Supervised Learning

Regression and Classification

Shan-Hung Wu
shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

NetDB-ML, Spring 2015

Outline

Outline

The Regression Problem

- Given
  - a ***training dataset*** $\mathcal{X} = \left\{ \left(\boldsymbol{x}^{(t)}, r^{(t)} \right) \right\}_{t=1}^{N}$, where $\boldsymbol{x}^{(t)} \in \mathbb{R}^d$'s are ***examples*** (or ***instances*** or ***observations***) consisting of ***attributes*** (or ***inputs*** or ***features***) and $r^{(t)} \in \mathbb{R}$'s are ***labels***, and
  - a ***testing instance*** $\boldsymbol{x}'$,

  predict the label y' of $\boldsymbol{x}'$
- Example: stock price forecasting

ML Process Revisited

1. Data collection and preprocessing (e.g., integration, cleaning, etc.)
2. Model development
   1. Assume a ***model*** that represents the posteriori knowledge we want to discover. The model has parameters
   2. Define an ***objective*** that measures "how good the model with a particular combination of parameters can explain the data"
3. ***Training***: employ an algorithm that optimizes the objective by finding the best (or good enough) parameters
4. ***Testing***: evaluate the model performance on hold-out data
5. Using the model

Modeling a Regressor

- Model: Let the model be a collection of functions, called ***hypothesis class*** and denoted as $\mathcal{H} = \{h : \mathcal{I} \times \Theta \to \mathbb{R}\}$, where $\mathcal{I}$ is the ***input space*** (or ***feature space***) and Θ is the set of all possible parameters
  - A particular $\theta \in \Theta$ instantiates a ***hypothesis*** h that makes the ***prediction*** (or ***output***) $y' = h(\boldsymbol{x}'; \theta) > 0$
- Objective: $\arg_\theta \min \sum_{t=1}^{N} l(h(\boldsymbol{x}^{(t)}; \theta), r^{(t)})$, where l is some ***loss function*** which penalizes the error of predictions made on the training dataset
  - We want the hypothesis to have the minimal ***empirical error***: $emp(\theta; \mathcal{X}) = \sum_{t=1}^{N} l(h(\boldsymbol{x}^{(t)}; \theta), r^{(t)})$

The Objective

- Common choice: $\arg_\theta \min \sum_{t=1}^{N} \left[r^{(t)} - h(\boldsymbol{x}^{(t)}; \theta) \right]^2$
  - $emp(\theta; \mathcal{X}) = \sum_{t=1}^{N} \left[r^{(t)} - h(\boldsymbol{x}^{(t)}; \theta) \right]^2$ has a specific name called the ***Sum of Square Errors (SSE)***
- Alternatively, the objective can be formed using the absolute error: $\arg_\theta \min \sum_{t=1}^{N} \left| r^{(t)} - h(\boldsymbol{x}^{(t)}; \theta) \right|$
  - What is the difference? [Homework]

Outline

Linear Regression

- Suppose $\boldsymbol{x}$ is a scalar and h is a line, i.e., $h(x;\theta) = w_1 x + w_0$, we have the objective:
  - To find w_0 and w_1 that minimizes
    $$emp(\theta;\mathcal{X}) = \sum_{t=1}^{N} \left(r^{(t)} - \boldsymbol{w}^\top \begin{bmatrix} 1 \\ x^{(t)} \end{bmatrix} \right)^2,$$
  - where $\boldsymbol{w} = [w_0, w_1]^\top \in \mathbb{R}^2$

Training: Analytic Solution (1)

- We take the partial derivatives of emp with respect to w_0 and w_1 and set them to 0
  - We have a system of linear equations

$$\begin{cases} \sum_{t=1}^{N} r^{(t)} = Nw_0 + w_1 \sum_{t=1}^{N} x^{(t)} \\ \sum_{t=1}^{N} x^{(t)} r^{(t)} = w_0 \sum_{t=1}^{N} x^{(t)} + w_1 \sum_{t=1}^{N} \left(x^{(t)}\right)^2 \end{cases}$$

  - Let $\boldsymbol{A} = \begin{bmatrix} N & \sum_{t=1}^{N} x^{(t)} \\ \sum_{t=1}^{N} x^{(t)} & \sum_{t=1}^{N} \left(x^{(t)}\right)^2 \end{bmatrix}$, $\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$, and $\boldsymbol{y} = \begin{bmatrix} \sum_{t=1}^{N} r^{(t)} \\ \sum_{t=1}^{N} r^{(t)} x^{(t)} \end{bmatrix}$, we can solve $\boldsymbol{w}$ by $\boldsymbol{w} = \boldsymbol{A}^{-1}\boldsymbol{y}$

Training: Analytic Solution (2)

- A bit of arithmetic leads to

$$\begin{cases} w_0 = \overline{r} - w_1 \overline{x} \\ w_1 = \left(\sum_{t=1}^{N} x^{(t)} r^{(t)} - \overline{x} \overline{r} N \right) / \left(\sum_{t=1}^{N} \left(x^{(t)} \right)^2 - N \overline{x}^2 \right) \end{cases},$$

where $\overline{x} = \frac{1}{N} \sum_{t=1}^{N} x^{(t)}$ and $\overline{r} = \frac{1}{N} \sum_{t=1}^{N} r^{(t)}$ [Proof]

Multivariate Linear Regression

- Given $\boldsymbol{x} \in \mathbb{R}^d$, suppose h is linear: $h(\boldsymbol{x};\theta) = \boldsymbol{w}^\top \begin{bmatrix} 1 \\ \boldsymbol{x}^{(t)} \end{bmatrix}$, where $\boldsymbol{w} = [w_0, w_1, \cdots, w_d]^\top \in \mathbb{R}^{d+1}$
- We can solve $\boldsymbol{w}$ by $\boldsymbol{w} = \boldsymbol{A}^{-1}\boldsymbol{y}$, where $\boldsymbol{y} = \begin{bmatrix} \sum_{t=1}^N r^{(t)} \\ \sum_{t=1}^N r^{(t)} x_1^{(t)} \\ \vdots \\ \sum_{t=1}^N r^{(t)} x_d^{(t)} \end{bmatrix}$ and

$$\boldsymbol{A} = \begin{bmatrix} N & \sum_{t=1}^N x_1^{(t)} & \cdots & \sum_{t=1}^N x_d^{(t)} \\ \sum_{t=1}^N x_1^{(t)} & \sum_{t=1}^N x_1^{(t)2} & \cdots & \sum_{t=1}^N x_1^{(t)} x_d^{(t)} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{t=1}^N x_d^{(t)} & \sum_{t=1}^N x_d^{(t)} x_1^{(t)} & \cdots & \sum_{t=1}^N x_d^{(t)2} \end{bmatrix}$$

[Proof]

From Least Squares to Linear Regression

- Let $\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}$, $\boldsymbol{w} = [w_0, w_1, \cdots, w_d]^\top$, and $\boldsymbol{r} = [r^{(1)}, r^{(2)}, \cdots, r^{(N)}]^\top$.
- Ideally, we want to solve $\boldsymbol{w}$ such that $\boldsymbol{X}\boldsymbol{w} = \boldsymbol{r}$, but impossible if $N > d$
- We can instead solve the "closet approximation:" $\arg\min_{\boldsymbol{w}} \|\boldsymbol{r} - \boldsymbol{X}\boldsymbol{w}\|^2$
  - $\|\boldsymbol{r} - \boldsymbol{X}\boldsymbol{w}\|^2$ is exactly the SSE!
- The ***least square problem***: find $\boldsymbol{w}$ such that $\|\boldsymbol{r} - \boldsymbol{X}\boldsymbol{w}\|^2$ is minimized. Solution?

