\dagger}\right)^T\widetilde{\boldsymbol{x}}$$

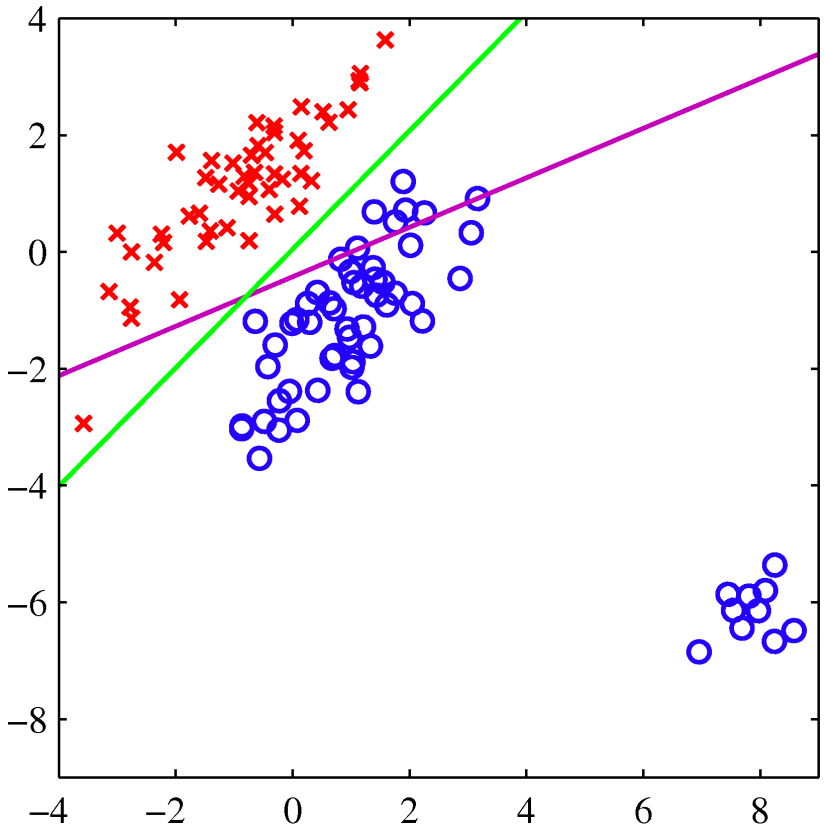
- Issues: poor robustness w.r.t. outliers
  - Least squares method corresponds to maximum likelihood under Gaussian conditional distribution
  - Binary target vectors clearly fail to follow Gaussian

Least Squares for Classification

MACQUARIE University

- Robustness w.r.t. Outliers





Perceptron

MACQUARIE University

- The Perceptron two-class model

$$y(\boldsymbol{x}) = f(\boldsymbol{w}^T \emptyset(\boldsymbol{x}))$$

- $f(\cdot)$ is a step function

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

- Target value encoding $+1$ for C_1, and -1 for C_2
- Observations
  - If $\boldsymbol{x}_i$ in C_1(i.e., $t_n = +1$), $\boldsymbol{w}^T \emptyset(\boldsymbol{x}_i) > 0$
  - If $\boldsymbol{x}_i$ in C_2(i.e., $t_n = -1$), $\boldsymbol{w}^T \emptyset(\boldsymbol{x}_i) < 0$
- $\Longrightarrow \boldsymbol{w}^T \emptyset(\boldsymbol{x}_i) t_n > 0$

Perceptron

MACQUARIE University

- Perceptron criterion: the error function used to determine $\boldsymbol{w}$
  - Correct classification: 0
  - Misclassification error: $-\boldsymbol{w}^T \emptyset(\boldsymbol{x}_i) t_i$
- We hope to drive $-\boldsymbol{w}^T \emptyset(\boldsymbol{x}_i) t_i$ to 0
- The Perceptron criterion function

$$E_P(\boldsymbol{w}) = -\sum_{i \in \mathcal{M}} \boldsymbol{w}^T \emptyset(\boldsymbol{x}_i) t_i$$

  - $\mathcal{M}$ denotes the set of misclassified patterns
- Key: How to determine $\mathcal{M}$?

Perceptron

MACQUARIE University

- Rewrite $E_P(\boldsymbol{w})$ as

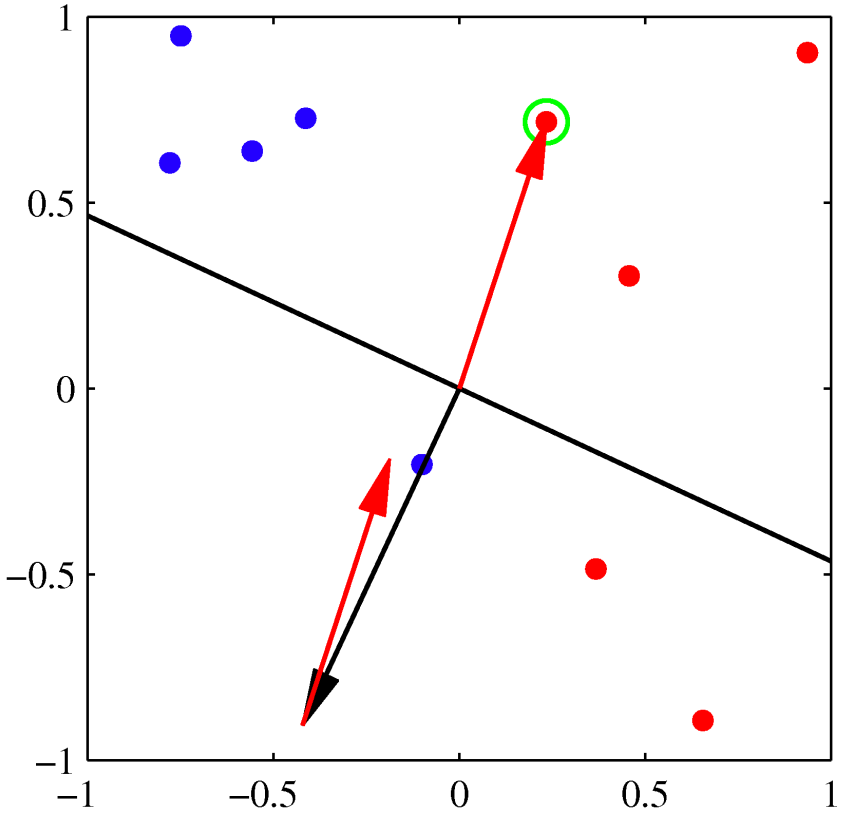
$$J(\boldsymbol{w}) = E_P(\boldsymbol{w}) = \sum_{i=1}^{n} J_i(\boldsymbol{w})$$

