Motivation

- Real-world problems often have multiple classes: text, speech, image, biological sequences.

- Algorithms studied so far: designed for binary classification problems.

- How do we design multi-class classification algorithms?
  - can the algorithms used for binary classification be generalized to multi-class classification?
  - can we reduce multi-class classification to binary classification?
Multi-Class Classification Problem

- **Training data:** sample drawn i.i.d. from set $X$ according to some distribution $D$,
  \[ S = ((x_1, y_1), \ldots, (x_m, y_m)) \in X \times Y, \]
  - mono-label case: $\text{Card}(Y) = k$.
  - multi-label case: $Y = \{-1, +1\}^k$.

- **Problem:** find classifier $h : X \to Y$ in $H$ with small generalization error,
  - mono-label case: $R(h) = E_{x \sim D}[1_{h(x) \neq f(x)}]$.
  - multi-label case: $R(h) = E_{x \sim D} \left[ \frac{1}{k} \sum_{l=1}^{k} 1_{[h(x)]_l \neq [f(x)]_l} \right]$. 

\[ X \]

\[ Y, \]
Notes

- In most tasks considered, number of classes $k \leq 100$.

- For $k$ large, problem often not treated as a multi-class classification problem (ranking or density estimation, e.g., automatic speech recognition).

- Computational efficiency issues arise for larger $k$s.

- In general, classes not balanced.
Multi-Class Classification - Margin

- **Hypothesis set** $H$:
  - functions $h: X \times Y \to \mathbb{R}$.
  - label returned: $x \mapsto \arg\max_{y \in Y} h(x, y)$.

- **Margin**:
  - $\rho_h(x, y) = h(x, y) - \max_{y' \neq y} h(x, y')$.
  - error: $1_{\rho_h(x, y) \leq 0} \leq \Phi_\rho(\rho_h(x, y))$.
  - empirical margin loss:
    \[
    \hat{R}_\rho(h) = \frac{1}{m} \sum_{i=1}^{m} \Phi_\rho(\rho_h(x, y)).
    \]
Multi-Class Margin Bound


**Theorem:** let \( H \subseteq \mathbb{R}^{X \times Y} \) with \( Y = \{1, \ldots, k\} \). Fix \( \rho > 0 \). Then, for any \( \delta > 0 \), with probability at least \( 1 - \delta \), the following multi-class classification bound holds for all \( h \in H \):

\[
R(h) \leq \widehat{R}_\rho(h) + \frac{4k}{\rho} \mathcal{R}_m(\Pi_1(H)) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}},
\]

with \( \Pi_1(H) = \{x \mapsto h(x, y) : y \in Y, h \in H\} \).
Kernel Based Hypotheses

- **Hypothesis set** \( H_{K,p} : \)
  - \( \Phi \) feature mapping associated to PDS kernel \( K \).
  - **functions** \((x, y) \mapsto w_y \cdot \Phi(x), y \in \{1, \ldots, k\} \).
  - **label returned:** \( x \mapsto \arg\max_{y \in \{1, \ldots, k\}} w_y \cdot \Phi(x) \).
  - **for any** \( p \geq 1, \)

\[
H_{K,p} = \{(x, y) \in X \times [1, k] \mapsto w_y \cdot \Phi(x) : W = (w_1, \ldots, w_k)\top, \|W\|_{H,p} \leq \Lambda\}.
\]
Multi-Class Margin Bound - Kernels

**Theorem:** let $K: X \times X \rightarrow \mathbb{R}$ be a PDS kernel and let $\Phi: X \rightarrow \mathbb{H}$ be a feature mapping associated to $K$. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following multiclass bound holds for all $h \in H_{K,p}$:

$$R(h) \leq \hat{R}_\rho(h) + 4k \sqrt{\frac{r^2 \Lambda^2}{\rho^2 m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}},$$

where $r^2 = \sup_{x \in X} K(x, x)$. 

(MM et al. 2012)
Approaches

- Single classifier:
  - Multi-class SVMs.
  - AdaBoost.MH.
  - Conditional Maxent.
  - Decision trees.