From Least Squares to Linear Regression

- Let $\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}$, $\boldsymbol{w} = [w_0, w_1, \cdots, w_d]^\top$, and $\boldsymbol{r} = [r^{(1)}, r^{(2)}, \cdots, r^{(N)}]^\top$.
- Ideally, we want to solve $\boldsymbol{w}$ such that $\boldsymbol{X}\boldsymbol{w} = \boldsymbol{r}$, but impossible if $N > d$
- We can instead solve the "closet approximation:" $\arg\min_{\boldsymbol{w}} \|\boldsymbol{r} - \boldsymbol{X}\boldsymbol{w}\|^2$
  - $\|\boldsymbol{r} - \boldsymbol{X}\boldsymbol{w}\|^2$ is exactly the SSE!
- The ***least square problem***: find $\boldsymbol{w}$ such that $\|\boldsymbol{r} - \boldsymbol{X}\boldsymbol{w}\|^2$ is minimized. Solution?
  - $\boldsymbol{w}^* = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{r}$ if $\boldsymbol{X}$ is full column rank (remember the normal equations?)
  - $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$ and $\boldsymbol{X}^\top \boldsymbol{r}$ are exactly $\boldsymbol{A}^{-1}$ and $\boldsymbol{y}$ seen previously

Analytic Solution Revisited

- What if $\boldsymbol{X}$ is not full column rank?

Analytic Solution Revisited

- What if $\boldsymbol{X}$ is not full column rank?

1. Anyone in the set $\boldsymbol{X}^{\dagger}\boldsymbol{r}+\mathcal{N}(\boldsymbol{X})$ is the solution (remember the SVD solution to least squares?)
2. Make $\boldsymbol{X}$ full column rank by changing the objective (to be explained later)

Training: Numeric Methods

- Machine learning solutions need not be accurate
  - Close-to-optimal solutions enough for making good predictions
- Numeric methods suffice
  - E.g., gradient descent:

Repeat until convergence {

$$\boldsymbol{w} := \boldsymbol{w} - \eta \nabla emp(\boldsymbol{w}; \mathcal{X}) = \boldsymbol{w} + 2\eta \sum_{t=1}^{N} (r^{(t)} - \boldsymbol{w}^\top \begin{bmatrix} 1 \\ \boldsymbol{x}^{(t)} \end{bmatrix}) \begin{bmatrix} 1 \\ \boldsymbol{x}^{(t)} \end{bmatrix};$$

}

- The step size η is called the ***learning rate***

Outline

Interpolation vs. Regression (1)

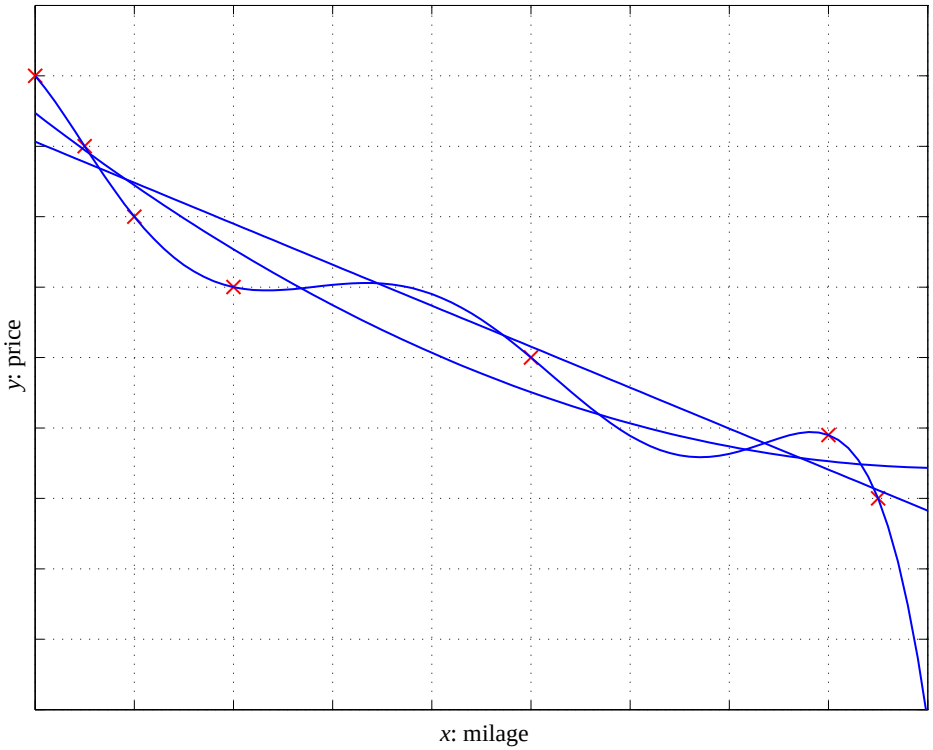
- Instead of regression, we can perform the ***interpolation*** that fits a hypothesis $h : \mathbb{R} \times \Theta \to \mathbb{R}$ to examples, i.e., $h(x^{(t)}; \theta) = r^{(t)}$
  - In polynomial interpolation, we can always fit a polynomial of degree $(N-1)$ to N 1-D points
  - Let $\theta = (w_0, \cdots, w_{N-1})$ and $h(x; \theta) = w_0 + w_1 x + \cdots + w_{N-1} x^{N-1}$, $x \in \mathbb{R}$
  - Obtain θ by solving

$$\begin{bmatrix} \left(x^{(1)}\right)^0 & \cdots & \left(x^{(1)}\right)^{N-1} \\ \vdots & \ddots & \vdots \\ \left(x^{(N)}\right)^0 & \cdots & \left(x^{(N)}\right)^{N-1} \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_{N-1} \end{bmatrix} = \begin{bmatrix} r^{(1)} \\ \vdots \\ r^{(N)} \end{bmatrix}$$

- The label of a new instance x' can be predicted by $y' = h(x'; \theta)$

Interpolation vs. Regression (2)

- Given 7 examples, the right shows the regression results using polynomials of degrees 1, 2, and 6
  - $x^{(t)}$ is the mileage of a used car and $r^{(t)}$ is the price
- It is unlikely that the real curve shapes like the 6th-order polynomial



Interpolation vs. Regression (3)

- In the presence of noise, we don't need an exact fitting
- The target of regression is to catch the trend
  - Differs from interpolation in finding a "simple" hypothesis (e.g., low degree polynomial) that is "close enough" to the examples

How About Nonlinear Trend/Regression? (1)

- In the case of univariate regression (where x's are scalars), we can assume a polynomial hypothesis with an arbitrary degree k: $h(x;\theta) = w_0 + w_1 x + \cdots + w_k x^k$,
- We can solve $\boldsymbol{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_k \end{bmatrix}$ by $\boldsymbol{w} = \boldsymbol{A}^{-1}\boldsymbol{y}$, where

$$\boldsymbol{A} = \begin{bmatrix} N & \sum_{t=1}^{N} x^{(t)} & \cdots & \sum_{t=1}^{N} x^{(t)k} \\ \sum_{t=1}^{N} x^{(t)} & \sum_{t=1}^{N} x^{(t)2} & \cdots & \sum_{t=1}^{N} x^{(t)(k+1)} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{t=1}^{N} x^{(t)k} & \sum_{t=1}^{N} x^{(t)(k+1)} & \cdots & \sum_{t=1}^{N} x^{(t)2k} \end{bmatrix},$$

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_k \end{bmatrix}, \text{ and } \boldsymbol{y} = \begin{bmatrix} \sum_{t=1}^{N} r^{(t)} \\ \sum_{t=1}^{N} r^{(t)} x^{(t)} \\ \vdots \\ \sum_{t=1}^{N} r^{(t)} x^{(t)k} \end{bmatrix}$$ [Proof]

How About Nonlinear Trend/Regression? (2)

- In multivariate regression, we seldom assume h to be a polynomial with degree higher than 1
  - Why?

