Data Mining

CS57300
Purdue University

Bruno Ribeiro

January 22, 2018

Differences Between Classification & Prediction

- Classification
    - Observing feature x predict discrete-valued label y

- Prediction (point estimate)
    - Observing feature x predict of real-value label y
    - Forecasting: predictions + confidence intervals

- Ranking prediction (rank estimate)
    - Predict item ranking
        - E.g.: Google results, Netflix recommendations

Predictive modeling

- Data representation:
  - Training set: Paired attribute vectors and labels $<y(i), \mathbf{x}(i)>$
    or
    $n \times p$ tabular data with label (y) and p-1 attributes $(\mathbf{x})$

- Task: estimate a predictive function $f(\mathbf{x};\theta)=y$
  - Assume that there is a function $y=f(x)$ that **maps** data instances $(\mathbf{x})$ to labels (y)
  - Construct a model that approximates the mapping
    - Classification: if y is categorical
    - Regression: if y is real-valued

Modeling approaches

Learning predictive models

- Choose a **data representation**
- Select a **knowledge representation** (a "model")
  - Defines a **space** of possible models $M=\{M_1, M_2, \ldots, M_k\}$
- Use **search** to identify "best" model(s)
  - Search the space of models (i.e., with alternative structures and/or parameters)
  - Evaluate possible models with **scoring function** to determine the model which best fits the data

Scoring functions

- Given a model M and dataset D, we would like to "score" model M with respect to D
    - Goal is to rank the models in terms of their utility (for capturing D) and choose the "best" model
    - Score function can be used to search over ***parameters*** and/or ***model structure***
- Score functions can be different for:
    - Models vs. patterns
    - Predictive vs. descriptive functions
    - Models with varying complexity (i.e., number parameters)

Predictive scoring functions

- Assess the quality of predictions for a set of instances
  - Measures **difference** between the prediction M makes for an instance i and the true class label value of i

$$S(M) = \underbrace{\sum_{i=1}^{N_{test}}}_{\text{Sum over examples}} \underbrace{d}_{\text{Distance between predicted and true}}\big[\underbrace{f(x(i); M)}_{\text{Predicted class label for item } i}, \underbrace{y(i)}_{\text{True class label for item } i}\big]$$

Predictive scoring functions

- Common score functions:
  - Zero-one loss

$$S_{0/1}(M) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} I\left[f(x(i); M), y(i)\right]$$

$$\text{where } I(a,b) = \begin{cases} 1 & a \neq b \\ 0 & \text{otherwise} \end{cases}$$

  - Squared loss

$$S_{sq}(M) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \left[f(x(i); M) - y(i)\right]^2$$

Careful with definition of class labels!
Why?

  - More later…
- Do we minimize or maximize these functions?

