AdaBoost

Note Title 8/1/2013

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

Input $\underline{x}$ output $y \in \{\pm 1\}$

Set of weak classifiers $\{\phi_\mu(\underline{x}) : \mu = 1, .., M\}$

$\phi_\mu(\underline{x}) \in \{\pm 1\}$

Strong classifier $\text{sign}\{\sum_{\mu=1}^{M} \lambda_\mu \phi_\mu(\underline{x})\}$.

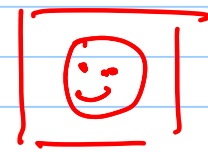
Intuition – weak classifiers are only 51%, 75% of the time

strong classifier – 99%

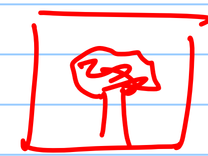
weights $\{\lambda_\mu : \mu = 1 \text{ to } M\}$ – learnt.

(2). **Task**: Input $\{(\underline{x}^i, y^i) : i = 1 \text{ to } N\}$

labelled examples – e.g.



Face $y = 1$



Non-Face $y = -1$

$\underline{x}$ is the image intensity values in an image region.

Set of weak classifiers – $\{\phi_\mu(\,) : \mu = 1 \text{ to } M\}$

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.

$\underline{\lambda} = (\lambda_1, \ldots, \lambda_M)$

Define. $Z[\underline{\lambda}] = \sum_{i=1}^{N} e^{-y_i \sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(\underline{x}_i)}$

Initialize $\underline{\lambda} = \underline{0}$,

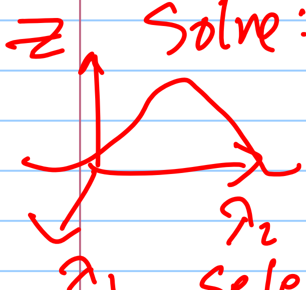
At time t, state $\underline{\lambda}^t = (\lambda_1^t, \ldots, \lambda_M^t)$

Minimize $Z[\underline{\lambda}]$ w.r.t. each component λ_μ others fixed,

Solve: $\frac{\partial}{\partial \lambda_\mu} Z[\lambda_1^t, \ldots, \lambda_\mu^t + \Delta_\mu^t, \ldots \lambda_M^t] = 0$ for each μ.

Let $\hat{\mu} = \arg\min_{\mu} Z[\lambda_1^t, \ldots, \lambda_\mu^t + \Delta_\mu^t, \ldots, \lambda_M^t]$

set $\lambda^{t+1}_{\hat{\mu}} = \lambda_{\hat{\mu}} + \Delta^t_{\hat{\mu}}$, $\lambda_\mu^{t+1} = \lambda_\mu^t$ $\mu \neq \hat{\mu}$



select one "direction", update that direction.

This is AdaBoost (basic)

| Coordinate descent |

(4) Why do this?

$$1 \le e^{\psi}, \text{ if } \psi \ge 0$$
$$0 \le e^{\psi}, \text{ if } \psi < 0$$

Loss function of strong classifier $\text{sign}\left(\sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x_i)\right)$

$$E[\lambda] = \sum_{i=1}^{N} \left\{ 1 - \mathbb{I}\left\{ y_i = \text{sign}\left(\sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x_i)\right)\right\}\right\}$$

($\mathbb{I}$: Indicator Function.)

i.e. error = 1, if $y_i \ne \text{sign}\left(\sum_{\mu=1}^{M} \lambda_\mu \varphi_\mu(x_i)\right)$

$Z[\lambda]$ is a convex upper bound of $E[\lambda]$

$$E[\lambda] \le Z[\lambda]$$

AdaBoost minimizes $Z[\lambda]$ w.r.t. λ

make $E[\lambda]$ small.

(5) The Algorithm

Iterate over time t.

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

$W_\mu^+ = \{ i : y^i \phi_\mu(x^i) = 1 \}$ classifier is correct, $W_\mu^- = \{ i : y^i \phi_\mu(x^i) = -1 \}$ classifier is wrong.

When selecting/weighting classifiers to use we need to take into account the weights $\{\lambda_\mu^t\}$ which we have already obtained.

To do this, we define weights.

$$D_i^t = \frac{e^{-y_i \sum_\mu \lambda_\mu^t \phi_\mu(\underline{x}_i)}}{\sum_j e^{-y_j \sum_\mu \lambda_\mu^t \phi_\mu(\underline{x}_j)}}$$

(6) For each weak classifier, ϕ_μ compute:

$$\Delta_\mu^t = \frac{1}{2} \log \frac{\sum_{i \in W_\mu^+} D_i^t}{\sum_{i \in W_\mu^-} D_i^t},$$

Note: if $D_i^t = 1/N$, $\forall i$ (at $t=0$)

then $\Delta_\mu^t = \frac{1}{2} \log \frac{|W_\mu^+|}{|W_\mu^-|}$.

