Multi-way classification.  
Decision trees.

Milos Hauskrecht  
milos@cs.pitt.edu  
5329 Sennott Square

Multi-way classification

- Binary classification  \( Y = \{0,1\} \)
- Multi-way classification
  - K classes  \( Y = \{0,1,\ldots,K-1\} \)
  - Goal: learn to classify correctly K classes
  - Or learn  \( f : X \rightarrow \{0,1,\ldots,K-1\} \)
- Errors:
  - Zero-one (misclassification) error for an example:
    \[
    Error_1(x_i, y_i) = \begin{cases} 
    1 & f(x_i, w) \neq y_i \\
    0 & f(x_i, w) = y_i 
    \end{cases}
    \]
  - Mean misclassification error (for a dataset):
    \[
    \frac{1}{n} \sum_{i=1}^{n} Error_1(x_i, y_i)
    \]
Multi-way classification

Approaches:

• Generative model approach
  – Generative model of the distribution \( p(x, y) \)
  – Learns the parameters of the model through density estimation techniques
  – Discriminant functions are based on the model
    • “Indirect” learning of a classifier

• Discriminative approach
  – Parametric discriminant functions
  – Learns discriminant functions directly
    • A logistic regression model.

---

Generative model approach

Indirect:

1. Represent and learn the distribution \( p(x, y) \)
2. Define and use probabilistic discriminant functions
   \[ g_i(x) = \log p(y = i | x) \]

Model \( p(x, y) = p(x | y) p(y) \)
• \( p(x | y) \) = Class-conditional distributions (densities)
  \[ p(x | y = i) \quad \forall i \quad 0 \leq i \leq K - 1 \]
• \( p(y) \) = Priors on classes
• - probability of class \( y \)
  \[ \sum_{i=1}^{K} p(y = i) = 1 \]
Multi-way classification. Example

Multi-way classification
Making class decision

**Discriminant functions** can be based on:

- **Likelihood of data** – choose the class (Gaussian) that explains the input data \( \mathbf{x} \) better (likelihood of the data)
  
  **Choice:**
  \[
  i = \arg \max_{i=0, \ldots, k-1} p(\mathbf{x} | \Theta_i)
  \]
  
  \[
  p(\mathbf{x} | \Theta_i) \approx p(\mathbf{x} | \mu_i, \Sigma_i) \quad \text{For Gaussians}
  \]

- **Posterior of a class** – choose the class with higher posterior probability
  
  **Choice:**
  \[
  i = \arg \max_{i=0, \ldots, k-1} p(y = i | \mathbf{x}, \Theta_i)
  \]
  
  \[
  p(y = i | \mathbf{x}) = \frac{p(\mathbf{x} | \Theta_i) p(y = i)}{\sum_{j=0}^{k-1} p(\mathbf{x} | \Theta_j) p(y = j)}
  \]

**Discriminative approach**

- **Parametric model** of discriminant functions
- Learns the discriminant functions directly

How to learn to classify multiple classes, say 0,1,2?

**Approach 1:**

- A binary logistic regression on every class versus the rest

  \[
  \begin{align*}
  1 & \quad 0 \text{ vs. (1 or 2)} \\
  x_i & \quad 1 \text{ vs. (0 or 2)} \\
  \ast & \quad 2 \text{ vs. (0 or 1)} \\
  x_d &
  \end{align*}
  \]
Multi-way classification. Example

Multi-way classification. Approach 1.
Multi-way classification. Approach 1.

- 0 vs \{1,2\}
- 1 vs \{0,2\}
- 2 vs \{0,1\}

Region of nobody

Ambiguous region
**Discriminative approach.**

How to learn to classify multiple classes, say 0,1,2?

**Approach 2:**
- A binary logistic regression on all pairs

![Diagram showing binary logistic regression for 0 vs. 1, 0 vs. 2, and 1 vs. 2 classes.]

---

**Multi-way classification. Example**

![Plot illustrating multi-way classification example with classes 0, 1, and 2.]
Multi-way classification. Approach 2

Ambiguous region

0 vs 1

0 vs 2

1 vs 2
Multi-way classification. Approach 2

Multi-way classification with softmax

- A solution to the problem of having an ambiguous region

$$p(y = i \mid x) = \mu_i = \frac{\exp(w_i^T x)}{\sum_j \exp(w_j^T x)} \quad \sum_i \mu_i = 1$$
Multi-way classification with softmax

Learning of the softmax model

- Learning of parameters $w$: statistical view

Assume outputs $y$ are transformed as follows

$$y \in \{0, 1, \ldots, k-1\}$$
Learning of the softmax model

- Learning of the parameters $w$: statistical view
- **Likelihood of outputs**
  \[ L(D, w) = p(Y | X, w) = \prod_{i=1}^{n} p(y_i | x_i, w) \]
- We want parameters $w$ that maximize the likelihood
- **Log-likelihood trick**
  - Optimize log-likelihood of outputs instead:
  \[ l(D, w) = \log \prod_{i=1}^{n} p(y_i | x, w) = \sum_{i=1}^{n} \log p(y_i | x, w) \]
  \[ = \sum_{i=1}^{n} \sum_{q=0}^{k-1} \log \mu_{i,q} = \sum_{i=1}^{n} \sum_{q=0}^{k-1} y_{i,q} \log \mu_{i,q} \]
- **Objective to optimize**
  \[ J(D, w) = -\sum_{i=1}^{n} \sum_{q=0}^{k-1} y_{i,q} \log \mu_{i,q} \]