How About Nonlinear Trend/Regression? (2)

- In multivariate regression, we seldom assume h to be a polynomial with degree higher than 1
  - Why?

1. Analytical simplicity
2. More descriptive model:
   - The sign of w_j tells whether x_j has positive or negative effect on the prediction
   - The absolute value of w_j indicates how important the feature is (provided that features are in the same range); if w_j is close to 0, the feature can even be removed
3. We can instead augment the inputs to achieve the effect of nonlinear regression (to be explained later)

Outline

Probability Interpretation (1)

- Given $\mathcal{X} = \{\boldsymbol{x}^{(t)}, r^{(t)}\}_{t=1}^{N}$, where $r^{(t)} \in \mathbb{R}$. Assume
  - $(\boldsymbol{x}^{(t)}, r^{(t)})$ are i.i.d samples drawn from some joint distribution of $\boldsymbol{x}$ and r (otherwise can never learn r from $\boldsymbol{x}$)
  - In particular, $r^{(t)} = f(\boldsymbol{x}^{(t)}; \theta) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \beta^{-1})$ for some ***hyperparameter*** (i.e., constant fixed during the objective solving) β
  - The marginal distribution $p(r|\boldsymbol{x})$ follows: $p(r|\boldsymbol{x}) = p_{N_{h(\boldsymbol{x};\theta),\beta^{-1}}}(r)$
- We want to estimate f using $\mathcal{X}$
  - Hypothesis: $h(\boldsymbol{x}; w_0, w_1, \cdots, w_d) = w_0 + w_1 x_1 + \cdots + w_d x_d$, a line
  - Once getting $w_0, w_1, \cdots, w_d$, we can predict the unknown r' of a new instance $\boldsymbol{x}'$ by $y' = \arg_y \max p(y|\boldsymbol{x}') = \arg_y \max p_{N_{h(\boldsymbol{x}';\theta),\beta^{-1}}}(y) = h(\boldsymbol{x}'; \theta)$
  - Note that we don't need to know β to make prediction

Probability Interpretation (2)

- How to obtain the estimate h of f? How to obtain θ?
- Now let θ be a random variable, we can pick θ maximizing $p(\theta|\mathcal{X})$, the ***posterior*** probability
- Or, by Baye's theorem, θ maximizing the ***likelihood*** $p(\mathcal{X}|\theta)$ (if we assume $p(\theta)$ remains the same for all θ)
- Or, θ maximizing the ***log likelihood*** $\log p(\mathcal{X}|\theta) = \log\left(\prod_{t=1}^{N} p(\boldsymbol{x}^{(t)}, r^{(t)}|\theta)\right) = \log\left(\prod_{t=1}^{N} p(r^{(t)}|\boldsymbol{x}^{(t)}, \theta)p(\boldsymbol{x}^{(t)}|\theta)\right) = \log\left(\prod_{t=1}^{N} p(h(\boldsymbol{x}^{(t)};\theta)+\epsilon|\boldsymbol{x}^{(t)}, \theta)p(\boldsymbol{x}^{(t)}|\theta)\right)$
- Ignoring $p(\boldsymbol{x}^{(t)}|\theta) = p(\boldsymbol{x}^{(t)})$ (since it is irrelevant to θ) and constants we have $\log p(\mathcal{X}|\theta) \propto -N\log\left(\sqrt{\frac{2\pi}{\beta}}\right) - \frac{\beta}{2}\sum_{t=1}^{N}\left(r^{(t)} - h(\boldsymbol{x}^{(t)};\theta)\right)^2$
- Dropping the first term and constants we have $\log p(\mathcal{X}|\theta) \propto -\sum_{t=1}^{N}\left(r^{(t)} - h(\boldsymbol{x}^{(t)};\theta)\right)^2$; that is, we seek for θ minimizing the SSE (sum of square errors)!

Outline

Two-Class Classification Problem

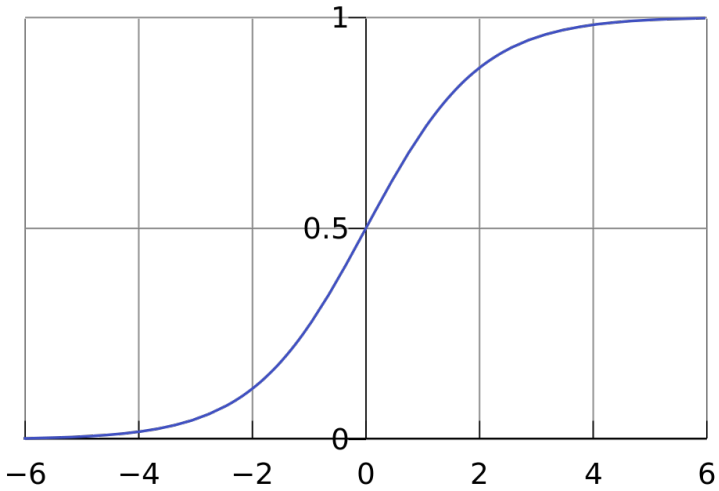
- Given a ***training dataset*** $\mathcal{X} = \left\{ \left(\boldsymbol{x}^{(t)}, r^{(t)} \right) \right\}_{t=1}^{N}$, where $r^{(t)} \in \{1, -1\}$, and a testing instance $\boldsymbol{x}'$, predict the label of $\boldsymbol{x}'$
- Model (or ***hypothesis class***): $\mathcal{H} = \{h : \mathcal{I} \times \Theta \rightarrow \{1, -1\}\}$
  - Or $\mathcal{H} = \{h : \mathcal{I} \times \Theta \rightarrow \mathbb{R}\}$ with prediction $\text{sgn}(h(\boldsymbol{x}'; \theta))$
- Objective: $\arg_\theta \min \sum_{t=1}^{N} l(h(\boldsymbol{x}^{(t)}; \theta), r^{(t)})$ with some loss function l
  - Example: the ***0-1 loss function***: $l(a, b) = 1$ if $a \neq b$; 0 otherwise

Outline

Logistic Function

- The ***logistic function*** (a special case of ***sigmoid functions***) is defined as

$$g(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$



- Always gives values between $(0, 1)$
  - The larger the z, the higher the $g(z)$
  - The smaller the z, the higher the $1 - g(z)$

