Outline

1. Introduction

2. Margin Bounds

3. Single classifier Approach
   - Multi-Class SVMs
   - AdaBoost
   - Decision Trees

4. Combination of binary classifiers Approach
   - One-vs-All
   - One-vs-One
   - Error-correcting codes
Outline

1. Introduction

2. Margin Bounds

3. Single classifier Approach
   - Multi-Class SVMs
   - AdaBoost
   - Decision Trees

4. Combination of binary classifiers Approach
   - One-vs-All
   - One-vs-One
   - Error-correcting codes
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 according to some distribution, $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in X \times Y$,
  - mono-label case: $|Y| = k$.
  - multi-label case: $|Y| = \{+1, -1\}^k$.
- **Problem**: find classifier $h : X \rightarrow Y$ in $H$ with small generalization error,
  - mono-label case: $R_D(h) = E_{x \sim D}[1_{h(x) \neq f(x)}]$.
  - multi-label case: $R_D(h) = E_{x \sim D}[\frac{1}{k} \sum_{l=1}^{k} 1_{h(x) \neq f(x)}]$.
In most tasks considered, number of classes $k \leq 100$. For $k$ large, problem often not treated as a multiclass classification problem (ranking or density estimation, e.g., automatic speech recognition).

Computational efficiency issues arise for larger values of $k$.

In general, classes not balanced.
Outline

1. Introduction
2. Margin Bounds
3. Single classifier Approach
   - Multi-Class SVMs
   - AdaBoost
   - Decision Trees
4. Combination of binary classifiers Approach
   - One-vs-All
   - One-vs-One
   - Error-correcting codes
Multi-Class Classification - Margin

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

- Margin:
  \[
  \rho_h(x, y) = h(x, y) - \max_{y' \neq y} h(x, y')
  \]

- empirical margin loss:
  \[
  \hat{R}_\rho(h) = \frac{1}{m} \sum_{i=0}^{m} \Phi_\rho(\rho_h(x, y))
  \]
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 $h \in H$:

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

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

Proof.
Sketch of Proof

- **STEP1**: Structure similar to binary classification bound.
- **STEP2**: Construct a set of hypothesis on binary classification?
  - $\tilde{H}$ margin function of $H$ and $\mathcal{H} = \Phi \circ \tilde{H}$
- **STEP3**: Applying binary bound theorem, we have the result in $\mathcal{R}_m(\tilde{H})$
- **STEP4**: $\mathcal{R}_m(\tilde{H})$ can be expressed as $\mathcal{R}_m(\Pi_1(H))$ using definition of $\rho_h(x, y)$
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: $x \mapsto \text{argmax}_{y \in \{1, \ldots, k\}} w_y \cdot \Phi(x)$.
  - for any $p \geq 1$,
    $$H_{K,p} = \left\{ (x, y) \in X \times [1, k] \mapsto w_y \cdot \Phi(x) : \right\} \left. W = (w_1, \ldots, w_k)^T, \|W\|_{H,p} \leq \Lambda \right\}$$
Multi-Class Margin Bound - Kernels

Theorem

Let $K : X \times X \rightarrow \mathbb{R}$ be a PDS kernel and $\Phi : X \rightarrow \mathbb{H}$ 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 $h \in H_{K,\rho}$:

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

where

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

Proof.
Sketch of Proof

- **STEP1**: By def. $\| W \|_{H, p} \leq \Lambda$.
- **STEP2**: Positive Definite Symmetric kernel with $K(x, x) \leq r^2$
- **STEP3**: Applying Cauchy-Schwartz and Jensen’s inequalities should yield $\mathcal{R}_m(\Pi_1(H)) \leq r\Lambda$ using the def. of $\rho_h$
Approaches

- **Single classifier**
  - Multi-class SVMs.
  - AdaBoost.MH.
  - Decision trees.

