**AdaBoost**

**Task**: Build a strong classifier from a set of weak classifiers.

Input $x$, output $y \in \{-1, 1\}$

Set of weak classifiers $\langle \phi_m(x) : m = 1, \ldots, M \rangle$

Strong classifier $\text{sign}\left\{ \sum_{m=1}^{M} \alpha_m \phi_m(x) \right\}$

Intuition: weak classifiers are only $51\%$, $75\%$ of the time
strong classifier $- 99\%$ weights $\langle \alpha_m : m = 1\ldots M \rangle$ - learnt.
(2). **Task:** Input \( \{(x_i, y_i) : i = 1 \to N\} \)

- labeled examples - e.g. \[\boxed{\begin{array}{c}
\text{Face } y = 1 \\
\text{Non-Face } y = -1
\end{array}}\]

\( x \) is the image intensity values in an image region.

Set of weak classifiers - \( \{ \Phi_\mu() : \mu = 1 \to M \} \)

**Note:** for real problems, the set of weak classifiers is critical. AdaBoost can select the best combination, but this may not be good enough if the set is badly chosen.
(3) Mathematical Formulation.

Define: \( Z[\lambda] = \sum_{i=1}^{n} e^{-y_i \sum_{\mu=1}^{m} \lambda_{\mu} \phi_{\mu}(x_i)} \)

Initialize: \( \lambda = 0 \), \( i = 1 \)

At time \( t \), state: \( \lambda^t = (\lambda_1^t, \ldots, \lambda_m^t) \)

Minimize \( Z[\lambda] \) w.r.t. each component \( \lambda_{\mu} \) others fixed,

Solve: \( \frac{\partial}{\partial \lambda_{\mu}} Z[\lambda_1^t, \ldots, \lambda_{\mu} + \Delta \hat{\lambda}_{\mu}, \ldots, \lambda_m^t] = 0 \) for each \( \mu \).

Let \( \hat{\mu} = \arg \min_{\mu} Z[\lambda_1^t, \ldots, \hat{\lambda}_{\mu}, \ldots, \lambda_m^t] \)

Set: \( \lambda^t+1 = \lambda^t + \Delta \hat{\lambda} \), \( \gamma_{\mu}^t = \lambda_{\mu} \) \( \mu \neq \hat{\mu} \)

Select one "direction" update that direction. This is AdaBoost (basic)

Coordinate descent
(4) Why do this?

Loss function of strong classifier:  
\[ \mathbb{E}[\mathcal{E}] = \frac{N}{\sum_{i=1}^{N} \left( 1 - [y_i = \text{sign}(\sum_{\mu=1}^{M} \lambda_{\mu} \phi_{\mu}(x_i))] \right) } \]

Indication Function.

i.e. error = 1, if \( y_i \neq \text{sign}(\sum_{\mu=1}^{M} \lambda_{\mu} \phi_{\mu}(x_i)) \)

\[ Z[\mathcal{E}] \text{ is a convex upper bound of } \mathbb{E}[\mathcal{E}] \]

\[ \mathbb{E}[\mathcal{E}] \leq Z[\mathcal{E}] \]

AdaBoost minimizes \( Z[\mathcal{E}] \) w.r.t. \( \lambda \)

\( \text{make } \mathbb{E}[\mathcal{E}] \text{ small} \)
(5) \textbf{The Algorithm} \hspace{1cm} \text{Iterate over time } t.

For each weak classifier \( \phi_{\mu}(\cdot) \) divide the data into two sets.

\[ w^+_\mu = \{ i : y^i \phi_{\mu}(x^i) = 1 \} \hspace{1cm} w^-_\mu = \{ i : y^i \phi_{\mu}(x^i) = -1 \} \]

classifier is correct \hspace{1cm} \text{classifier is wrong}

When selecting/weighting classifiers to use we need to take into account the weights \( \lambda^t_\mu \) which we have already obtained.