Logistic Regression

- In regression, we learn $p(r|\boldsymbol{x};\theta)$ from $\mathcal{X}$ and make predictions by $y' = \arg\max_y p(y|\boldsymbol{x}';\theta)$
- In ***logistic regression*** everything is the same except that $P(r|\boldsymbol{x};\theta)$ is modeled by a Bernoulli distribution parametrized by ϕ:

$$P(r|\boldsymbol{x};\theta) = \begin{cases} \phi, & \text{if } r = 1, \\ 1-\phi, & \text{otherwise,} \end{cases}$$

  - We can simply write $P(r|\boldsymbol{x};\theta) = \phi^q(1-\phi)^{(1-q)}$, where $q = \frac{r+1}{2}$
- Furthermore, $\phi = \pi(\boldsymbol{x};\boldsymbol{\beta}) = \frac{e^{\boldsymbol{\beta}^\top \widetilde{\boldsymbol{x}}}}{e^{\boldsymbol{\beta}^\top \widetilde{\boldsymbol{x}}}+1} = \frac{1}{1+e^{-\boldsymbol{\beta}^\top \widetilde{\boldsymbol{x}}}}$ is a deterministic function, where $\widetilde{\boldsymbol{x}} = [1, \boldsymbol{x}]^\top$
  - So the larger the projection of $\widetilde{\boldsymbol{x}}$ onto a line, the higher the ϕ
- Prediction: $y' = \arg\max_y p(y|\boldsymbol{x}';\theta) = \arg\max_y \{\phi, 1-\phi\} = \operatorname{sgn}(\boldsymbol{\beta}^\top \widetilde{\boldsymbol{x}}') = \operatorname{sgn}(\boldsymbol{w}^\top \boldsymbol{x}' + b)$

Fitting Logistic Regression Models (1)

- How to obtain β?

Fitting Logistic Regression Models (1)

- How to obtain $\boldsymbol{\beta}$?
  - By $\boldsymbol{\beta}$ maximizing $p(\boldsymbol{\beta}|\mathcal{X})$
  - Or, by Bayes' Rule and assuming uniform $p(\boldsymbol{\beta})$, $\boldsymbol{\beta}$ maximizing $p(\mathcal{X}|\boldsymbol{\beta})$
- Log-likelihood:

$$\begin{aligned} l(\boldsymbol{\beta}) &= \log \prod_{t=1}^{N} p\left(\boldsymbol{x}^{(t)}, r^{(t)}|\boldsymbol{\beta}\right) \\ &= \log \prod_{t=1}^{N} P\left(r^{(t)}|\boldsymbol{x}^{(t)}, \boldsymbol{\beta}\right) p\left(\boldsymbol{x}^{(t)}|\boldsymbol{\beta}\right) \\ &\propto \log \prod_{t=1}^{N} \pi\left(\boldsymbol{x}^{(t)}; \boldsymbol{\beta}\right)^{q^{(t)}} \left(1-\pi\left(\boldsymbol{x}^{(t)}; \boldsymbol{\beta}\right)\right)^{\left(1-q^{(t)}\right)} \end{aligned}$$

  - $p\left(\boldsymbol{x}^{(t)}|\boldsymbol{\beta}\right) = p\left(\boldsymbol{x}^{(t)}\right)$ can be dropped
- $l(\boldsymbol{\beta}) = \sum_{t=1}^{N} \left\{q^{(t)} \log \pi\left(\boldsymbol{x}^{(t)}; \boldsymbol{\beta}\right) + \left(1-q^{(t)}\right) \log\left(1-\pi\left(\boldsymbol{x}^{(t)}; \boldsymbol{\beta}\right)\right)\right\} = \sum_{t=1}^{N} \left\{q^{(t)} \boldsymbol{\beta}^{\top} \widetilde{\boldsymbol{x}}^{(t)} - \log\left(1+e^{\boldsymbol{\beta}^{\top} \widetilde{\boldsymbol{x}}^{(t)}}\right)\right\}$ [Homework]

Fitting Logistic Regression Models (2)

- To maximize the log-likelihood, we set its derivative to zero:

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{t=1}^{N} \widetilde{\boldsymbol{x}}^{(t)\top} \left(q^{(t)} - \pi\left(\boldsymbol{x}^{(t)}; \boldsymbol{\beta} \right) \right) = \mathbf{0}^{\top}$$

- Unlike the linear regression, we cannot solve $\boldsymbol{\beta}$ analytically in a closed-form
- How to obtain $\boldsymbol{\beta}$?

Fitting Logistic Regression Models (2)

- To maximize the log-likelihood, we set its derivative to zero:

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{t=1}^{N} \tilde{\boldsymbol{x}}^{(t)\top} \left(q^{(t)} - \pi\left(\boldsymbol{x}^{(t)}; \boldsymbol{\beta} \right) \right) = \mathbf{0}^{\top}$$

- Unlike the linear regression, we cannot solve $\boldsymbol{\beta}$ analytically in a closed-form
- How to obtain $\boldsymbol{\beta}$? Iterative algorithms
- Gradient descent:

Repeat until convergence {

$\boldsymbol{\beta} := \boldsymbol{\beta} + \eta \nabla l(\boldsymbol{\beta}) = \boldsymbol{\beta} + \eta \sum_{t=1}^{N} \tilde{\boldsymbol{x}}^{(t)\top} \left(q^{(t)} - \pi\left(\boldsymbol{x}^{(t)}; \boldsymbol{\beta} \right) \right)$;

}

- Observe that $l(\boldsymbol{\beta})$ is concave [Homework]
  - So iterative algorithms approach to global optimal

Newton's Method for Logistic Regression*

Update rule: $\boldsymbol{\beta} := \boldsymbol{\beta} - \left(\nabla^2 -l(\boldsymbol{\beta})\right)^{-1} \nabla -l(\boldsymbol{\beta}) = \boldsymbol{\beta} - \left(\nabla^2 l(\boldsymbol{\beta})\right)^{-1} \nabla l(\boldsymbol{\beta})$

- Given $\boldsymbol{q} \in \mathbb{R}^N$ the vector of $q^{(t)}$'s, $\boldsymbol{X} \in \mathbb{R}^{N \times (d+1)}$ the row matrix of $\widetilde{\boldsymbol{x}}^{(t)}$'s, $\boldsymbol{\pi} \in \mathbb{R}^N$ with the tth element $\pi\left(\boldsymbol{x}^{(t)}; \boldsymbol{\beta}\right)$, and $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ a diagonal matrix with the tth diagonal element $\pi\left(\boldsymbol{x}^{(t)}; \boldsymbol{\beta}\right)\left(1 - \pi\left(\boldsymbol{x}^{(t)}; \boldsymbol{\beta}\right)\right)$, then

$$\nabla l(\boldsymbol{\beta}) = \left(\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right)^\top = \boldsymbol{X}^\top (\boldsymbol{q} - \boldsymbol{\pi}),$$

$$\nabla^2 l(\boldsymbol{\beta}) = \left(\frac{\partial \nabla l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\right)^\top = \left(\frac{\partial \sum_{t=1}^N \widetilde{\boldsymbol{x}}^{(t)} \left(q^{(t)} - \pi\left(\boldsymbol{x}^{(t)}; \boldsymbol{\beta}\right)\right)}{\partial \boldsymbol{\beta}}\right)^\top$$

$$= \left(-\sum_{t=1}^N \widetilde{\boldsymbol{x}}^{(t)} \pi\left(\boldsymbol{x}^{(t)}; \boldsymbol{\beta}\right)\left(1 - \pi\left(\boldsymbol{x}^{(t)}; \boldsymbol{\beta}\right)\right) \boldsymbol{x}^{(t)\top}\right)^\top = -\boldsymbol{X}^\top \boldsymbol{W} \boldsymbol{X}$$

  - Note that $g'(z) = \frac{1}{(1+e^{-z})^2} e^{-z} = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right) = g(z)(1 - g(z))$.