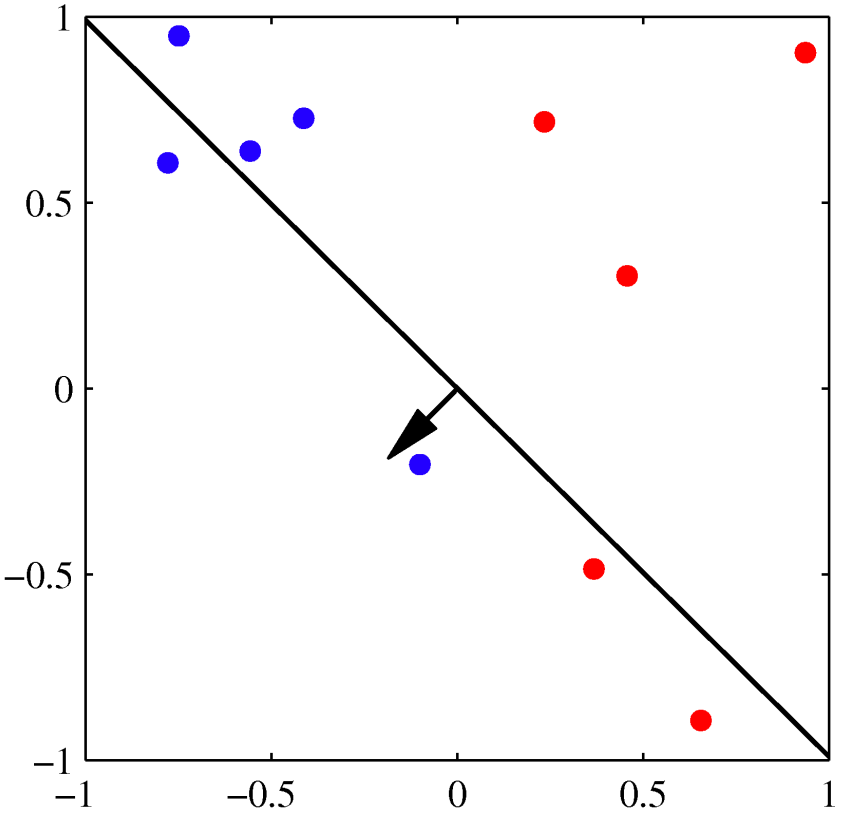
  - $J_i(\boldsymbol{w}) = \begin{cases} 0, & \boldsymbol{w}^T \emptyset(\boldsymbol{x}_i) t_i \geq 0 \\ -\boldsymbol{w}^T \emptyset(\boldsymbol{x}_i) t_i, & \boldsymbol{w}^T \emptyset(\boldsymbol{x}_i) t_i < 0 \end{cases}$

- Then, we can use gradient descent method to estimate $\boldsymbol{w}$
- Perceptron convergence theorem guarantees solutions if existing (data set is linearly separable)