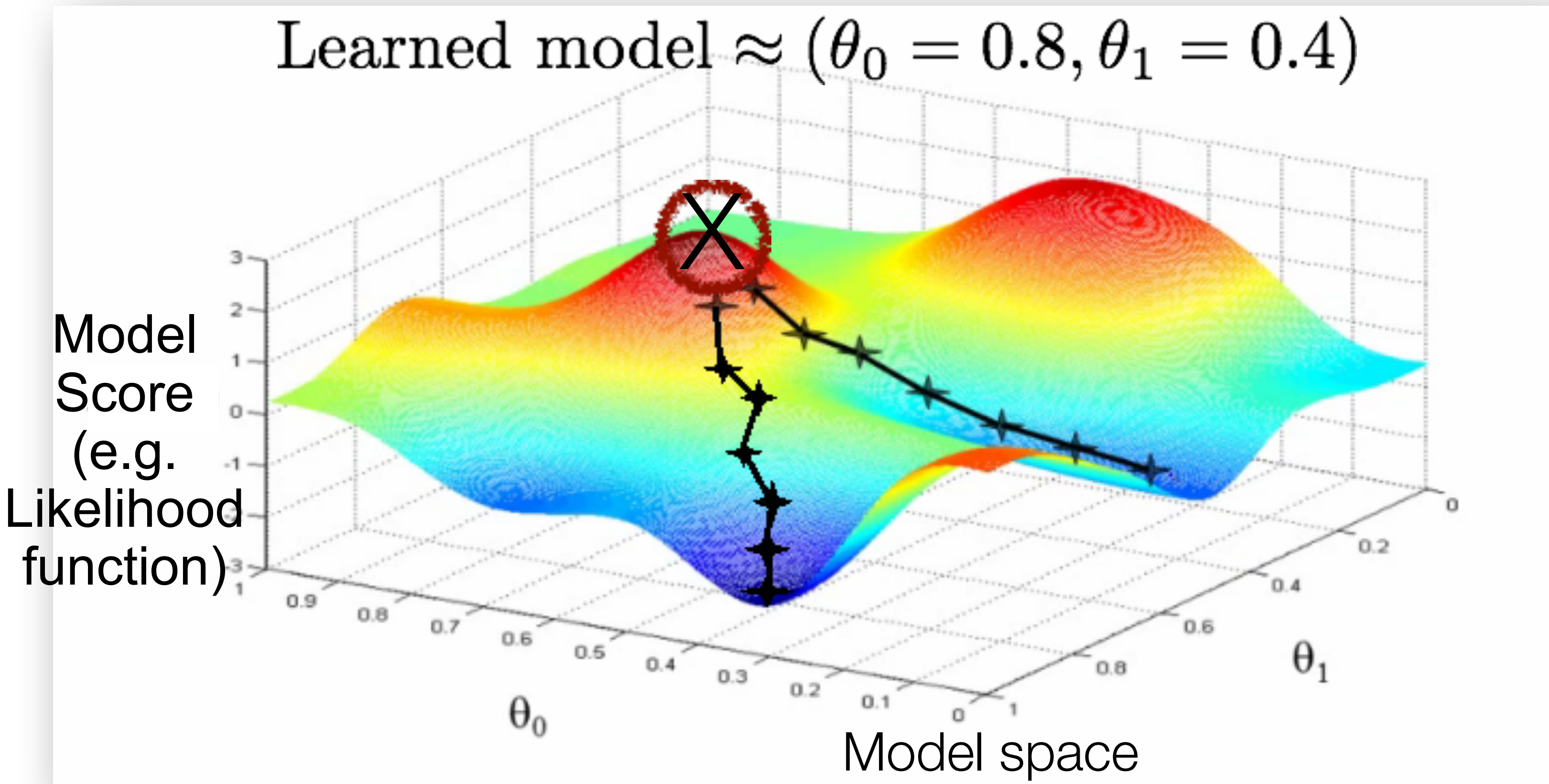
Scoring functions

- Guide search *inside* of algorithms
  - Select path in heuristic search
  - Decide when to stop
  - Identify whether to include model or pattern in output
- Evaluate results *outside* of algorithms
  - Measure the absolute quality of model or pattern
  - Compare the relative quality of different algorithms or outputs

Where's the search?

Find Parameters that Minimize Model Score (Error)

Usually we maximize (- score)



Alex Holehuse, Notes from Andrew Ng's Machine Learning Class, http://www.holehouse.org/mlclass/01_02_Introduction_regression_analysis_and_gr.html

Searching over models/patterns

- Consider a **space** of possible models
  $M=\{M_1, M_2, ..., M_k\}$ with parameters θ

- Search could be over model structures or parameters, e.g.:

  - **Parameters**: In a linear regression model, find the regression coefficients (β) that minimize squared loss on the training data

  - **Model structure**: In a decision trees, find the tree structure that maximizes accuracy on the training data