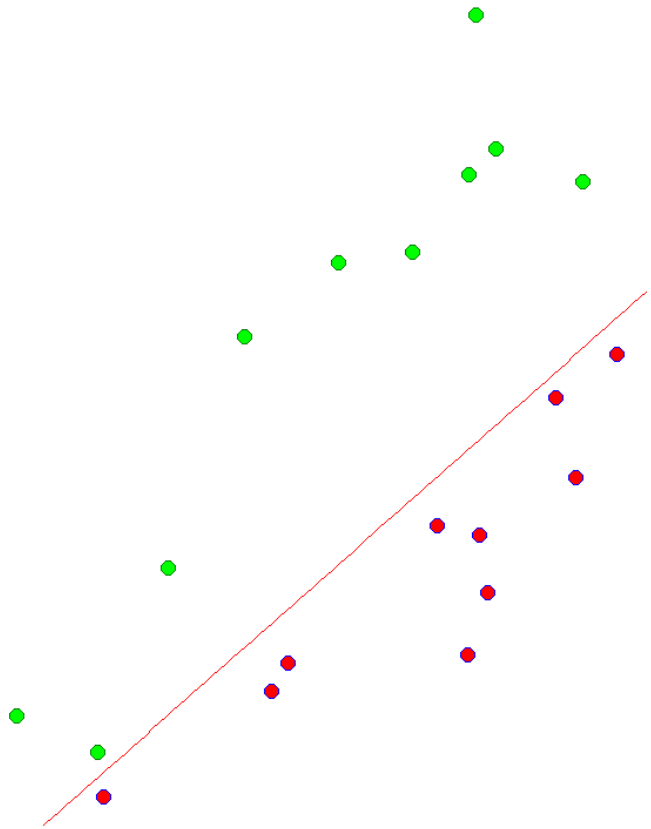
Outline

Perceptron (1)

- Recall that in logistic regression, we make prediction by $y' = \arg\max_y p(y|\boldsymbol{x}';\theta) = \arg\max_y \{\phi, 1-\phi\} = \text{sgn}(\boldsymbol{\beta}^\top \widetilde{\boldsymbol{x}'}) = \text{sgn}(\boldsymbol{w}^\top \boldsymbol{x}' + b)$
- Why not just making prediction based on $\text{sgn}(\boldsymbol{w}^\top \boldsymbol{x}' + b)$ directly?

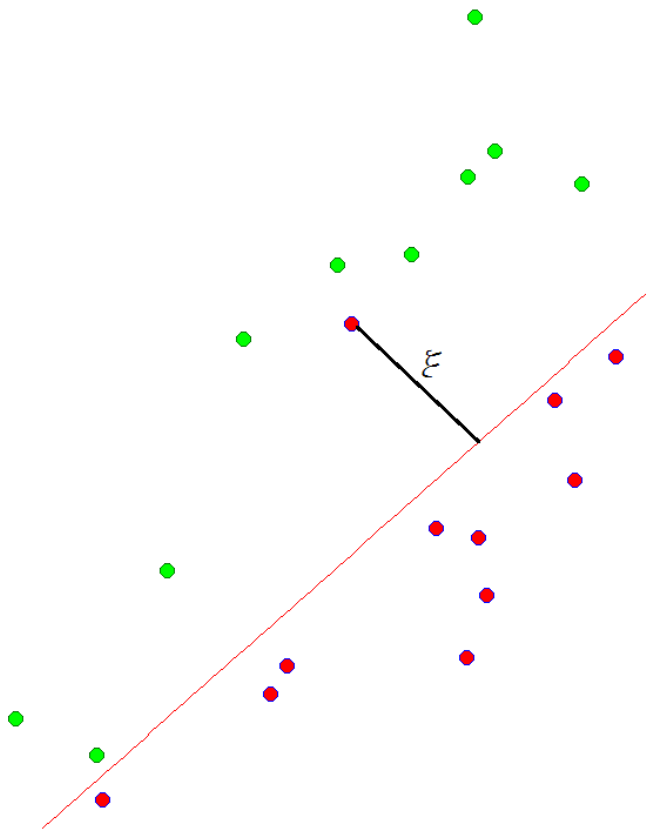
Perceptron (2)

- Model: $\mathcal{H} = \{f : f : \mathbb{R}^d \to \mathbb{R}, f(\boldsymbol{x}; \theta) = \boldsymbol{w}^\top \boldsymbol{x} + b\}$
  $\Theta = \{\boldsymbol{w}, b : \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$
  - A collection of hyperplanes
- Prediction: $y' = \text{sgn}(f(\boldsymbol{x}'))$
- Objective 1: any $f \in \mathcal{H}$ such that
  - $\boldsymbol{w}^\top \boldsymbol{x}^{(t)} + b > 0$ if $r^{(t)} = 1$;
    $\boldsymbol{w}^\top \boldsymbol{x}^{(t)} + b < 0$ otherwise
  - or simply $r^{(t)}(\boldsymbol{w}^\top \boldsymbol{x}^{(t)} + b) > 0$, $\forall t$



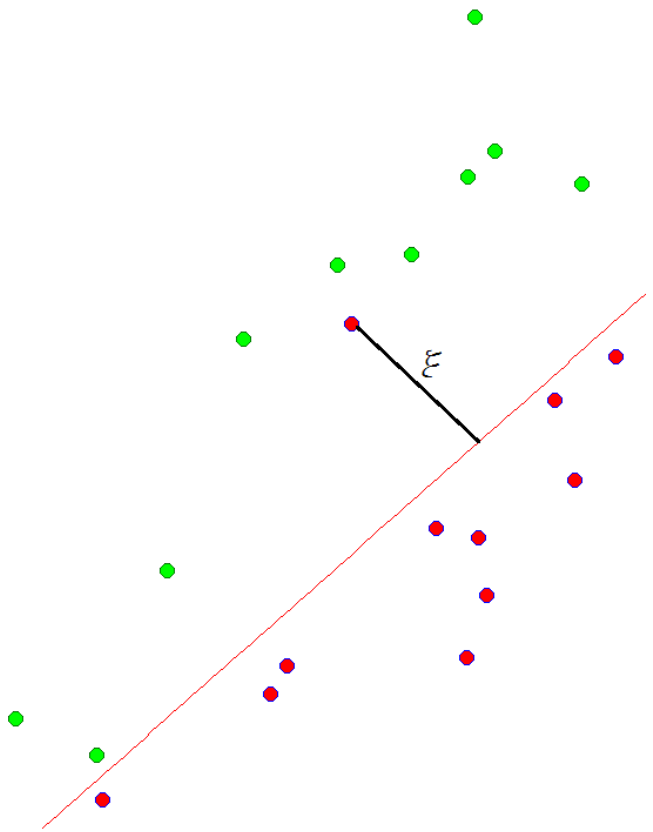
Non Separable Datasets

- What if the examples are ***not*** separable by a hyperplane?