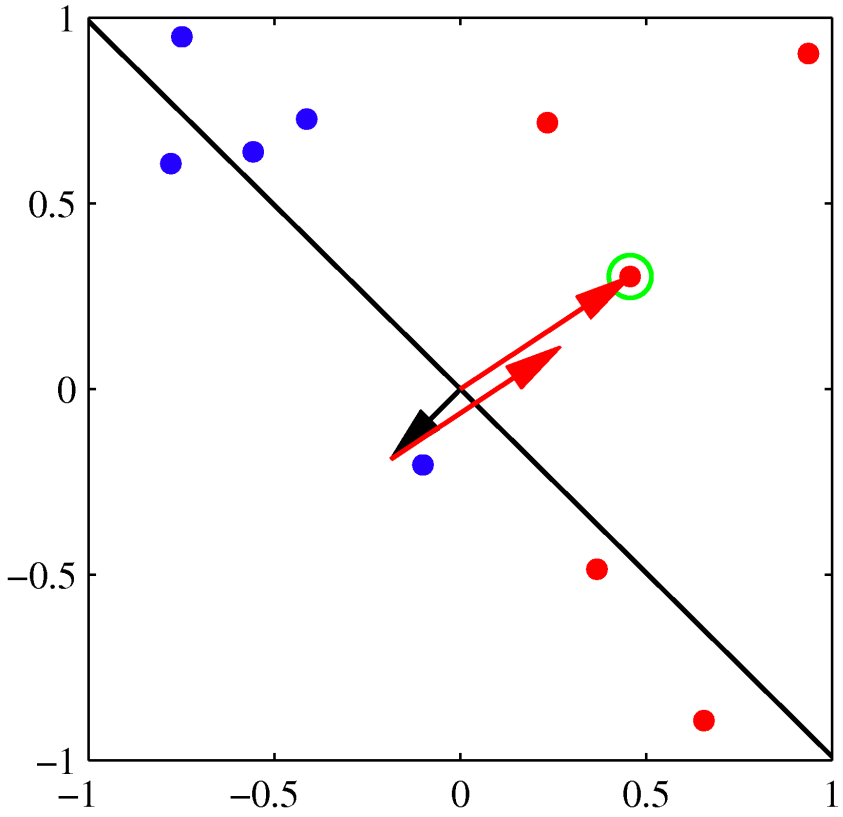
Perceptron

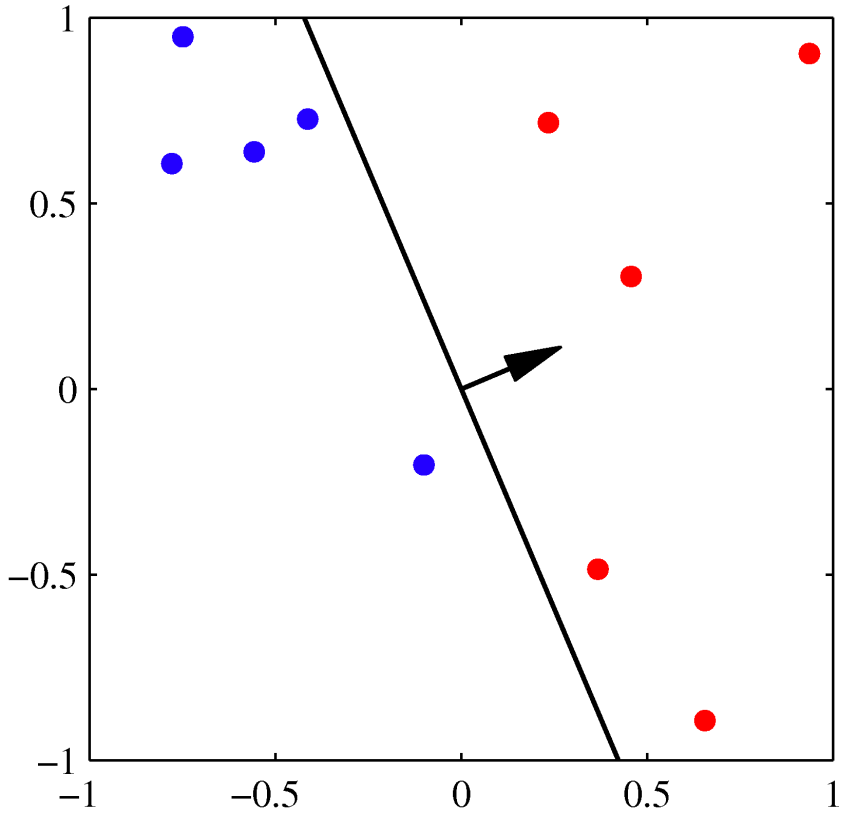
MACQUARIE University





Perceptron





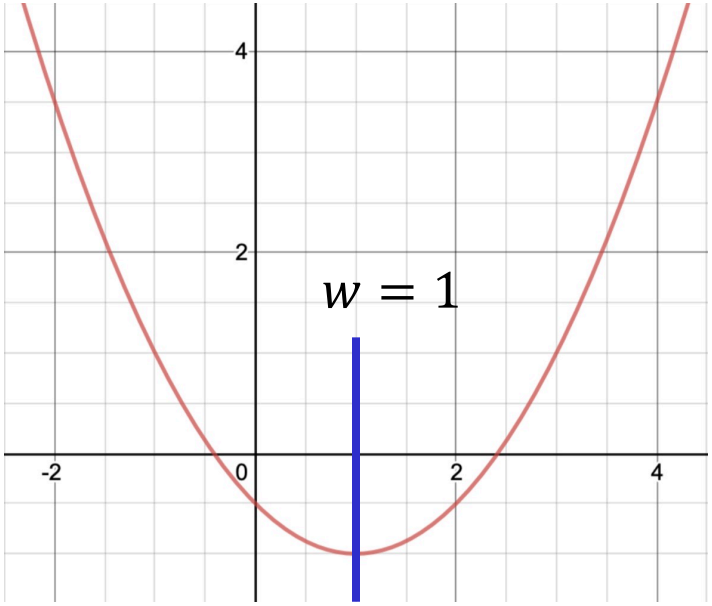
Gradient Descent

MACQUARIE University

- Question: which w minimize the following function?

$$f(w) = \frac{1}{2}(w-1)^2 - 1$$

- Option I: observe its curve (middle/high school)



Gradient Descent (Cont'd)

MACQUARIE University

- Option 2: analytical method with derivative (calculus)

$$f'(w) = \frac{1}{2} \cdot 2 \cdot (w-1) \cdot 1 - 0 = w - 1$$

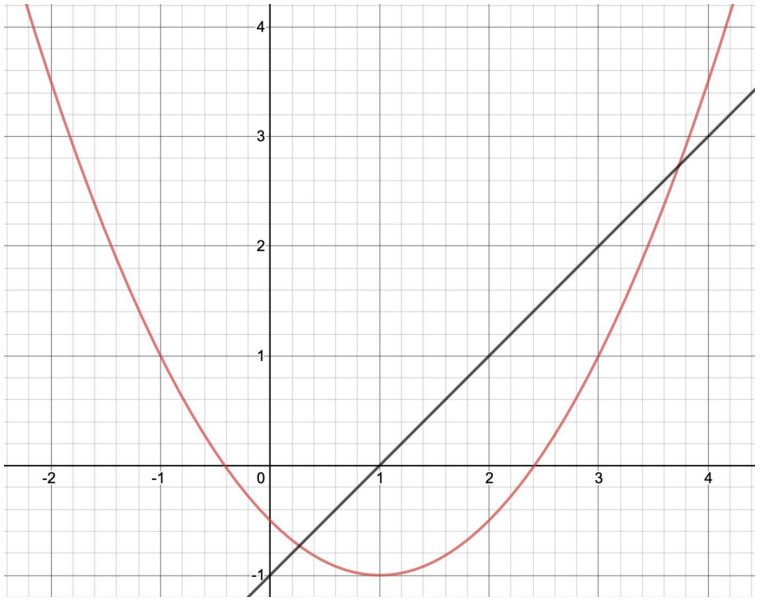
  - To find stationary points, let $f'(w) = 0$

$$f'(w) = w - 1 = 0$$

$$\Rightarrow w^* = 1$$

$$f''(w) = (w-1)' = 1 > 0$$

  - $w^* = 1$ is the value minimizing the function
  - Linear regression adopt this method to compute weights



Gradient Descent (Cont'd)

MACQUARIE University

- Option 3: Numerical method
  - Randomly initialize the value ($w^{(0)}$), and optimize it gradually
  - Needs an update rule (still make use of derivative)

$$w^{(\tau+1)} \leftarrow w^{(\tau)} - \eta f'(w)\Big|_{w^{(\tau)}} = w^{(\tau)} - \eta(w^{(\tau)} - 1)$$

    - η is step size (or learning rate)
  - Stops after the result is good enough
    - E.g., difference between two iterations is small enough
- Unlike Option 2, we don't need to solve $f'(w) = 0$
  - When a function is more complicated, solving the equation system analytically is really challenging
  - So, this method is more general with an approximate result

Gradient Descent (Cont'd)