To do this, we define weights:

\[ D_i^t = \frac{e^{-y_i \sum_{\mu} \lambda^t_\mu \phi_{\mu}(x^i)}}{\sum_i e^{-y_i \sum_{\mu} \lambda^t_\mu \phi_{\mu}(x^i)}} \]
For each weak classifier, compute:

\[ \Delta_{\mu}^t = \frac{1}{2} \log \frac{\sum_{i \in \text{w}_\mu^+} D_i^t}{\sum_{i \in \text{w}_\mu^-} D_i^t}, \quad \text{Note: if } D_i^t = \frac{1}{n}, \quad \forall i \]

Then solve \( \hat{\mu} = \arg \min \sqrt{\sum_{i \in \text{w}_\mu^+} D_i^t} \left\{ \sqrt{\sum_{i \in \text{w}_\mu^-} D_i^t} \right\} \)

Update: \( \lambda_{\hat{\mu}}^{t+1} = \lambda_{\hat{\mu}}^t + \Delta_{\mu}^t \), \( \lambda_{\mu}^{t+1} = \lambda_{\mu}^t \), \( \mu \neq \hat{\mu} \)

Key Idea: This corresponds to coordinate descent on \( \mathbb{R}^D \) (earlier pages)
(7) **Convergence.**

The algorithm converges until the error of all weak classifiers (with weights \( D \)) is 50%.

This is the global minimum of \( Z(\Omega) \). \( \Delta_n = 0 \) for all \( n \).

Convergence occurs when \( \sum_{i \in \omega^+} D^t_i = \sum_{i \in \omega^-} D^t_i = \frac{1}{2} \) when "weighted" error is zero for all classifiers.

But, better to stop earlier and cross-validate — test performance on other data — Test set.

Avoid overfitting the data (memorization).
Alternative viewpoint - Regression

Compare to regression

\[ P(y | x; \Lambda) = \frac{e^{-y \sum_{\mu=1}^{\Lambda} \lambda_{\mu} \phi_{\mu}(x)}}{e^{\sum_{\mu=1}^{\Lambda} \lambda_{\mu} \phi_{\mu}(x)} + e^{-\sum_{\mu=1}^{\Lambda} \lambda_{\mu} \phi_{\mu}(x)}} \]

Estimate \( \Lambda \) to maximize

\[ \prod_{i=1}^{N} P(y_i | x_i; \Lambda) \] Or better add

\[ \text{Sparsity} \ P(\Lambda) = e^{-\frac{\Lambda}{2}} \]

Results from this approach are as good (or better?) than AdaBoost. 

But more computation required.

K. Muller et al. Lebanon and Lafferty
(9) **Support Vector Machines**

classify a similar form \( \hat{y}(x) = \text{sign}(\lambda \cdot \Phi(x)) \)

\( \Phi(x) \) any function (i.e. not weak classifier).

Initially \( \Phi(x) = x, 1 \)

Intuition → separate data by hyperplane

require \( y_i (\alpha \cdot x_i + b) \geq 1 - \xi_i \)

\( \xi_i \) slack variable, width of margin \( \frac{1}{\alpha} \).

\( \xi_i \) allows us to move data which is misclassified (at a cost)
Minimize
\[ \frac{1}{2} \| \alpha \|^2 + \sum_{i=1}^{N} z_i \]

subject to
\[ y_i (\alpha, x_i + b) \geq 1 - z_i \quad \text{for all } z_i. \]

Intuition
- Make margin \( \frac{1}{\| \alpha \|} \) as big as possible while
- Keeping amount of slack variables small (more precisely as little as possible).

Intuition
- \( \frac{1}{2} \| \alpha \|^2 + \sum_{i=1}^{N} \max \{ 0, 1 - y_i (\alpha, x_i + b) \} \)
- Regularize hinge loss
- Convex upper bound of error.
(11) **Primal & Dual Formulation.**

Introduce Lagrange parameters \( \langle t_i \rangle, \langle w_i \rangle \)

Impose constraints: \( z_i > 0, \ y_i (a_i x_i + b) > 1 - z_i \)

**Primal:**

\[
\begin{align*}
\min & \quad \alpha, b, z, t, x \\
\text{w.r.t.} & \quad \alpha, b, z \\
\max & \quad t, x \\
\text{subject to} & \quad - \sum z_i t_i - \sum x_i \{ y_i (a_i x_i + b) - 1 + z_i \} \\
\end{align*}
\]