Non Separable Datasets

- What if the examples are ***not*** separable by a hyperplane?



- Don't insist perfect separation as in Objective 1

Slacks

- Objective 2:

$$\arg\min_{\boldsymbol{w},b,\xi} \sum_{t=1}^{N} \xi_t$$
$$\text{subject to } r^{(t)}(\boldsymbol{w}^\top \boldsymbol{x}^{(t)} + b) > -\xi_t \text{ and } \xi_t \geqslant 0,\ \forall t = 1, \cdots, N$$

  - ξ_t's are called the ***slacks***
  - We minimize $\sum_{t=1}^{N} \xi_t$ instead of $\sum_{t=1}^{N} \xi_t^2$ to make the hypothesis robust to outliers
- Alternative form: $\arg\min_{\boldsymbol{w},b} \sum_{t=1}^{N} \max(0, -r^{(t)}(\boldsymbol{w}^\top \boldsymbol{x}^{(t)} + b))$
  - No slack to solve, no constraint, ***convex***
- $l(h(\boldsymbol{x}^{(t)}; \theta), r^{(t)}) := \max(0, -r^{(t)}(\boldsymbol{w}^\top \boldsymbol{x}^{(t)} + b))$ is called the ***hinge loss*** function (why?)
  - $emp(\theta; \mathcal{X}) = \sum_{t=1}^{N} l(h(\boldsymbol{x}^{(t)}; \theta), r^{(t)})$

Training the Perceptron Classifier

- Let $\widetilde{\boldsymbol{x}}^{(t)} = \begin{bmatrix} \boldsymbol{x}^{(t)} \\ 1 \end{bmatrix}$ and $\widetilde{\boldsymbol{w}}^{(t)} = \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix}$, we need to solve
  $\arg\min_{\widetilde{\boldsymbol{w}}} emp(\widetilde{\boldsymbol{w}}) := \sum_{t=1}^{N} \max(0, -r^{(t)} \widetilde{\boldsymbol{w}}^{\top} \widetilde{\boldsymbol{x}}^{(t)})$
- Let's consider the Gradient descent method due to its simplicity
- $\nabla emp(\widetilde{\boldsymbol{w}}) = \sum_{t=1}^{N} \nabla l^{(t)}(\widetilde{\boldsymbol{w}})$, where
  $$\nabla l^{(t)}(\widetilde{\boldsymbol{w}}) = \begin{cases} 0, & \text{if } r^{(t)} \widetilde{\boldsymbol{w}}^{\top} \widetilde{\boldsymbol{x}}^{(t)} > 0 \\ -r^{(t)} \widetilde{\boldsymbol{x}}^{(t)}, & \text{otherwise} \end{cases}$$
  - Can be also written as $\nabla emp(\widetilde{\boldsymbol{w}}) = -\frac{1}{2} \sum_{t=1}^{N} \widetilde{\boldsymbol{x}}^{(t)\top} \left(r^{(t)} - \operatorname{sgn}(\widetilde{\boldsymbol{w}}^{\top} \widetilde{\boldsymbol{x}}) \right)$

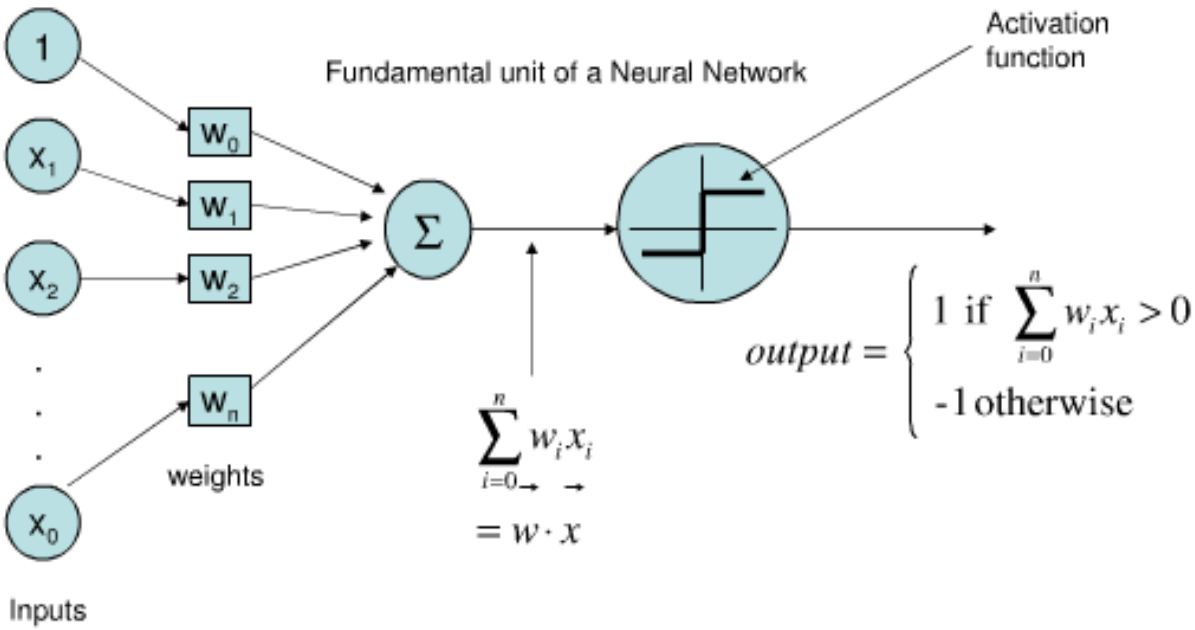
Repeat until convergence {

$$\begin{aligned} \widetilde{\boldsymbol{w}} &:= \widetilde{\boldsymbol{w}} - \eta \nabla emp(\widetilde{\boldsymbol{w}}) \\ &= \widetilde{\boldsymbol{w}} + \eta' \sum_{t=1}^{N} \widetilde{\boldsymbol{x}}^{(t)\top} \left(r^{(t)} - \operatorname{sgn}(\widetilde{\boldsymbol{w}}^{\top} \widetilde{\boldsymbol{x}}) \right), \\ &\quad \text{where } \eta' = \eta/2 \end{aligned}$$

}

Remarks (1)

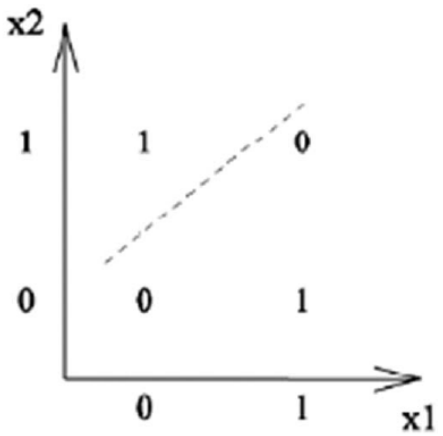
- A model for how individual neurons in human brain work



- Not good at recognizing non-linear classes/patterns
  - E.g., identifying an object in an image
- Improvements:
  - Chained to form a ***neural network***
  - Make instances linearly separable (to be discussed later)

Remarks (2)

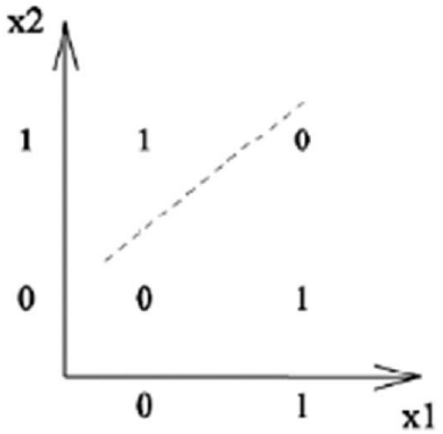
- If positive and negative examples are not linearly separable by $\text{sgn}(\boldsymbol{\beta}^\top \widetilde{\boldsymbol{x}}')$, the training algorithm will ***not*** converge



- Solution?