MACQUARIE University

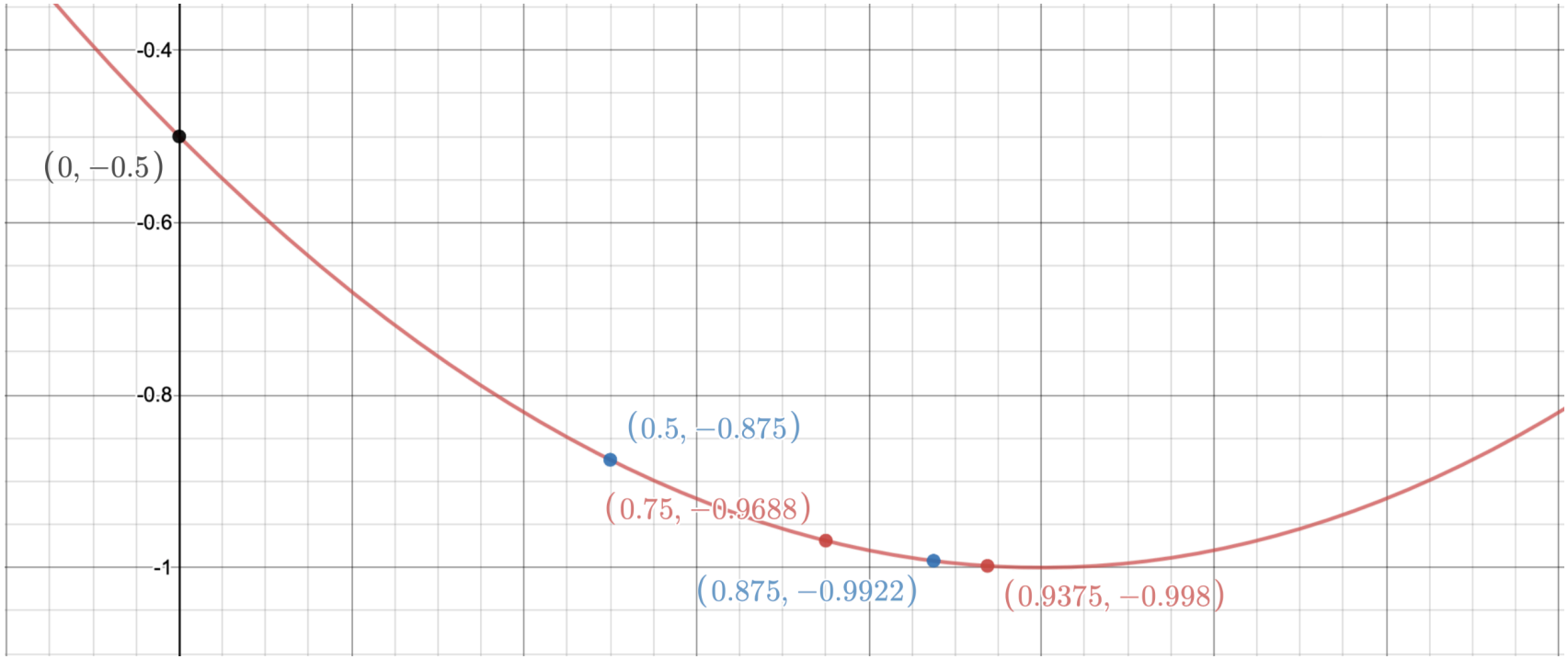
- Option 3: Numerical method
  - Show an example when $w^{(0)} = 0$, and $\eta = \frac{1}{2}$

τ	w	$f'(w)$	$f(w)$	$\lvert f^{(\tau+1)}(w) - f^{(\tau)}(w)\rvert$
0	0	-1	$-\frac{1}{2}$	
1	$0 - \frac{1}{2}(-1) = \frac{1}{2}$	$-\frac{1}{2}$	$-\frac{7}{8}$	0.375
2	$\frac{1}{2} - \frac{1}{2}\left(-\frac{1}{2}\right) = \frac{3}{4}$	$-\frac{1}{4}$	$-\frac{31}{32}$	0.093
3	$\frac{3}{4} - \frac{1}{2}\left(-\frac{1}{4}\right) = \frac{7}{8}$	$-\frac{1}{8}$	$-\frac{127}{128}$	0.0234
4	$\frac{7}{8} - \frac{1}{2}\left(-\frac{1}{8}\right) = \frac{15}{16}$	$-\frac{1}{16}$	$-\frac{511}{512}$	0.0058
…	…	…	…	…

Gradient Descent (Cont'd)

MACQUARIE University

- Option 3: Numerical method
  - $w^* \leftarrow w^{(4)} = \frac{15}{16}$ is a good approximate to the true value 1