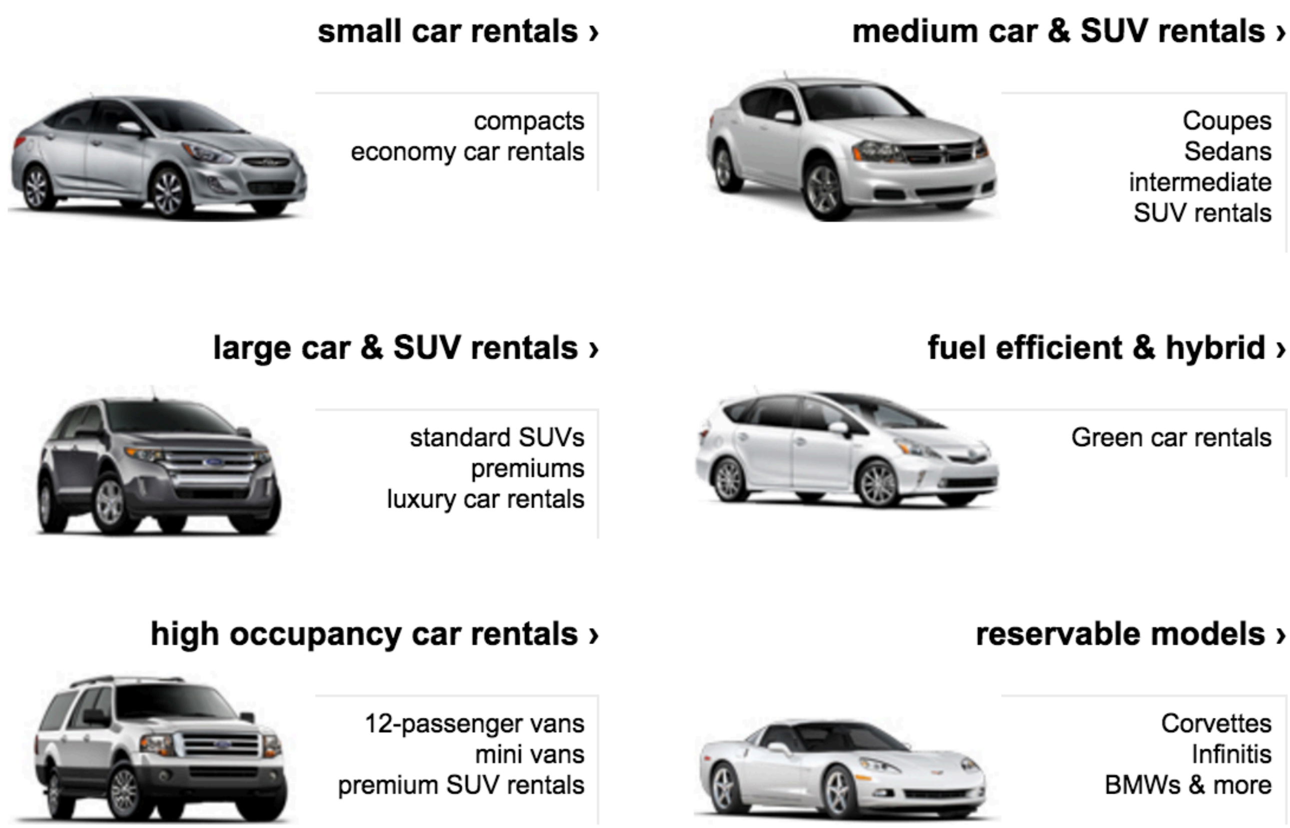
Optimization over score functions

- **Smooth** functions:
  - If a function is *smooth*, it is differentiable and the derivatives are continuous, then we can use gradient-based optimization
    - If function is *convex*, we can solve the minimization problem in closed form: $\nabla S(\theta)$ using **convex optimization**
    - If function is smooth but non-linear, we can use iterative search over the surface of S to find a local minimum (e.g., hill-climbing)
- **Non-smooth** functions:
  - If the function is *discrete*, then traditional optimization methods that rely on smoothness are not applicable. Instead we need to use **combinatorial optimization**

Linear Methods for Classification

Motivation 1/2

- Given x features of a car (length, width, mpg, maximum speed,…)
- Classify cars into categories based on x



Motivation 2/2

- A person arrives at the emergency room with a set of symptoms that could possibly be attributed to one of three medical conditions. Which of the three conditions does the individual have?
- An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, on the basis of the user's IP address, past transaction history, and so forth.
- On the basis of DNA sequence data for a number of patients with and without a given disease, a biologist would like to figure out which DNA mutations are deleterious (disease-causing) and which are not.

Linear Discriminant Function (Two Classes)

- Two classes
- x is a D-dimensional real-valued vector (set of features)
- y is the car class

$$y_c = \begin{cases} 1 & \text{, if car } c \text{ is "small"} \\ -1 & \text{, if car } c \text{ is "luxury"} \end{cases}$$