Remarks (2)

- If positive and negative examples are not linearly separable by $\text{sgn}(\boldsymbol{\beta}^\top \widetilde{\boldsymbol{x}}')$, the training algorithm will ***not*** converge



- Solution?
  - Limit the maximum number of iterations, or
  - Stop if $|emp(\widetilde{\boldsymbol{w}}^{(new)}) - emp(\widetilde{\boldsymbol{w}}^{(old)})| < \epsilon$

Update Rules: Perceptron vs. Logistic Regression

- If we "harden" the logistic function $\pi(\boldsymbol{x};\boldsymbol{\beta}) = \frac{1}{1+e^{-\boldsymbol{\beta}^\top \widetilde{\boldsymbol{x}}}}$ to

$$\pi(\boldsymbol{x};\boldsymbol{\beta}) = \begin{cases} 1, & \text{if } \boldsymbol{\beta}^\top \widetilde{\boldsymbol{x}} \geqslant 0, \\ 0, & \text{otherwise,} \end{cases}$$

  so that the gradient descent update rule becomes:

Repeat until convergence {

$$\begin{aligned} \boldsymbol{\beta} &:= \boldsymbol{\beta} + \eta \nabla l(\boldsymbol{\beta}) = \boldsymbol{\beta} + \eta \sum_{t=1}^{N} \widetilde{\boldsymbol{x}}^{(t)\top} \left(q^{(t)} - \pi\left(\boldsymbol{x}^{(t)};\boldsymbol{\beta}\right)\right) \\ &= \boldsymbol{\beta} + \eta' \sum_{t=1}^{N} \widetilde{\boldsymbol{x}}^{(t)\top} \left(r^{(t)} - \operatorname{sgn}(\boldsymbol{\beta}^\top \widetilde{\boldsymbol{x}}')\right), \end{aligned}$$

where $q^{(t)} = \frac{r^{(t)}+1}{2}$ and $\eta' = \eta/2$

}

- Despite its cosmetic similarity with logistic regression, perceptron learning has no simple probabilistic interpretation

Outline

Learning Multiple Classes

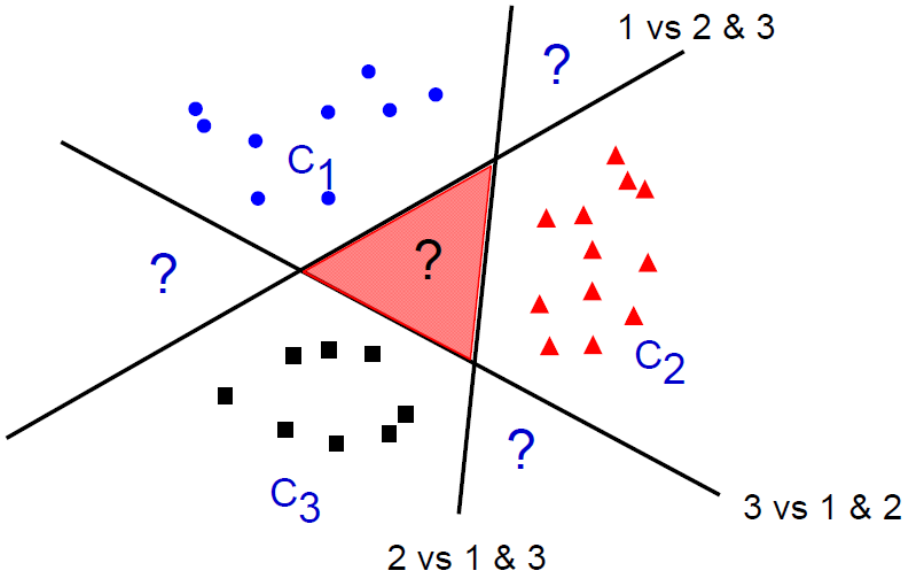
- What if we have K classes instead of 2?
- Applications:
  - OCR (Optical Character Recognition)
  - Medical diagnosis
  - Surveillance, etc.
- Training set: $\mathcal{X} = \{\boldsymbol{x}^{(t)}, \boldsymbol{r}^{(t)}\}_{t=1}^{N}$, where $\boldsymbol{r}^{(t)} \in \mathbb{R}^K$ and

$$r_i^{(t)} = \begin{cases} 1, & \boldsymbol{x}^{(t)} \in C_i \\ -1, & otherwise \end{cases}$$

Outline

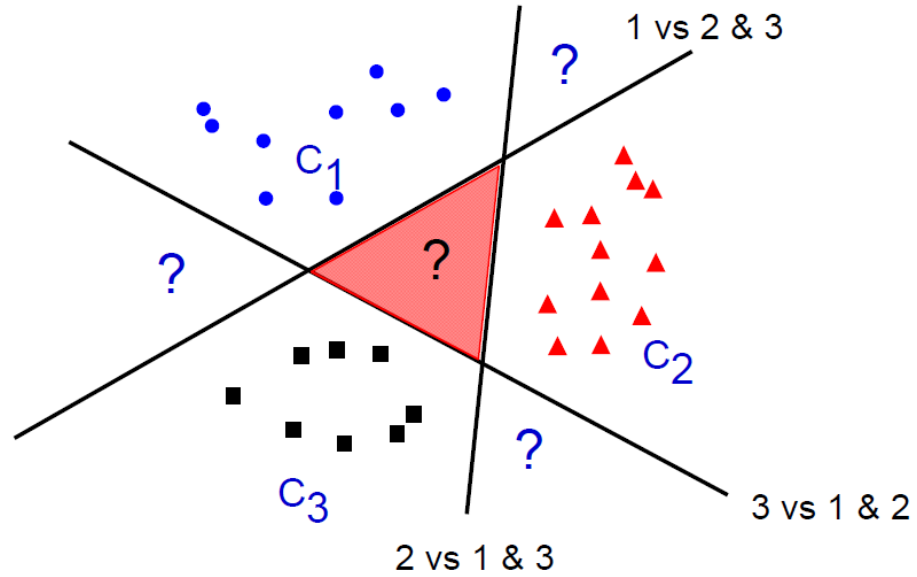
1 vs. All (1)