**Dual:**

Solve

\[
\frac{\partial L}{\partial \alpha} = 0, \quad \frac{\partial L}{\partial b} = 0, \quad \frac{\partial L}{\partial z_i} = 0 \quad \text{for} \quad \alpha, b, z
\]

Substitute back - obtain

\[
L_d (\lambda) = \sum_{\mu} \lambda_{\mu} M_{\mu, 0} + \frac{1}{2} \sum_{\mu} \lambda_{\mu} d_{\mu} y_{\mu, 0} y_{\mu, 0}
\]

with constraints \( 0 \leq \lambda_{\mu} \leq C, \quad \sum_{\mu} \lambda_{\mu} y_{\mu, 0} y_{\mu, 0} = 0 \).
Support Vectors:

\[ \frac{\partial L}{\partial \alpha} = 0 \implies \alpha = \sum_{i=1}^{n} \hat{z}_i y_i x_i. \]

\( \hat{z}_i \) obtained by maximizing \( L_d(x) \).

Now, theory of Lagrange multipliers implies that \( \hat{z}_i = 0 \), unless \( y_i (\alpha \cdot x_i + b) > 1 \).

\( \hat{z}_i = 0 \), only for points \( y_i x_i \) on the margin.

These are the support vectors.
Hence the final classifier is of the form

\[ \text{sign} \left( \hat{\alpha} \cdot \mathbf{x} + \hat{b} \right) = \text{sign} \left( \sum_{i=1}^{N} \hat{\alpha}_i y_i \mathbf{x}_i \cdot \mathbf{x} + \hat{b} \right) \]

Dependency on support vectors means that the algorithm only pays attention to the important data—the data near the decision boundary.

Also, motivates efficient algorithms to deal with very large datasets.
(14) **Kernel Trick.**

\[ \underbrace{x \rightarrow \Phi(x)} \]

**Result:** the final classifier depends only on the kernel \( K(x, x') = \Phi(x), \Phi(x') \).

No need to specify \( \Phi(x) \).

**Kernels:** \( K(x, x') = x \cdot x' \), or polynomials, radial basis functions \( K(x, x') = e^{-\frac{1}{2}||x - x'||^2} \).

**What kernels are allowed?** Mercer's Thm: \( K(x, x') = \sum_{x' \in \text{neighb}} \lambda_i \phi_i(x', y') \).

**Spectral Thm:** Matrices.
(15) \text{Classifier} \quad \text{Sign} \left( \sum_{i=1}^{N} \hat{z}_i y_i K(x_i, x) \right)

K(x, x') = x \cdot x' / \text{plane, Polynomial.}

\text{Radial Basis Function} \rightarrow \text{For RBF, the classifier is the weighted sum of the neighbors}

K(x, x') = \frac{1}{\sigma^2} \left(1 - \frac{|x - x'|^2}{\sigma^2} \right)

\text{Nearest Neighbor} \quad \text{fall off with distance.}

\text{?} \rightarrow +, \quad ?? \rightarrow -
Generalization to multiclass

Both classified are of form \( \text{sign} \langle \mathbf{w}, \phi(x) \rangle \)

\[
= \arg \max \{ \mathbf{w} \cdot \phi(x) y \} \quad \text{for } y \in \{\pm 1\}
= \arg \max \{ \mathbf{w} \cdot \phi(x) y \} \quad \text{with } \phi(x) y = \phi(x,y)
\]

Formally, define

\[
\phi(x,y) = \langle \phi_\mu(x,y) : \mu = 1, \ldots, M \rangle
\]

decision rule \( \hat{y}(x) = \arg \max_y \{ \mathbf{w} \cdot \phi(x,y) \} \)

convex upper bounds for error + regularize.
Detecting and Reading Text in Natural Scenes

Xiangrong Chen, Alan L. Yuille
{xrchen, Yuille}@stat.ucla.edu

Statistics dept, UCLA
Outline

- Background
- Overview of our method
- Detecting text
- Reading text
- Experiments
- Summary
Text detection methods