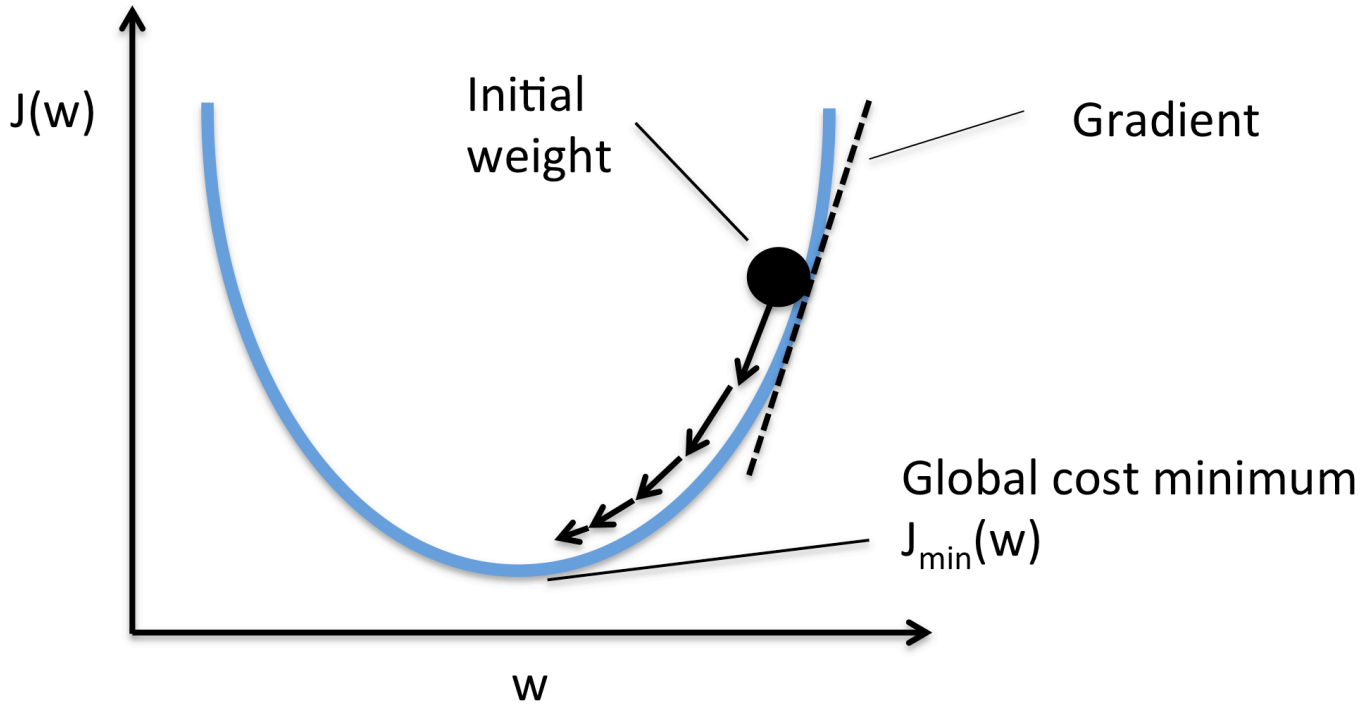


Gradient Descent (Cont'd)

MACQUARIE University

- General rule for a cost function $J(w)$

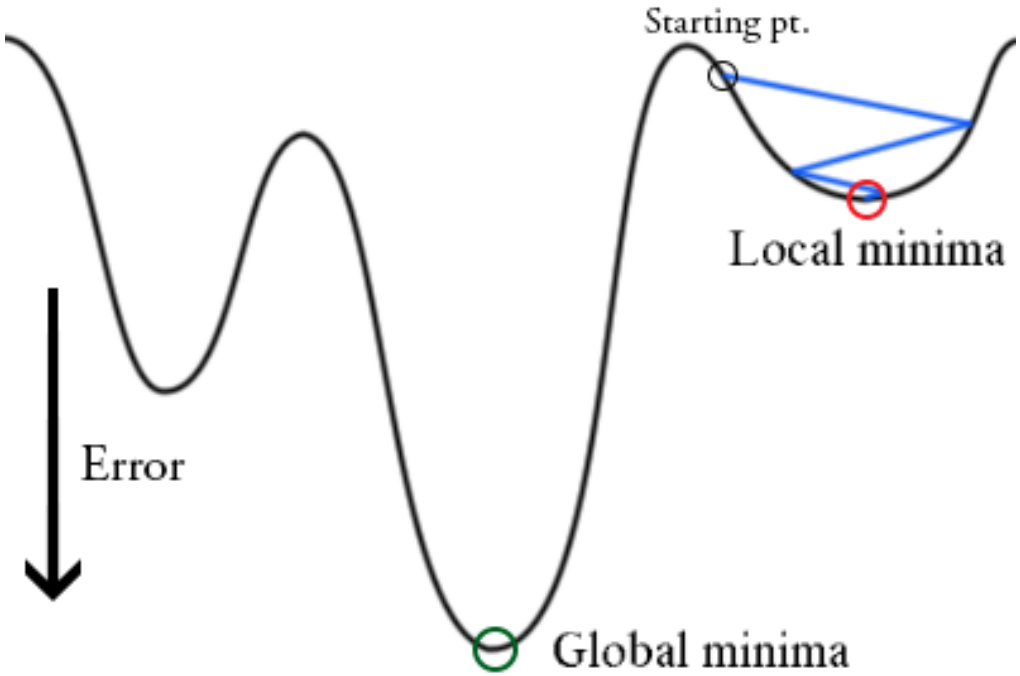
$$w^{(\tau+1)} \leftarrow w^{(\tau)} - \eta \cdot J'(w)\Big|_{w^{(\tau)}}$$



Gradient Descent (Cont'd)

MACQUARIE University

- What if the function is non-convex?
  - Can get stuck in local minima



Gradient Descent (Cont'd)

MACQUARIE University

- Extend to multiple variable function
  - This is a much more common case in machine learning
- Beyond 'derivative', we need 'gradient'
- For a function $f(\boldsymbol{w}) = f(w_1, \dots, w_i, \dots, w_n)$
- Gradient of $f(\boldsymbol{w})$

$$\nabla f(\boldsymbol{w}) = \left(\frac{\partial f(\boldsymbol{w})}{\partial w_1}, \dots, \frac{\partial f(\boldsymbol{w})}{\partial w_i}, \dots, \frac{\partial f(\boldsymbol{w})}{\partial w_n}\right)$$