Note $\Delta_\mu^t = 0$ if $|W_\mu^+| = |W_\mu^-|$

Then solve $\hat{\mu} = \arg\min \sqrt{\sum_{i \in W_\mu^+} D_i^t} \sqrt{\sum_{i \in W_\mu^-} D_i^t}$

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

Key Idea: This corresponds to coordinate descent on $Z[\vec{\lambda}]$ (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 $Z[\lambda]$. $\Delta_\mu^t = 0$, for all μ

Convergence occurs when $\sum_{i \in W_\mu^+} D_i^t = \sum_{i \in W_\mu^-} D_i^t = 1/2$ 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).

(8) Alternative viewpoint - Regression

Compare to regression

$$P(y \mid \underline{x}; \underline{\lambda}) = \frac{e^{-y \sum_{\mu=1}^{M} \lambda_\mu \phi_\mu(x)}}{e^{\sum_{\mu=1}^{M} \lambda_\mu \phi_\mu(\underline{x})} + e^{-\sum_{\mu=1}^{M} \lambda_\mu \phi_\mu(\underline{x})}}$$

Estimate $\underline{\lambda}$ to maximize $\prod_{i=1}^{N} P(y_i \mid \underline{x}_i; \underline{\lambda})$. Or better add sparsity $P(\underline{\lambda}) = \frac{1}{Z} e^{-|\underline{\lambda}|}$

Results from this approach are as good (or better?) than AdaBoost.
But more computation required.

K. Muller et al
Lebanon and Lafferty

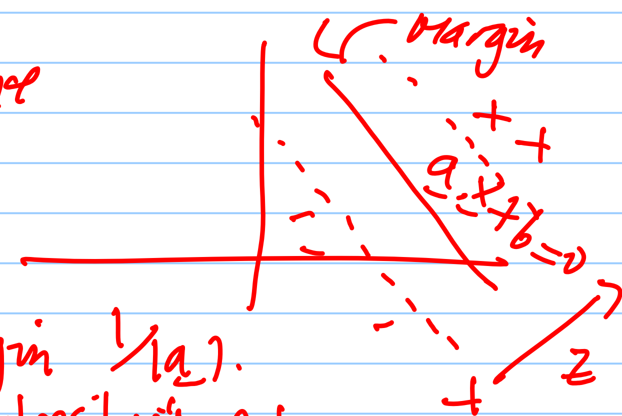
(9) Support Vector Machines

Classify of similar form $\hat{y}(\underline{x}) = \text{sign}(\underline{\lambda} \cdot \underline{\varphi}(\underline{x}))$

$\underline{\varphi}(\underline{x})$ any function (ie. not weak classifier).

Initially $\underline{\varphi}(\underline{x}) = \underline{x}, 1$

Intuition → seperate data by hyperplane



require $y_i(\underline{a} \cdot \underline{x}_i + b) \geq 1 - z_i$

z_i slack variable, width of margin $1/|\underline{a}|$.

z_i allows us to move data which is missclassified (at a cost)

(10) <u>Minimize</u> $\frac{1}{2}|\underline{a}|^2 + C\sum_{i=1}^{N} z_i$ $z_i \geq 0$

s.t. $y_i(\underline{a}.\underline{x}_i + b) \geq 1 - z_i$ for all z_i.

<u>Intuition</u> → make margin $\frac{1}{|\underline{a}|}$ as big as possible while keeping amount of slack variables small. (more points as little as possible.)

<u>Intuition</u>. $\frac{1}{2}|\underline{a}|^2 + C\sum_{i=1}^{N} \max\{0, 1 - y_i(\underline{a}.\underline{x}_i + b)\}$

regularizer ↗ Hinge loss ↗ convex upper bound of error.

(11) Primal & Dual Formulation.

Introduce Lagrange parameters $\{\tau_i\}$, $\{\alpha_i\}$

impose constraints $z_i \geq 0$, $y_i(\underline{a}\cdot\underline{x}_i + b) \geq 1 - z_i$

Primal

$$L_p(\underline{a}, b, z; \tau, \alpha) = \frac{1}{2}|\underline{a}|^2 + C\sum_{i=1}^{N} z_i$$

min w.r.t. $\underline{a}, b, z$

max w.r.t. τ, α

$$- \sum_i \tau_i z_i - \sum_i \alpha_i \{ y_i(\underline{a}\cdot\underline{x}_i + b) - 1 + z_i \}$$

Dual: solve $\frac{\partial L_p}{\partial \underline{a}} = 0$, $\frac{\partial L_p}{\partial b} = 0$, $\frac{\partial L_p}{\partial z_i} = 0$ for $\underline{a}, b, z$

Substitute back - obtain $L_d(\alpha) = \sum_\mu \alpha_\mu - \frac{1}{2}\sum_{\mu,\nu} \alpha_\mu \alpha_\nu y_\mu y_\nu \underline{x}_\mu \cdot \underline{x}_\nu$

with constraints $0 \leq \alpha_\mu \leq C$, $\sum_\mu \alpha_\mu y_\mu = 0$.