- Find linear discriminant weights $\mathbf{w}$

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

- Score function is the **squared error**

$$\sum_{c \in \mathrm{TestDataCars}} (y_c - y(x_c))^2$$

- **Search algorithm: least squares algorithm**

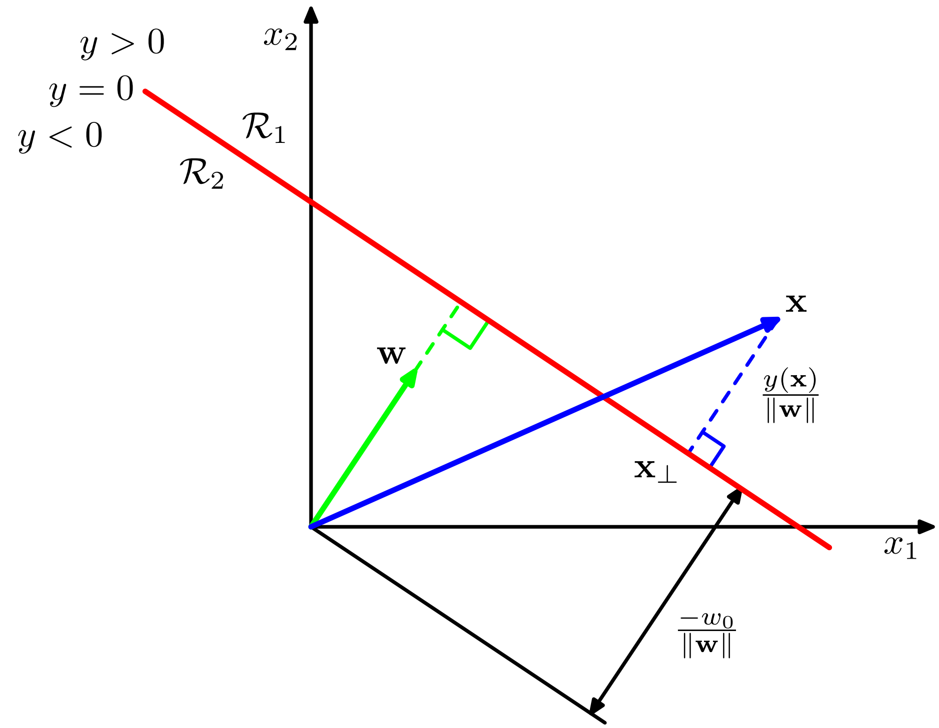


Figure: C. Bishop

How to Deal with Multiple Classes?

Naïve Approach: one vs. many Classification

- How to classify objects into multiple types?

$$y_c^{(1)} = \begin{cases} 1 & \text{, if car } c \text{ is ``small''} \\ -1 & \text{, if car } c \text{ is ``luxury''} \end{cases}$$

$$y_c^{(2)} = \begin{cases} 1 & \text{, if car } c \text{ is ``small''} \\ -1 & \text{, if car } c \text{ is ``medium''} \end{cases}$$

$$y_c^{(3)} = \begin{cases} 1 & \text{, if car } c \text{ is ``medium''} \\ -1 & \text{, if car } c \text{ is ``luxury''} \end{cases}$$