- Combination of binary classifiers:
  - One-vs-all.
  - One-vs-one.
  - Error-correcting codes.
Multi-Class SVMs

(Weston and Watkins, 1999; Crammer and Singer, 2001)

- Optimization problem:

\[
\min_{\mathbf{w}, \xi} \frac{1}{2} \sum_{l=1}^{k} \| \mathbf{w}_l \|^2 + C \sum_{i=1}^{m} \xi_i
\]

subject to:

\[
\mathbf{w}_{y_i} \cdot \mathbf{x}_i + \delta_{y_i,l} \geq \mathbf{w}_l \cdot \mathbf{x}_i + 1 - \xi_i
\]

\((i, l) \in [1, m] \times Y.\)

- Decision function:

\[
h : \mathbf{x} \mapsto \arg\max_{l \in Y} (\mathbf{w}_l \cdot \mathbf{x}).
\]
Notes

- Directly based on generalization bounds.

- **Comparison with** (Weston and Watkins, 1999): single slack variable per point, maximum of slack variables (penalty for worst class):

\[
\sum_{l=1}^{k} \xi_{il} \rightarrow \max_{l=1}^{k} \xi_{il}.
\]

- PDS kernel instead of inner product

- Optimization: complex constraints, \(m^k\)-size problem.
  - specific solution based on decomposition into \(m\) disjoint sets of constraints (Crammer and Singer, 2001).
Dual Formulation

- **Optimization problem:** \( \alpha \) \textit{i\textsuperscript{th} row of matrix} \( \alpha \in \mathbb{R}^{m \times k} \)

\[
\max_{\alpha = [\alpha_{ij}]} \sum_{i=1}^{m} \alpha_i \cdot e_{y_i} - \frac{1}{2} \sum_{i=1}^{m} (\alpha_i \cdot \alpha_j)(x_i \cdot x_j)
\]

subject to: \( \forall i \in [1, m], (0 \leq \alpha_{iy_i} \leq C) \land (\forall j \neq y_i, \alpha_{ij} \leq 0) \land (\alpha_i \cdot 1 = 0). \)

- **Decision function:**

\[
h(x) = \arg\max_{l=1}^{k} \left( \sum_{i=1}^{m} \alpha_{il}(x_i \cdot x) \right).
\]
AdaBoost

Training data (multi-label case):

\[(x_1, y_1), \ldots, (x_m, y_m) \in X \times \{-1, 1\}^k.\]

Reduction to binary classification:

- each example leads to \(k\) binary examples:

\[(x_i, y_i) \rightarrow ((x_i, 1), y_i[1]), \ldots, ((x_i, k), y_i[k]), i \in [1, m].\]

- apply AdaBoost to the resulting problem.

- choice of \(\alpha_t\).

Computational cost: \(mk\) distribution updates at each round.
AdaBoost.MH

\( H \subseteq \{-1, +1\}^k \times X \times Y \).

**AdaBoost.MH**

\[ S = ((x_1, y_1), \ldots, (x_m, y_m)) \]

1. for \( i \leftarrow 1 \) to \( m \) do
2. for \( l \leftarrow 1 \) to \( k \) do
3. \[ D_1(i, l) \leftarrow \frac{1}{mk} \]
4. for \( t \leftarrow 1 \) to \( T \) do
5. \( h_t \leftarrow \) base classifier in \( H \) with small error \( \epsilon_t = \Pr_{D_t}[h_t(x_i, l) \neq y_i[l]] \)
6. \( \alpha_t \leftarrow \) choose \[ \triangleright \text{to minimize } Z_t \]
7. \( Z_t \leftarrow \sum_{i,l} D_t(i, l) \exp(-\alpha_t y_i[l] h_t(x_i, l)) \)
8. for \( i \leftarrow 1 \) to \( m \) do
9. for \( l \leftarrow 1 \) to \( k \) do
10. \[ D_{t+1}(i, l) \leftarrow \frac{D_t(i, l) \exp(-\alpha_t y_i[l] h_t(x_i, l))}{Z_t} \]
11. \( f_T \leftarrow \sum_{t=1}^{T} \alpha_t h_t \)
12. return \( h_T = \text{sgn}(f_T) \)
Bound on Empirical Error