- **Combination of binary classifiers**
  - One-vs-all.
  - One-vs-one.
  - Error-correcting codes.
Outline

1. Introduction
2. Margin Bounds
3. Single classifier Approach
   - Multi-Class SVMs
   - AdaBoost
   - Decision Trees
4. Combination of binary classifiers Approach
   - One-vs-All
   - One-vs-One
   - Error-correcting codes
Multi-Class SVMs

- Optimization problem:

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

subject to:

\[
wy_i \cdot x_i + \delta_{y_i,l} \geq w_l \cdot x_i + 1 - \xi_i \quad (i, l) \in [1, m] \times Y.
\]

- Decision function:

\[
h : x \mapsto \arg\max_{l \in Y} (w_l \cdot x)
\]
-:NOTE:-

- Directly based on generalization bounds.
- 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, \(mk\)-size problem.
  - specific solution based on decomposition into \(m\)-disjoint sets of constraints.
Dual Formulation

- **Optimization problem:** $\alpha_i$ $i$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:**

  $h = \arg\max_{k} \left( \sum_{i=1}^{m} \alpha_{ik}(x_i \cdot x) \right)$
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.
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).
$H \subseteq (\{-1, +1\}^k)(X \times Y)$.

\[
\text{AdaBoost.MH}(S = ((x_1, y_1), \ldots, (x_m, y_m)))
\]

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

Choice of \( \alpha_t \):

- for \( H \subset (\{-1, +1\}^k)^{X \times Y} \) as for AdaBoost,
  \[ \alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}. \]
- for \( H \subset ([-1, +1]^k)^{X \times Y} \), same choice: minimize upper bound.
- other cases: numerical/approximation method.
Decision Trees
Different Types of Questions

- Decision trees
  - $X \in \{blue, white, red\}$ 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.
Hypotheses

- 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,
    $$\hat{y}_t = \frac{1}{|S \cap R_t|} \sum_{x_i \in R_t, i \in [1, m]} y_i.$$

- 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

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

\[
\text{DETECTION-TREES}(S = ((x_1, y_1), \ldots, (x_m, y_m)))
\]

1. \( P \leftarrow \{S\} \triangleright \text{initial partition} \)
2. \textbf{for} each region \( R \in P \text{ such that } \text{Pred}(R) \) \textbf{do} \[
(j, \theta) \leftarrow \text{argmin}_{(j, \theta)} \text{error}(R^-(j, \theta)) + \text{error}(R^+(j, \theta))
\]
3. \( P \leftarrow P - R \cup \{R^-(j, \theta), R^+(j, \theta)\} \)
4. \textbf{return} \( P \)
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: $F(T) = \text{Loss}(T) + \alpha |T|$, $\alpha \geq 0$ 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.
Outline

1. Introduction
2. Margin Bounds
3. Single classifier Approach
   - Multi-Class SVMs
   - AdaBoost
   - Decision Trees
4. Combination of binary classifiers Approach
   - 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: $h = \arg\max_{l' \in Y} |\{l : h_{ll'}(x) = 1\}|$.

- **Problem:**
  - computational: train $\frac{k(k-1)}{2}$ 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) $O(k^2-\alpha m^\alpha)$</td>
<td>smaller $N_{SV}$ per $B$</td>
</tr>
</tbody>
</table>

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

Technique:

- assign $F$-long binary code word to each class:

$$M = [M_{ij}] \in \{0, 1\}^{[1,k] \times [1,F]}.$$

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

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

$$h: x \mapsto \arg\min_{l \in \mathcal{Y}} d_{Hamming}(M_l, f(x))$$
Illustration

- 8 classes, code-length: 6.

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

new example $x$

\[
\begin{array}{ccccccc}
  f_1(x) & f_2(x) & f_3(x) & f_4(x) & f_5(x) & f_6(x) \\
  0 & 1 & 1 & 0 & 0 & 1 & 1 \\
\end{array}
\]
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 \(yf(x)\), 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:
    \[
    h(x) = \arg\min_{l \in \{1, \ldots, k\}} \sum_{j=1}^{F} L(M_{lj}f_j(x))
    \]
Ideas

- Continuous codes: real-valued matrix.
- Learn matrix code $M$.
- Similar optimization problems with other matrix norms.
- Kernel $K$ used for similarity between matrix row and prediction vector.
Continuous Codes

- Optimization problem: \((M_i \text{ } l\text{th row of } M)\)

\[
\min_{M, \xi} \|M\|_2^2 + C \sum_{i=1}^{m} \xi_i
\]

subject to : \(K(f(x_i), M_{y_i}) \geq K(f(x_i), M_l) + 1 - \xi_i\)

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

- Decision function:

\[
h : x \mapsto \arg\min_{l \in \{1, \ldots, k\}} K(f(x), M_l)
\]
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.