Might work OK in some scenarios… but not clear in this case

Issue with using binary classifiers for K classes

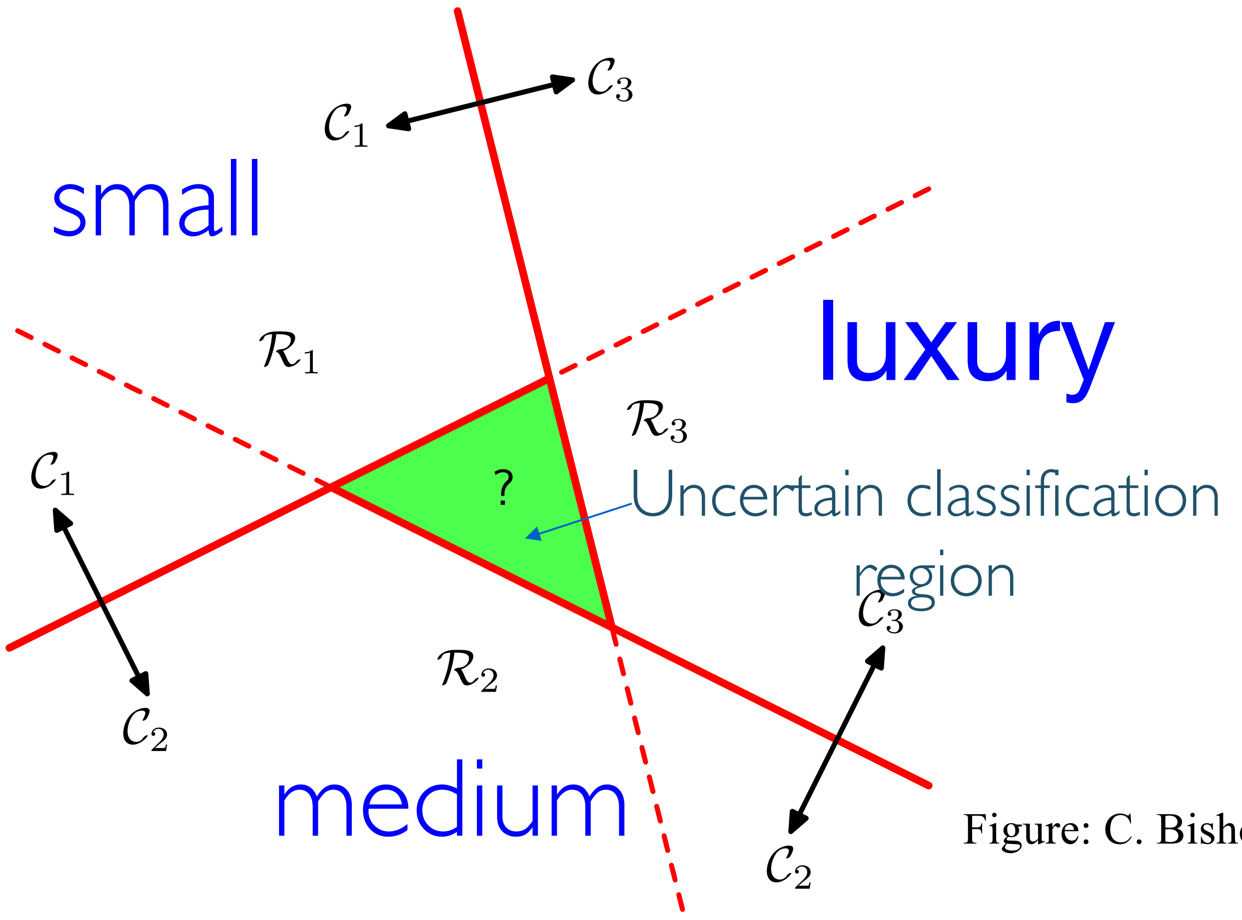


Figure: C. Bishop

Using Least Squares for Multiple Classes (Solution)

Encoding the K Classes

$$\mathbf{I}_K = \begin{array}{c} \begin{array}{ccccc} & & i_3 & & \end{array} \\ \underbrace{\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}}_{i_1, i_2, \ldots, i_K} \end{array} \left.\vphantom{\begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}\right\} K$$

- We start by encoding the classes as a **one-hot** binary coding scheme
- Class 3 is encoded as i_3 , the 3rd column of identity matrix I_K

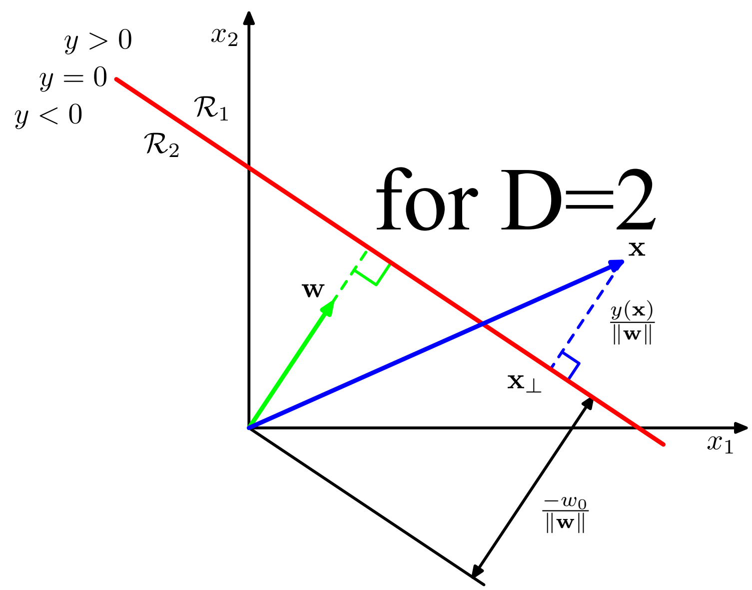
Linear Function

- Assign item with features $\mathbf{x}$ to class k if $y_k(\mathbf{x}) > y_j(\mathbf{x}),\ \forall \mathrm{j} \neq \mathrm{k}$

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

- The decision boundary is a (D-1) dimensional hyperplane

$$(\mathbf{w}_k - \mathbf{w}_j)^{\mathrm{T}}\mathbf{x} + (w_{k0} - w_{j0}) = 0$$



Least Squares Solution in Matrix Form

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}}$$

where

$$\widetilde{\mathbf{W}} = \begin{bmatrix} w_{10} & \cdots & w_{k0} & \cdots & w_{K0} \\ \mathbf{w}_1^T & \cdots & \mathbf{w}_k^T & \cdots & \mathbf{w}_K^T \end{bmatrix}$$

and $\widetilde{\mathbf{x}} = (1, \mathbf{x}^T)^T$

The least squares solution is

Pseudoinverse

$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^{\mathrm{T}}\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^{\mathrm{T}}\mathbf{T} = \widetilde{\mathbf{X}}^{\dagger}\mathbf{T}$$

where

$$\widetilde{\mathbf{X}} = \begin{bmatrix} 1 & \cdots & 1 \\ \mathbf{x}_1^T & \cdots & \mathbf{x}_n^T \end{bmatrix}$$