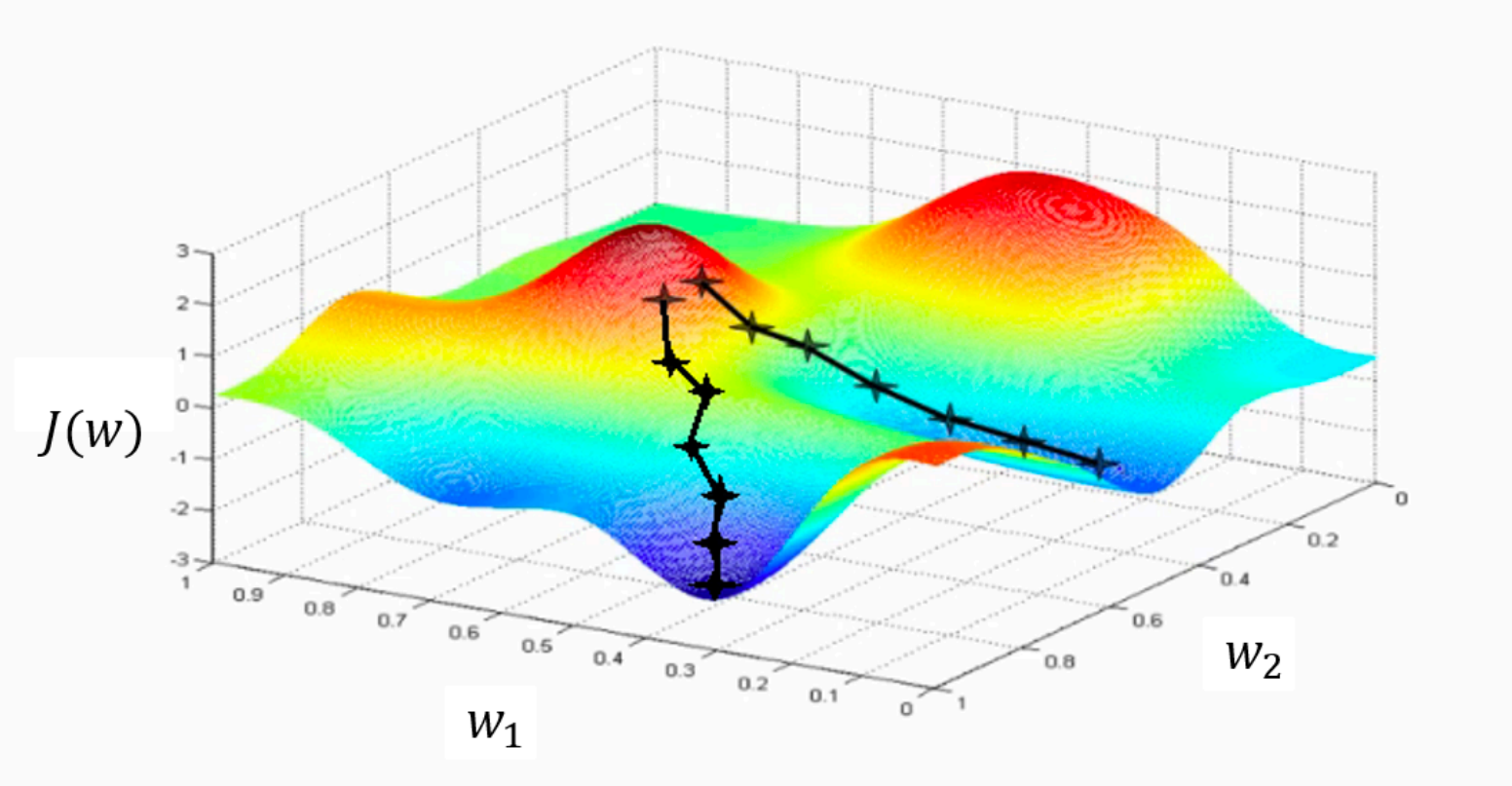
  - $\frac{\partial f(\boldsymbol{w})}{\partial w_i}$ is partial derivative w.r.t. w_i
  - Direction: direction of steepest increase
  - Magnitude: rate of change in that direction

Gradient Descent (Cont'd)

MACQUARIE University

- Update rule for a cost function $J(\boldsymbol{w})$

$$\boldsymbol{w}^{(\tau+1)} \leftarrow \boldsymbol{w}^{(\tau)} - \eta \cdot \nabla J(\boldsymbol{w}) \Big|_{\boldsymbol{w}^{(\tau)}}$$