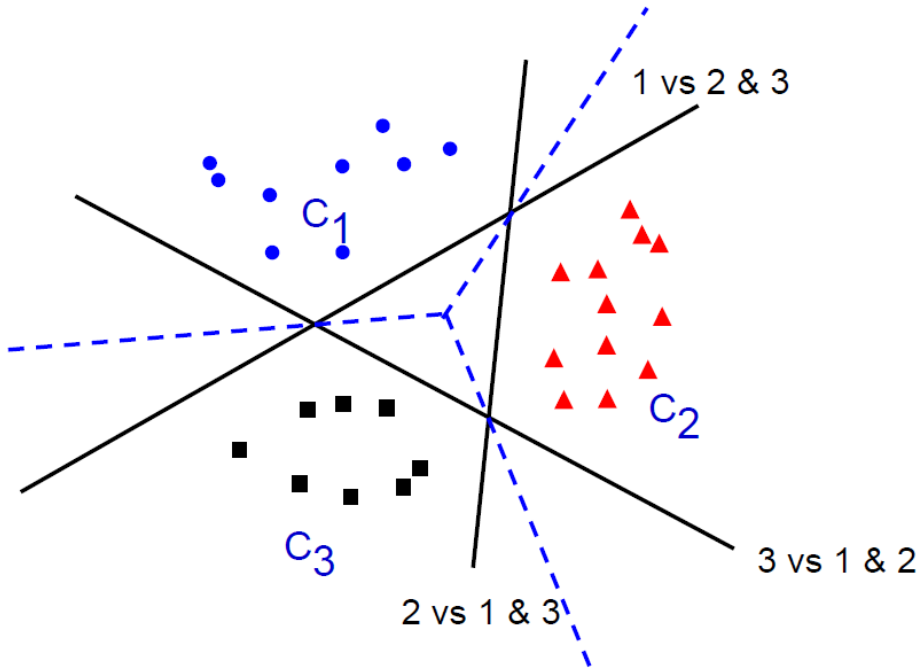
- Assume a model consisting of K hypotheses h_i
  - There is no need to for these K hypotheses to belong to the same hypotheses class
- Perform the two-class classification K times
  - Each time treat the examples of a certain class as positive and the rest as negative
- How to handle the ***cases of doubt***?



1 vs. All (1)

- Assume a model consisting of K hypotheses h_i
  - There is no need to for these K hypotheses to belong to the same hypotheses class
- Perform the two-class classification K times
  - Each time treat the examples of a certain class as positive and the rest as negative
- How to handle the ***cases of doubt***?
  - Define decision boundaries, e.g., $y' := \arg\max_i h_i(\boldsymbol{x}'; \theta_i)$





1 vs. All (2)

- Pros:
  - Easy to implement
  - # classifiers grows with K
- Cons:
  - Time consuming (each of the K classifiers takes the whole dataset as input)
  - Each classifier deals with imbalance dataset

1 vs 1

- Perform 1 vs. 1 classification $\binom{K}{2}$ times, and predict by voting
- Pros:
  - Avoid creating imbalanced dataset for each classifier
  - Faster and memory economic (each classifier takes only two classes in the dataset as input)
- Cons:
  - # classifiers grows with K^2, not suitable for datasets with massive classes

Wrappers based on Error-Correcting Codes

	h_1		$\cdots$		h_L
C_1	-1	-1	-1	-1	1
$\vdots$	1	-1	1	1	-1
C_K	1	1	-1	-1	1

- Rows: predefined codewords of length L
- Columns: a particular grouping of examples for training a two-class classifier

- To make prediction:
  1. Obtain a codeword for $\boldsymbol{x}'$ based on the predictions of L classifiers
  2. Assign $\boldsymbol{x}'$ to the label with the most similar codeword
- If codewords are designed such that each pair has Hamming distance at least s, then $\left\lfloor \frac{s-1}{2} \right\rfloor$ wrong predictions can be tolerated

Outline

Generalized Linear Models

TBA

Multi-Hyperplane Classifier

- Learn K separating hyperplanes simultaneously:

$$\arg\min_{\{\boldsymbol{w}_i, b_i\}_{i=1}^{K}, \xi} \sum_{t=1}^{N} \xi_{t,r}$$
$$\text{subject to } (\boldsymbol{w}_{idx(\boldsymbol{r}^{(t)})}^{\top} \boldsymbol{x}^{(t)} - b_{idx(\boldsymbol{r}^{(t)})}) - (\boldsymbol{w}_r^{\top} \boldsymbol{x}^{(t)} - b_r) > -\xi_{t,r}$$
$$\text{and } \xi_{t,r} \geqslant 0, \ \forall t, r \neq idx(\boldsymbol{r}^{(t)})$$

  - For an example of class r, the corresponding hyperplane should give value higher than those given by other hyperplanes
  - Prediction: $y' := \arg\max_i \boldsymbol{w}_i^{\top} \boldsymbol{x}' - b_i$
- Hyperplanes are correlated
  - No one will give values significantly higher than the others
- In practice,
  - There is little or no performance improvement over the wrappers
  - Very slow and memory hungry

Outline

k-NN Methods

- There are another simple ways, call k-NN methods, to make predictions
- Given a test instance $\boldsymbol{x}'$, predict its label by the (weighted) average of labels of k examples in $\mathcal{X}$ most similar to $\boldsymbol{x}'$
  - Applies to both continuous and discrete labels
- Needs a similarity metric $k(\boldsymbol{x}, \boldsymbol{y})$ between any two instances
  - E.g., *cosin* similarity: $k(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x}^\top \boldsymbol{y}}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|} \in [-1, 1]$
- Training: simply remember $\mathcal{X}$

Non-Parametric Methods

- k-NN methods are special cases of ***non-parametric*** (or ***memory-based***) methods
  - Non-parametric in the sense that f cannot be described by parameters
  - Memory-based in that all data (rather than just parameters) need to be memorized during the training process
- ***Lazy*** since the hypothesis is obtained only before the prediction
- This allows the development of ***local models***

Local Weighted Linear Regression

- Recall in (eager) linear regression, we fit $\boldsymbol{w} \in \mathbb{R}^{d+1}$ to minimize the SSE: $\sum_i (r^{(i)} - \boldsymbol{w}^\top \begin{bmatrix} 1 \\ \boldsymbol{x}^{(i)} \end{bmatrix})^2$
- Local model: fit $\boldsymbol{w}$ to minimize SSE ***local to the instance $\boldsymbol{x}'$ we want to predict***:

$$\sum_i l(\boldsymbol{x}^{(i)}; \boldsymbol{x}') (r^{(i)} - \boldsymbol{w}^\top \begin{bmatrix} 1 \\ \boldsymbol{x}^{(i)} \end{bmatrix})^2$$

  where $l : \mathbb{R}^d \to \mathbb{R}$ is a weighting function
  - Idea: only examples nearby (or local to) $\boldsymbol{x}'$ should be taken into account in $emp(\theta; \mathcal{X})$
- Possible choice for l: $l(\boldsymbol{x}^{(i)}; \boldsymbol{x}') := \exp\left(-\frac{(\boldsymbol{x}^{(i)} - \boldsymbol{x}')^2}{2\tau^2}\right)$ for some τ (mimics k-NN)

Summary of Supervised Learning Models

- Three main categories (either parametric or non-parametric):

1. Those learning the ***discriminant functions*** f's (no probability interpretation)
   - E.g., perceptron, kNN, etc.
2. Those based on probability and learn $p(r|\boldsymbol{x})$ directly
   - E.g., linear regression, logistic regression, etc.
   - $p(r|\boldsymbol{x};\theta)$ with θ (constant) estimated from $\mathcal{X}$
   - Methods in 1 and 2 are called ***discriminative methods***
3. Those learn $p(r|\boldsymbol{x})$ indirectly from $p(\boldsymbol{x}|r)p(r)$
   - To be discussed later
   - These are called ***generative methods***, as $p(\boldsymbol{x}|r)p(r)$ explains how $\mathcal{X}$ is generated