and

$$\mathbf{T} = \begin{bmatrix} i_{y_1}, \cdots, i_{y_n} \end{bmatrix}$$

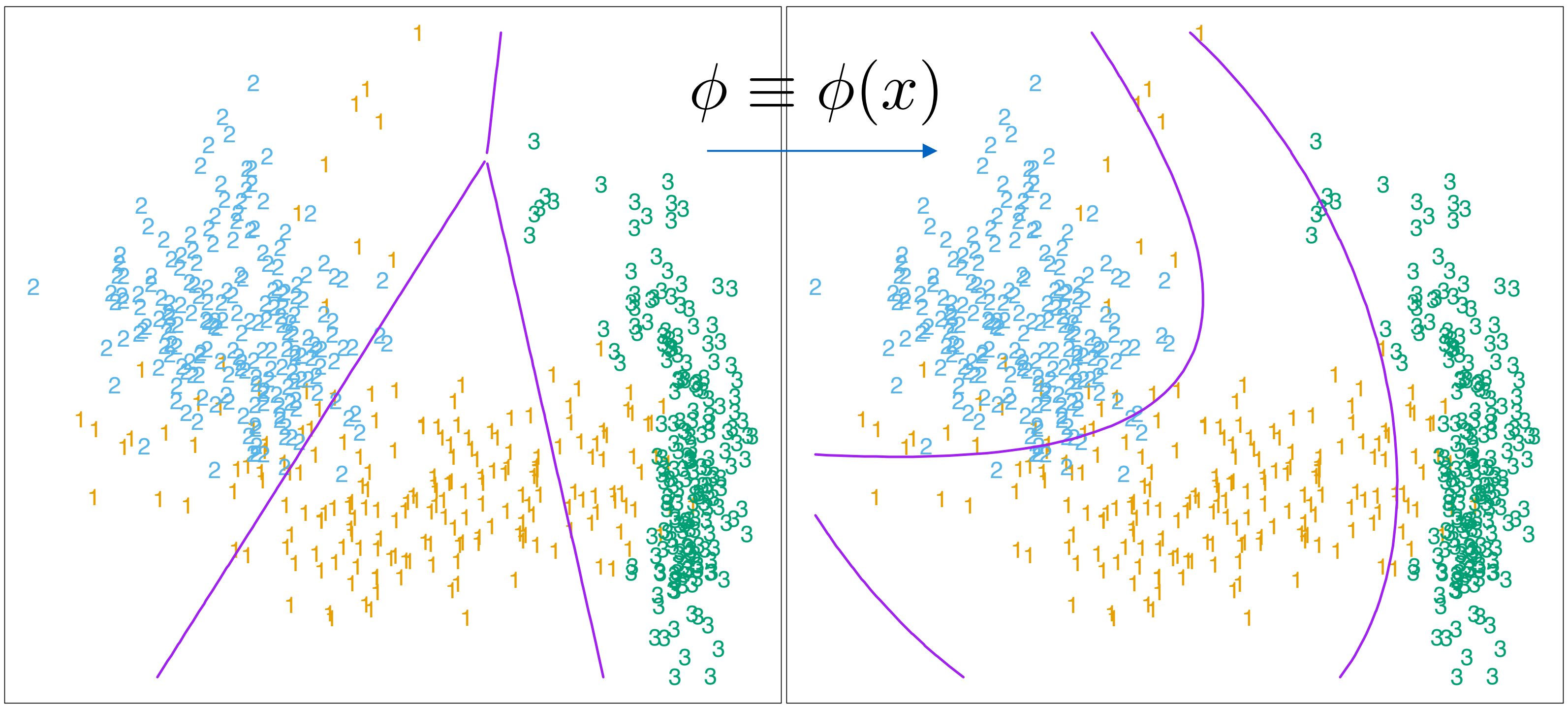
One hot encoding of class $\mathbf{y}_1$

Working with non-linearly separable data…

Linear Methods → Linear Boundaries?

Data from three classes, with linear decision boundaries

Linear boundaries in the five-dimensional space $\phi(\mathbf{x})=(X_1,X_2,X_1X_2,X_1^2,X_2^2)$