Learning of the softmax model

- **Error to optimize:**
  \[ J(D_i, w) = -\sum_{i=1}^{n} \sum_{q=0}^{k-1} y_{i,q} \log \mu_{i,q} \]
- **Gradient**
  \[ \frac{\partial}{\partial w_{jq}} J(D_i, w) = \sum_{i=1}^{n} -x_{i,j} (y_{i,q} - \mu_{i,q}) \]
- The same very easy **gradient update** as used for the binary logistic regression
  \[ w_{q} \leftarrow w_{q} + \alpha \sum_{i=1}^{n} (y_{i,q} - \mu_{i,q}) x_i \]
- But now we have to update the weights of $k$ networks
Multi-way classification

• Yet another approach 3

Decision trees

• An alternative approach to classification:
  – Partition the input space to regions
  – Regress or classify independently in every region
Decision trees

- The partitioning idea is used in the decision tree model:
  - Split the space recursively according to inputs in $x$
  - Regress or classify at the bottom of the tree

**Example:**
Binary classification $\{0,1\}$
Binary attributes $x_1, x_2, x_3$

```
0 1 0 0 1 1 0
```

classify

---

Decision trees

How to construct the decision tree?

- **Top-bottom algorithm:**
  - Find the best split condition (quantified based on the impurity measure)
  - Stops when no improvement possible

- **Impurity measure:**
  - Measures how well are the two classes separated
  - Ideally we would like to separate all 0s and 1

- Splits of **finite vs. continuous value attributes**
  Continuous value attributes conditions: $x_3 \leq 0.5$
Impurity measure

Let \(| D |\) - Total number of data entries
\(| D_i |\) - Number of data entries classified as \(i\)

\[ p_i = \frac{| D_i |}{| D |} \] - ratio of instances classified as \(i\)

- **Impurity measure** defines how well the classes are separated
- In general the impurity measure should satisfy:
  - Largest when data are split evenly for attribute values
    \[ p_i = \frac{1}{\text{number of classes}} \]
  - Should be 0 when all data belong to the same class

Impurity measures

- There are various impurity measures used in the literature
  - **Entropy based measure** (Quinlan, C4.5)
    \[ I(D) = \text{Entropy} (D) = -\sum_{i=1}^{k} p_i \log p_i \]

    Example for \(k=2\)

  - **Gini measure** (Breiman, CART)
    \[ I(D) = \text{Gini} (D) = 1 - \sum_{i=1}^{k} p_i^2 \]
Impurity measures

- **Gain due to split** – expected reduction in the impurity measure (entropy example)

\[
Gain(D, A) = Entropy(D) - \sum_{v \in \text{Values}(A)} \frac{|D^v|}{|D|} \cdot Entropy(D^v)
\]

| \(D^v|\) - a partition of \(D\) with the value of attribute \(A = v\)

![Decision tree structure](image)

Decision tree learning

- **Greedy learning algorithm:**
  - Repeat until no or small improvement in the purity
    - Find the attribute with the highest gain
    - Add the attribute to the tree and split the set accordingly

- Builds the tree in the top-down fashion
  - Gradually expands the leaves of the partially built tree
- The method is greedy
  - It looks at a single attribute and gain in each step
  - May fail when the combination of attributes is needed to improve the purity (parity functions)
Decision tree learning

- **Limitations of greedy methods**
  Cases in which a combination of two or more attributes improves the impurity

By reducing the impurity measure we can grow **very large trees**

**Problem: Overfitting**
- We may split and classify very well the training set, but we may do worse in terms of the generalization error

**Solutions to the overfitting problem:**
- **Solution 1.**
  - Prune branches of the tree built in the first phase
  - Use validation set to test for the overfit
- **Solution 2.**
  - Test for the overfit in the tree building phase
  - Stop building the tree when performance on the validation set deteriorates
K-Nearest-Neighbours for Classification

• Given a data set with \( N_k \) data points from class \( C_k \) and \( \sum_k N_k = N \), we have

\[
p(x) = \frac{K}{NV}
\]

• and correspondingly

\[
p(x|C_k) = \frac{K_k}{N_k V}.
\]

• Since \( p(C_k) = N_k/N \) Bayes’ theorem gives

\[
p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} = \frac{K_k}{K}.
\]
Nonparametric kernel-based classification

- **Kernel function:** $k(x, x')$
  - Models similarity between $x, x'$
  - **Example:** Gaussian kernel we used in kernel density estimation
    \[
    k(x, x') = \frac{1}{(2\pi h^2)^{D/2}} \exp\left(-\frac{(x - x')^2}{2h^2}\right)
    \]
    \[
    p(x) = \frac{1}{N} \sum_{i=1}^{N} k(x, x_i)
    \]

- **Kernel for classification**
  \[
  p(y = C_k \mid x) = \frac{\sum_{x', y = C_k} k(x, x')}{\sum_{x'} k(x, x')}
  \]