**Theorem:** The empirical error of the classifier output by AdaBoost.MH verifies:

\[ \hat{R}(h) \leq \prod_{t=1}^{T} Z_t. \]

**Proof:** similar to the proof for AdaBoost.

**Choice of \( \alpha_t \):**

- for \( H \subseteq (\{-1, +1\}^k)^X \times Y \), as for AdaBoost, \( \alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t} \).
- for \( H \subseteq ([-1, 1]^k)^X \times Y \), same choice: minimize upper bound.
- other cases: numerical/approximation method.
Objective function:

\[
F(\alpha) = \sum_{i=1}^{m} \sum_{l=1}^{k} e^{-y_i[l]f_n(x_i,l)} = \sum_{i=1}^{m} \sum_{l=1}^{k} e^{-y_i[l]} \sum_{t=1}^{n} \alpha_t h_t(x_i,l).
\]

All comments and analysis given for AdaBoost apply here.

Alternative: Adaboost.MR, which coincides with a special case of RankBoost (ranking lecture).
Decision Trees
Different Types of Questions

- Decision trees
  - $X \in \{\text{blue, white, red}\}$: categorical questions.
  - $X \leq a$: continuous variables.

- Binary space partition (BSP) trees:
  - $\sum_{i=1}^{n} \alpha_i X_i \leq a$: partitioning with convex polyhedral regions.

- Sphere trees:
  - $||X - a_0|| \leq a$: partitioning with pieces of spheres.
In each region $R_t$,

- **classification**: majority vote - ties broken arbitrarily,
  \[ \hat{y}_t = \arg\max_{y \in Y} \left| \{ x_i \in R_t : i \in [1, m], y_i = y \} \right|. \]

- **regression**: average value,
  \[ \hat{y}_t = \frac{1}{|S \cap R_t|} \sum_{\substack{x_i \in R_t \atop i \in [1, m]}} y_i. \]

- **Form of hypotheses**:
  \[ h : x \mapsto \sum_t \hat{y}_t 1_{x \in R_t}. \]
Training

Problem: general problem of determining partition with minimum empirical error is NP-hard.

Heuristics: greedy algorithm.

- for all $j \in [1, N]$, $\theta \in \mathbb{R}$,
  \[
  R^+(j, \theta) = \{x_i \in R : x_i[j] \geq \theta, i \in [1, m]\},
  \]
  \[
  R^-(j, \theta) = \{x_i \in R : x_i[j] < \theta, i \in [1, m]\}.
  \]

Decision-Trees($S = ((x_1, y_1), \ldots, (x_m, y_m))$)

1. $P \leftarrow \{S\} \triangleright$ initial partition
2. for each region $R \in P$ such that Pred$(R)$ do
3. 
   $(j, \theta) \leftarrow \text{argmin}_{(j, \theta)} \text{error}(R^-(j, \theta)) + \text{error}(R^+(j, \theta))$
4. 
   $P \leftarrow P - R \cup \{R^-(j, \theta), R^+(j, \theta)\}$
5. return $P$
Splitting/Stopping Criteria

- **Problem**: larger trees overfit training sample.

- **Conservative splitting**:
  - split node only if loss reduced by some fixed value $\eta > 0$.
  - issue: seemingly bad split dominating useful splits.

- **Grow-then-prune technique (CART)**:
  - grow very large tree, $\text{Pred}(R): |R| > |n_0|$.
  - prune tree based on: $F(T) = \widehat{\text{Loss}}(T) + \alpha |T|$, $\alpha \geq 0$ parameter determined by cross-validation.
Decision Tree Tools

Most commonly used tools for learning decision trees:

- **CART** (classification and regression tree) (Breiman et al., 1984).
- **C4.5** (Quinlan, 1986, 1993) and **C5.0** (RuleQuest Research) a commercial system.

Differences: minor between latest versions.
Approaches

- Single classifier:
  - SVM-type algorithm.
  - AdaBoost-type algorithm.
  - Conditional Maxent.
  - Decision trees.