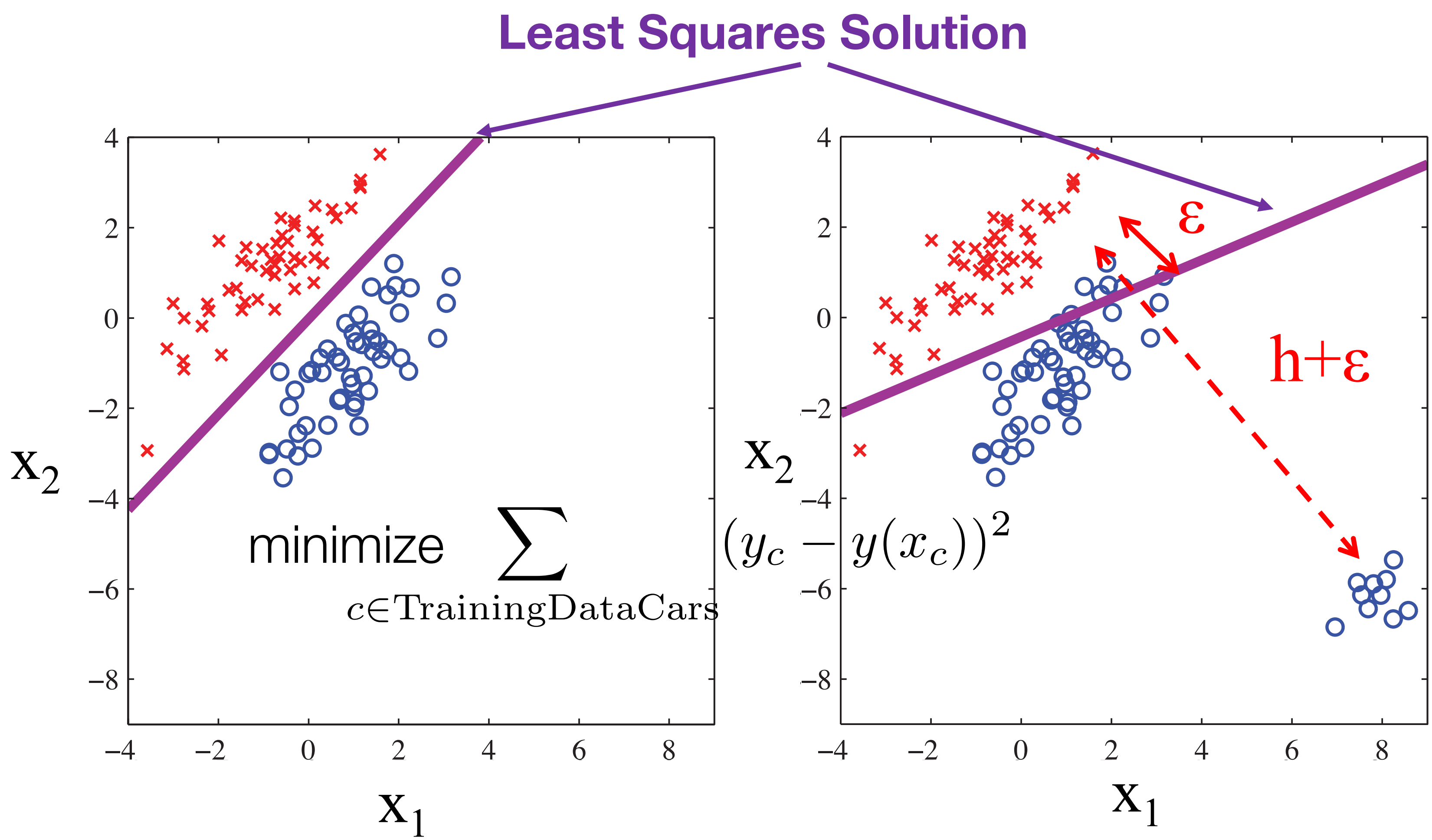
Linear inequalities in the transformed space are quadratic inequalities in the original space.

Tips

- Helps if $\phi_n = \phi(x_n)$ is binary, quantized or normalized
  - Example: if 1st feature $x_n(1)$ is age of user n, then
    - $\phi_n(1)$ could be indicator if $x_n(1)$ belongs to 1st quantile
    - $\phi_n(2)$ indicator whether $x_n(1)$ belongs to 2nd quantile
    - …

- Useful to add interaction terms $\phi_h = \phi(x_n) \circ \phi(x_m)$, where "$\circ$" is the Hadamard product (or element-wise product)
  - XOR operator can be better than Hadamard product for binary variables.

Issues with Least Squares Classification



$$\text{minimize} \sum_{c \in \text{TrainingDataCars}} (y_c - y(x_c))^2$$

With square loss (score), optimization cares too much about reducing distance to boundary of well separable items…

…solution is to change the score function

Logistic Regression (for Classification)

- Back to two classes, C_1 and C_2
- Logistic regression is often used for two classes

$$p(\mathcal{C}_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}\right)$$

where

$$\sigma(a) = \frac{\exp(a)}{1+\exp(a)}$$

Logistic function

and $\phi \equiv \phi(x)$

Transformation of feature vector (possibly non-linear)

- But can be easily generalized to K classes

Finding Logistic Regression Parameters (binary classification)