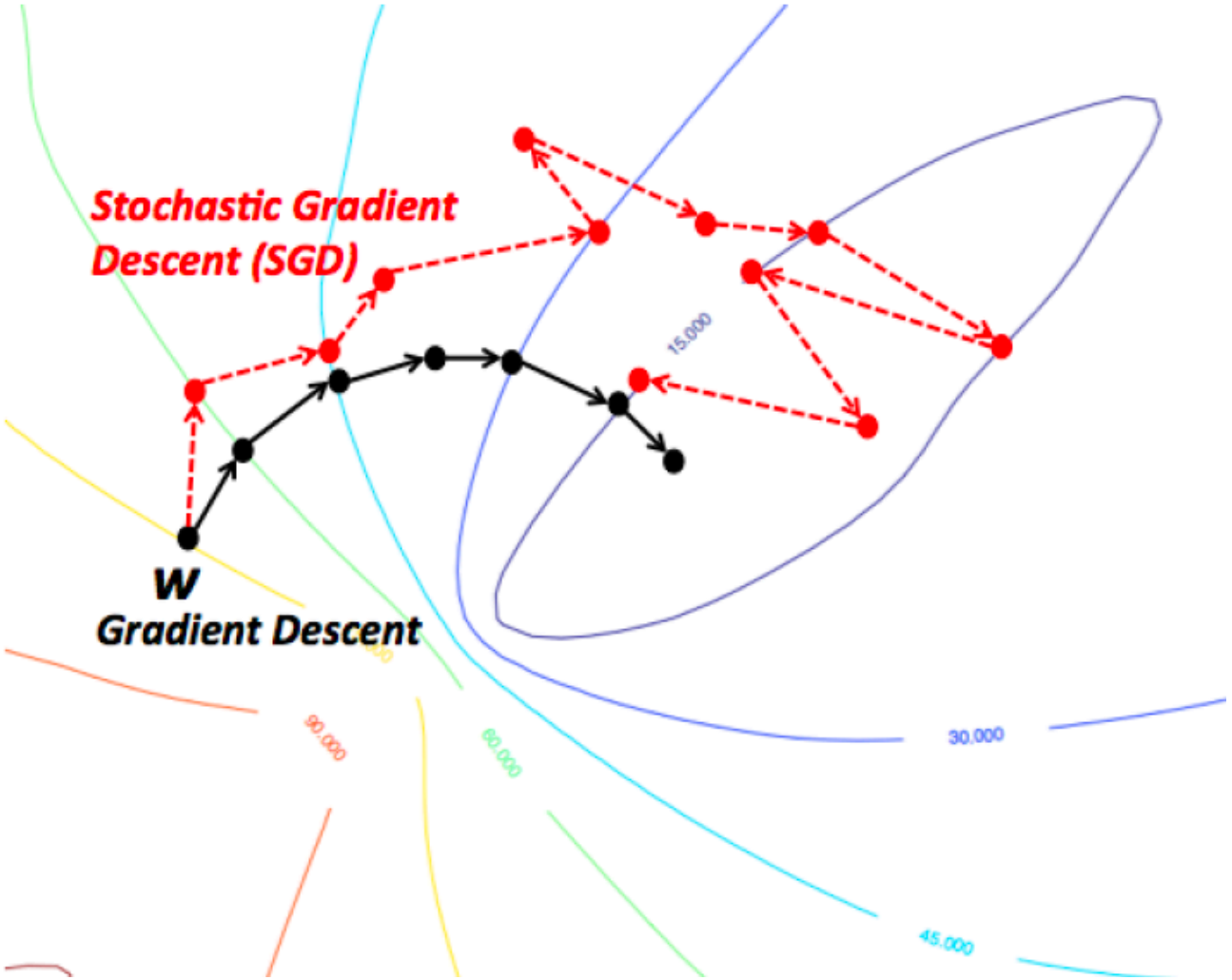


Gradient Descent

MACQUARIE University

- Batch Gradient Descent
  - Uses the whole training set to compute gradients
  - Slow when the training set is large
- Stochastic Gradient Descent
  - Picks a random instance at each step to compute gradients
  - Fast but not optimal
  - Learning rate is determined by a learning schedule
- Mini-batch Gradient Descent
  - Computes the gradient on a small random set of instances
  - Gets performance boost over SGD when using GPUs

Stochastic Gradient Descent



Nearest Neighbors Classification

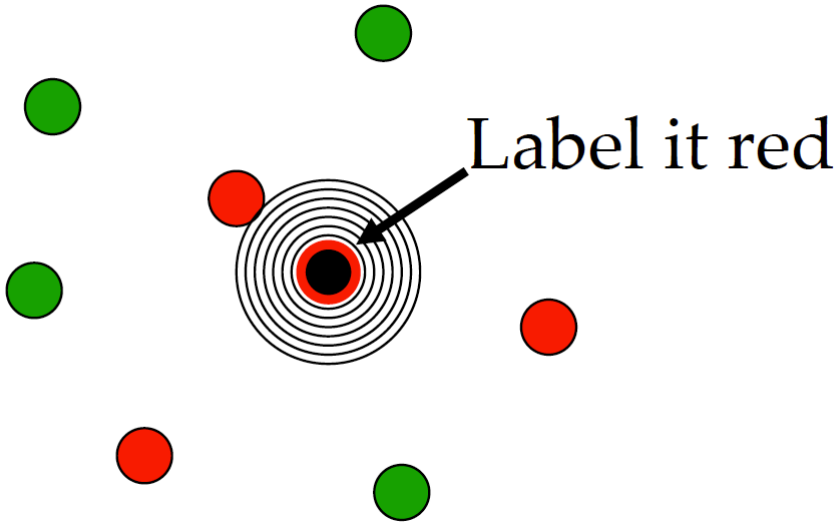
MACQUARIE University

- Idea: instance-based learning
  - Similar examples have similar labels
  - Classify new examples like similar training examples
- Algorithm
  - Given some new example $\boldsymbol{x}$ for which we need to predict its class y
  - Find the most similar training examples
  - Classify $\boldsymbol{x}$ "like" these most similar examples
- Questions:
  - How to determine similarity?
  - How many similar training examples to consider?

1-Nearest Neighbor

MACQUARIE University

- One of the simplest classifiers
- Basic idea: label a new point the same as the closest known data instance
- E.g.,



1-Nearest Neighbor (Cont'd)

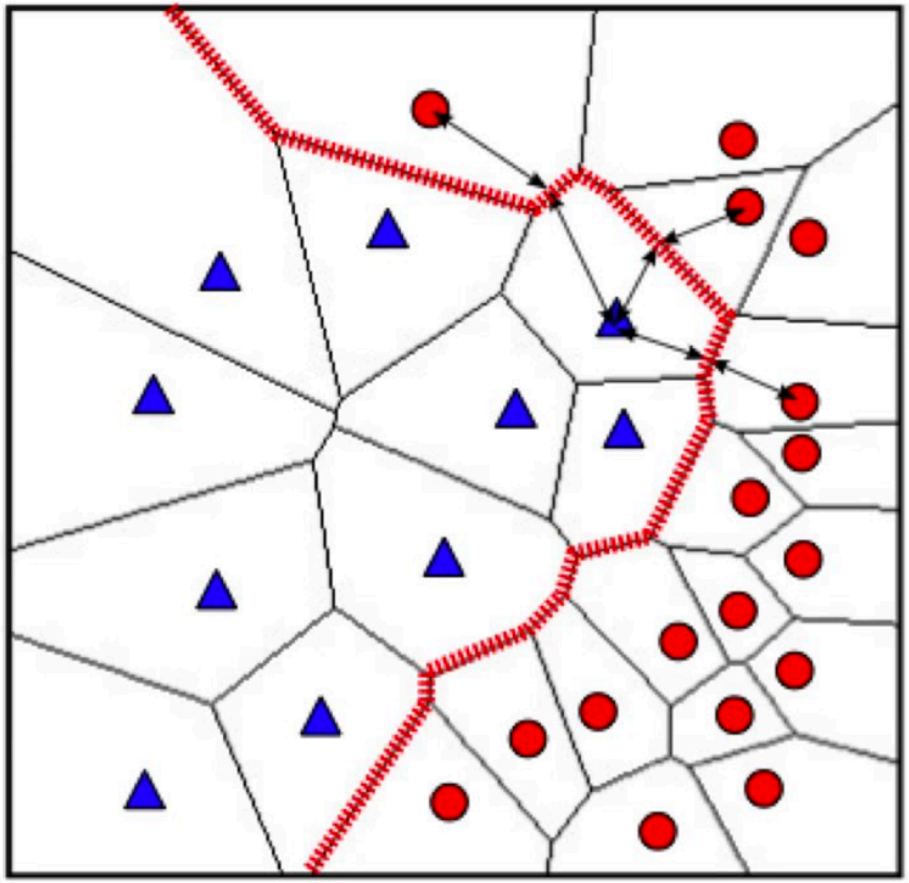
MACQUARIE University

- A distance metric (to measure similarity)
  - Euclidean distance is commonly-used
  - When different units are used for each dimension
    - Standardization can apply
  - For categorical data, we can use hamming distance
    - $d(x_1, x_2)$ = number of features on which x_1 and x_2 differ
  - Others (e.g., cosine, Manhattan)
- How many nearby neighbors to look at?
  - Only one
- How to fit with training data instances?
  - Just predict the same output as the nearest neighbor