- Combination of binary classifiers:
  - One-vs-all.
  - One-vs-one.
  - Error-correcting codes.
One-vs-All

- **Technique:**
  - for each class \( l \in Y \) learn binary classifier \( h_l = \text{sgn}(f_l) \).
  - combine binary classifiers via voting mechanism, typically majority vote: \( h : x \mapsto \arg\max_{l \in Y} f_l(x) \).

- **Problem:** poor justification (in general).
  - calibration: classifier scores not comparable.
  - nevertheless: simple and frequently used in practice, computational advantages in some cases.
One-vs-One

Technique:

- for each pair $(l, l') \in Y, l \neq l'$ learn binary classifier $h_{ll'} : X \rightarrow \{0, 1\}$.
- combine binary classifiers via majority vote:
  \[
  h(x) = \arg\max_{l': Y} \left| \left\{ l : h_{ll'}(x) = 1 \right\} \right|.
  \]

Problem:

- computational: train $k(k - 1)/2$ binary classifiers.
- overfitting: size of training sample could become small for a given pair.
### Computational Comparison

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>One-vs-all</strong></td>
<td>$O(kB_{\text{train}}(m))$</td>
<td>$O(kB_{\text{test}})$</td>
</tr>
<tr>
<td></td>
<td>$O(km^\alpha)$</td>
<td></td>
</tr>
<tr>
<td><strong>One-vs-one</strong></td>
<td>$O(k^2B_{\text{train}}(m/k))$</td>
<td>$O(k^2B_{\text{test}})$</td>
</tr>
<tr>
<td></td>
<td>(on average)</td>
<td>smaller $N_{SV}$ per $B$</td>
</tr>
<tr>
<td></td>
<td>$O(k^{2-\alpha}m^\alpha)$</td>
<td></td>
</tr>
</tbody>
</table>

Time complexity for SVMs, $\alpha$ less than 3.
Error-Correcting Code Approach

(Dietterich and Bakiri, 1995)

Idea:

- assign $F$-long binary code word to each class:
  \[ M = [M_{lj}] \in \{0, 1\}^{[1,k]} \times [1,F]. \]

- learn binary classifier $f_j: X \rightarrow \{0, 1\}$ for each column. Example $x$ in class $l$ labeled with $M_{lj}$.

- classifier output: \( f(x) = (f_1(x), \ldots, f_F(x)) \),

\[ h: x \mapsto \arg\min_{l \in Y} \text{d}_{\text{Hamming}}(M_l, f(x)). \]
Illustration

- **8 classes, code-length: 6.**

<table>
<thead>
<tr>
<th>codes</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$f_1(x)$</th>
<th>$f_2(x)$</th>
<th>$f_3(x)$</th>
<th>$f_4(x)$</th>
<th>$f_5(x)$</th>
<th>$f_6(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

new example $x$
Error-Correcting Codes - Design

Main ideas:

- independent columns: otherwise no effective discrimination.
- distance between rows: if the minimal Hamming distance between rows is $d$, then the multi-class can correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors.
- columns may correspond to features selected for the task.
- one-vs-all and one-vs-one (with ternary codes) are special cases.
Extensions

Matrix entries in \{-1, 0, +1\}:
- examples marked with 0 disregarded during training.
- one-vs-one becomes also a special case.

Margin loss \(L\): function of \(y f(x)\), e.g., hinge loss.
- Hamming loss:
  \[ h(x) = \arg\min_{l \in \{1, \ldots, k\}} \sum_{j=1}^{F} \frac{1 - \text{sgn} (M_{lj} f_{j}(x))}{2}. \]
- Margin loss:
  \[ h(x) = \arg\min_{l \in \{1, \ldots, k\}} \sum_{j=1}^{F} L(M_{lj} f_{j}(x)). \]

(Allwein et al., 2000)
Applications

- One-vs-all approach is the most widely used.
- No clear empirical evidence of the superiority of other approaches (Rifkin and Klautau, 2004).
  - except perhaps on small data sets with relatively large error rate.
- Large structured multi-class problems: often treated as ranking problems (see ranking lecture).
References


References