- For a dataset $\{\phi_c, t_c\}$ where ϕ_c is transformed feature vector of car c, $t_c \in \{0,1\}$ is the class of car $c \in \text{TrainingDataCars}$
- The likelihood function over the training data is

$$p(\mathbf{t}|\mathbf{w}) = \prod_{c \in \text{TrainingDataCars}} y(\phi_c)^{t_c} (1 - y(\phi_c))^{1-t_c}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$, $y(\phi_n) = \sigma(\mathbf{w}^T \phi_n)$

- The log of the likelihood function (log-likelihood) is

$$\log p(\mathbf{t}|\mathbf{w}) = \sum_{c \in \text{TrainingDataCars}} t_c \log y(\phi_c) + (1 - t_c) \log(1 - y(\phi_c))$$

- To solve the above equation, we maximize the log-likelihood over parameters $\mathbf{w}$
- The above equation is also described as *cross-entropy loss, logistic loss, log loss,* on Tensorflow, Sklearns, and pyTorch.

Solving Logistic Regression via Maximum Likelihood

- The above equations give the following gradient

$$\nabla_{\mathbf{w}} \log p(\mathbf{t}|\mathbf{w}) = \sum_{c \in \mathrm{TrainingDataCars}} (t_c - y(\phi_c))\phi_c = \mathbf{\Phi}^T(\mathbf{t} - \mathbf{y})$$

Exercise: what are the dimensions of $\mathbf{\Phi}$?

- The logistic function has derivative

$$\frac{d}{da}\sigma(a) = \sigma(a)(1 - \sigma(a))$$

- The second derivative (Hessian) is then (verify!)

$$\mathbf{H} = \mathbf{\Phi}^T \mathbf{R} \mathbf{\Phi}$$

where $R_c = y(\phi_c)(1 - y(\phi_c))$ and $\mathbf{H}$ is a positive definite matrix. Because $\mathbf{H}$ is positive definite the optimization is concave on $\mathbf{w}$ and has a unique maximum.

Iterative Update

- The iterative parameter update is (Newton-Raphson)

$$\begin{aligned} \mathbf{w}^{(\text{new})} &= \mathbf{w}^{(\text{old})} - (\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{R}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}(\mathbf{y}-\mathbf{t}) \\ &= (\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{R}\boldsymbol{\Phi})^{-1}\left\{\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{R}\boldsymbol{\Phi}\mathbf{w}^{(\text{old})} - \boldsymbol{\Phi}^{\mathrm{T}}(\mathbf{y}-\mathbf{t})\right\} \\ &= (\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{R}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{z} \end{aligned}$$

where $\mathbf{z}$ is an N-dimensional vector with elements

$$\mathbf{z} = \boldsymbol{\Phi}\mathbf{w}^{(\text{old})} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})$$

Training Logistic Regression with Data Selection Bias

- Let π_i be the probability of sampling example i in the training data
- We say a data sample is biased when π_i is **not** uniform
- Correcting for bias in the likelihood function:

$$\log p(\mathbf{t}|\mathbf{w}) = \sum_{c \in \text{TrainingDataCars}} \frac{1}{\pi_c} \left(t_c \log y(\phi_c) + (1 - t_c) \log(1 - y(\phi_c))\right)$$

  where C_c is the class of car *c*.

- Generally, it is a good idea to test training data for different weights
  - The weights can be used to protect against sampling designs which could cause selection bias.
  - The weights can be used to protect against misspecification of the model.
- Unbalanced datasets:
  - Suppose there are more males than females in dataset.
    What will happen to decision boundary? How to fix it?

Multiclass Logistic Regression (MLR)

- Consider K classes and N observations
- Let C_i be the class of the i-th example with feature vector ϕ_i

$$P(C_c = t_c|\phi_c) = \frac{\exp(\mathbf{w}_k^T\phi_c)}{\sum_{h=1}^K \exp(\mathbf{w}_h^T\phi_c)}$$

a.k.a. **softmax**

- If we assume one-hot encoding of target variable t_c, the log-likelihood function is

$$\sum_{c\in\text{TrainingDataCars}} \sum_{k=1}^{K} t_{c,k} \log \frac{\exp(\mathbf{w}_k^T\phi_c)}{\sum_{h=1}^K \exp(\mathbf{w}_h^T\phi_c)}$$

$$= \sum_{c\in\text{TrainingDataCars}} \sum_{k=1}^{K} t_{c,k}\mathbf{w}_k^T\phi_c - \sum_{c\in\text{TrainingDataCars}} \log \sum_{h=1}^{K} \exp(\mathbf{w}_h^T\phi_c)$$