1-Nearest Neighbor (Cont'd)

MACQUARIE University

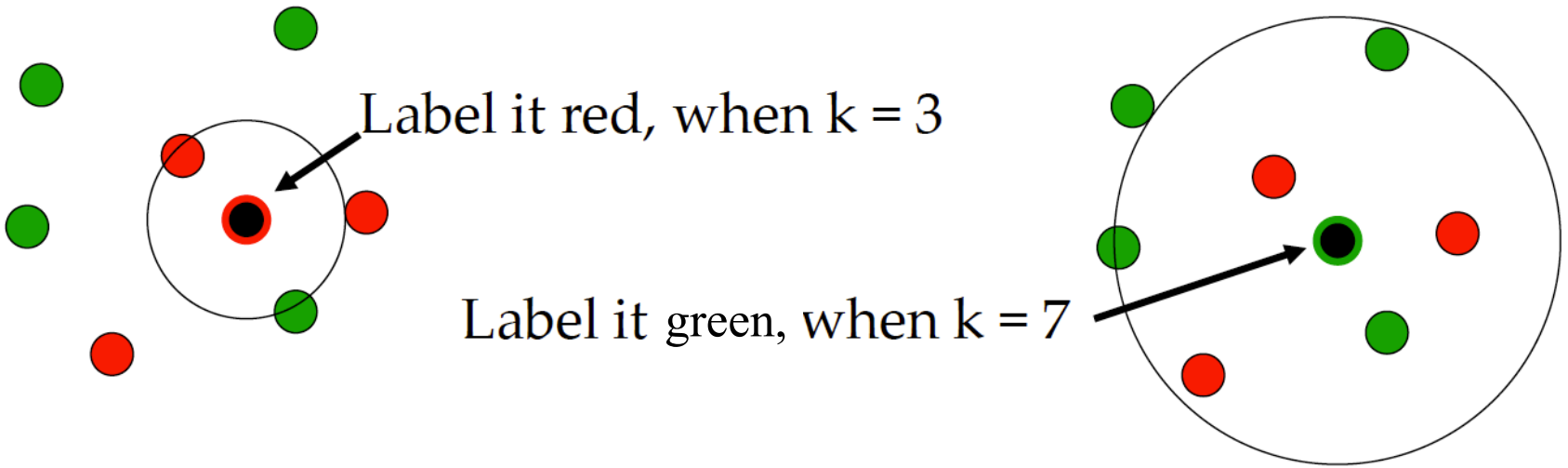
- The resultant space partition is a Voronoi diagram



K-Nearest Neighbor

MACQUARIE University

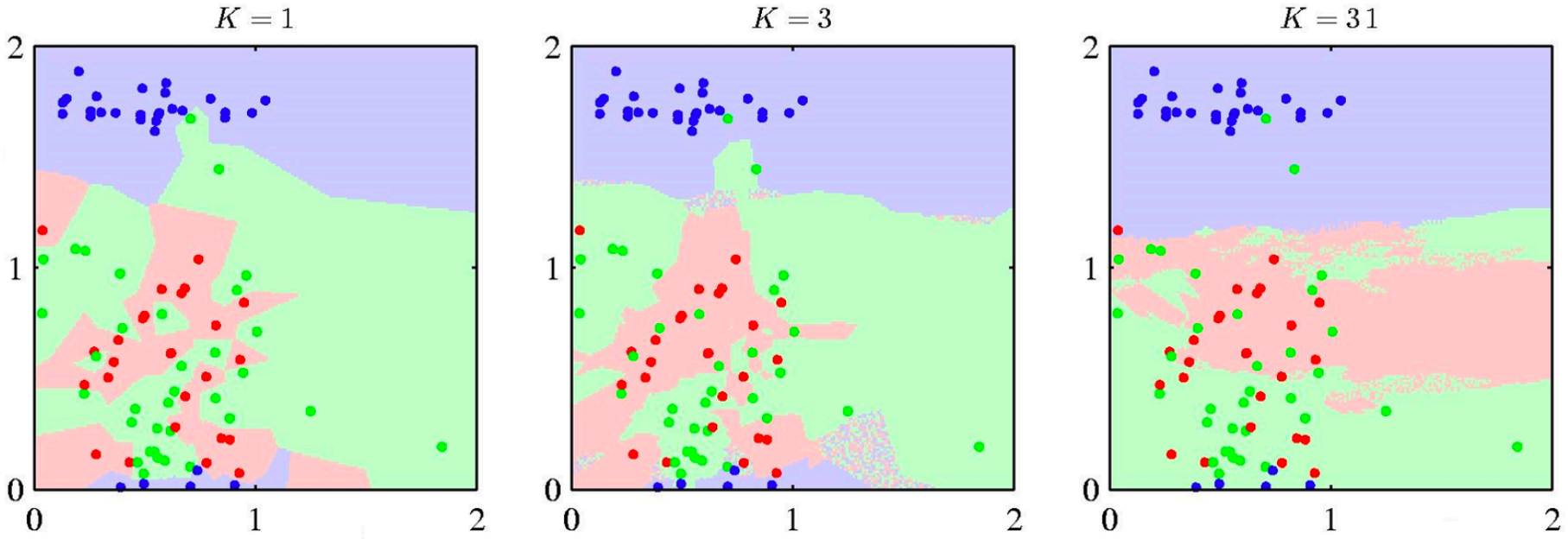
- Generalizes 1 NN to smooth away noise in the labels
- A new data instance is now assigned the most frequent label of its k nearest neighbors
- E.g., $k = 3$, and $k = 7$



K-Nearest Neighbor (Cont'd)

MACQUARIE University

- K acts as a smoother and controls model complexity



Selecting k

MACQUARIE University

- Increase k
  - Make KNN less sensitive to noise
  - Avoid overfitting
- Decrease k
  - Allow capturing finer structure of space
  - Avoid underfitting
- Pick k not too large, but not too small
  - Data-specific
  - This is model selection by hyperparameter tuning
  - Cross validation can be used to find a suitable k
  - Theoretically, this is related to bias-variance tradeoff