Text as texture

Text as connected component
## Comparison

<table>
<thead>
<tr>
<th>Text as Feature</th>
<th>Texture analysis</th>
<th>connected component</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Searching method</strong></td>
<td>Scan the image using a small window in different scales</td>
<td>Enumerate all the CCPS; need image segmentation to obtain the CCPs</td>
</tr>
<tr>
<td><strong>Pros</strong></td>
<td>Easy to deal with scale and complex background; scan quickly</td>
<td>Easily lead to generative model and thus can guide recognition task</td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td>Discriminant model; a black box, not easy to guide recognition task</td>
<td>No good enough segmentation algorithm available to get CCPs</td>
</tr>
</tbody>
</table>
Combination

- Find candidate area using text as texture
- Verify using text as connected component
Proposed method

Text as texture

AdaBoost for text detection → Adaptive binarization

OCR engine ← Connected components evaluation

Text as connected component
Why using AdaBoost

- Improves classification accuracy
- Can be used with many different classifiers
- Simple to implement
- Not prone to overfitting
Training data

- 162 Source images by normal and blind people
- Manually label text regions
- Cut the text regions into overlapped training samples with fixed width-to-height ratio, 2:1
Features – Criterion

- **Informative**
  - Invariant for text regions
  - Discriminating between text and non-text regions

- **Cost**
  - Computation
Features-Training samples

<table>
<thead>
<tr>
<th>Face</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Raw data</strong></td>
<td><img src="Image" alt="Images of faces" /> <img src="Image" alt="Images of text" /></td>
</tr>
<tr>
<td><strong>Align, Crop &amp; Scale</strong></td>
<td>4,000 faces 32 x 32</td>
</tr>
<tr>
<td><strong>PCA</strong></td>
<td>First 50 PCs capture 90% energy</td>
</tr>
<tr>
<td><strong>Features</strong></td>
<td><img src="Image" alt="Feature images" /></td>
</tr>
</tbody>
</table>
Features – Set I

- 1st order derivatives

Mean of $\frac{dI}{dx}$

Mean of $\frac{dI}{dy}$

$1/6$

$2/3$

$1/6$
Features – Set II

- Histogram of Intensity and gradient
Features – Set III

- Edge linking features

  edge map → thinning → linking

Using statistics of the length of the linked edges
Weak learners

- Ability of the strong classifier is determined by the ability of the weak learners
- Strong classifier with 1D stub weak learners can’t deal with the example

We use log-likelihood ratio test on distributions of both single features and pairs of features as weak learners (Konishi and Yuille, 2003)
An example of Weak learners

- Joint distribution of a pair of features form the first weak learner AdaBoost selected
Cascade of strong classifiers

Candidates

- \(\mu\) and \(\sigma\)
- Derivative features
- Derivative features
- All features

Results

Ruled out

- Images of text candidates
- Images of text ruled out
Text detection examples
Fail to detect

- Vertically aligned text
- Individual letters
- Extreme cases
Adaptive binarization

- Ni’Black’s method

\[ T_r(x) = \mu_r(x) + k \sigma_r(x) \]

- Determine range of neighborhood size
  - Relative to the sub-window height \( h \)

\[ r(x) = \min_{r \subset R(h)} \{ \sigma_r(x) > T_0 \} \]
OCR engine

- Currently we use a commercial OCR engine
- A generative model for reading text is under developing
Text reading examples

12 Folsom to Army

Administration

Cashier

Restrooms

CASHIER

Line 22

Maui

California

5046

SACRAMENTO

2500

STEINER

Admin

2300

Cashier

Restrooms

CASHIER

Line 22

Maui

California

5046

SACRAMENTO
False positives

- Building structures
- Signs or icons
- Tree leaves and branches
Results

➢ Accuracy
  - False Negative for detection 2.8%
  - False Positive for detection ~ 1/200,000
  - False Negative for reading 7%
  - False Positive for reading 10% (1% w/ constraint to form coherent word)

➢ Speed
  - 3 Seconds for 2,048*1536 image ~ 15fps for 320*240 video frames
Summary

- Using Adaboost to learn a strong classifier for detecting text in unconstrained scenes

- Selection of informative features with consideration of computation cost

- Detecting and reading over 90% text regions in our database

- Real-time (15fps) for video quality images (320 * 240)
ICDAR’s competition

- Database