(12) <u>Support Vectors</u>:

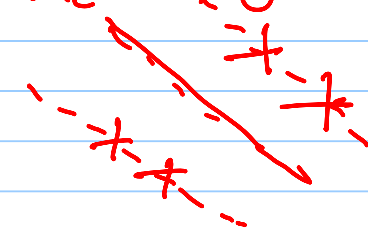
$$\frac{\partial L_p}{\partial \underline{a}} = 0 \quad \Rightarrow \quad \hat{\underline{a}} = \sum_{i=1}^{N} \hat{\alpha}_i \, y_i \, \underline{x}_i .$$

$\hat{\alpha}_i$ obtained by maximizing $L_d(\alpha)$

<u>Now</u> theory of lagrange multipliers implies that $\hat{\alpha}_i = 0$, unless $y_i(\underline{a} \cdot \underline{x}_i + b) > 1$.

$\hat{\alpha}_i = 0$, only for point $y_i, \underline{x}_i$ on the margin

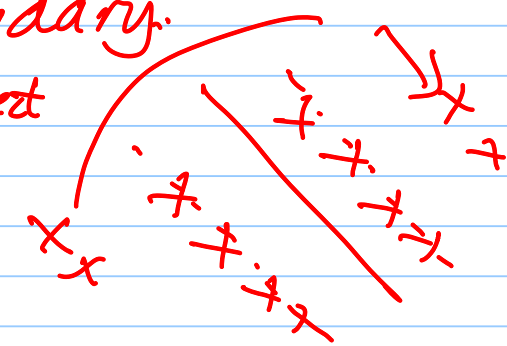
These are the <u>support vectors</u>.



(13) Hence the final classifier is of form

$$\text{sign}(\hat{\underline{a}}\cdot\underline{x}+\hat{b}) = \text{sign}\left\{\sum_{i=1}^{N}\hat{\alpha}_i y_i \underline{x}_i\cdot\underline{x}+\hat{b}\right\},$$

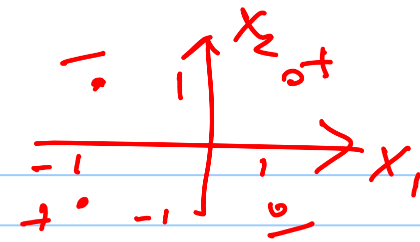
Dependence on support vectors means ~~that~~ the algorithm only pays attention to ~~the~~ important data — the data near the decision boundary.



Also, motivates efficient algorithms to deal with very large datasets.

(14)

Kernel Trick.



$x_1 x_2$

Extend $\underline{x} \to \underline{\varphi}(\underline{x})$

Result – the final classifier depends only on the kernel $K(\underline{x},\underline{x}') = \underline{\varphi}(\underline{x})\cdot\underline{\varphi}(\underline{x}')$

No need to specify $\underline{\varphi}(\underline{x})$.

kernels $K(\underline{x},\underline{x}') = \underline{x}\cdot\underline{x}'$, or polynomials

radial basis functions $K(\underline{x},\underline{x}') = e^{-\|\underline{x}-\underline{x}'\|^2}$ nearest neighbor.

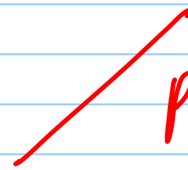
Spectral Thm **Matrices**

What. Kernels are allowed? **Mercer's Thm** $K(\underline{x},\underline{x}') = \sum \lambda_i \underline{\varphi}_i(\underline{x})\cdot\underline{\varphi}_i(\underline{x}')$

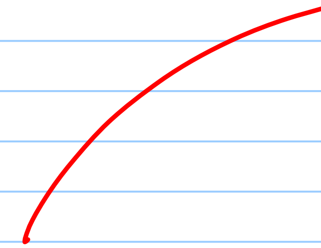
λ' +ve

(15) Classifier $\quad \text{sign}\left(\sum_{i=1}^{N} \hat{\alpha}_i \, y_i \, K(\underline{x}, \underline{x}^i) \right)$

$K(\underline{x}, \underline{x}') = \underline{x} \cdot \underline{x}'$

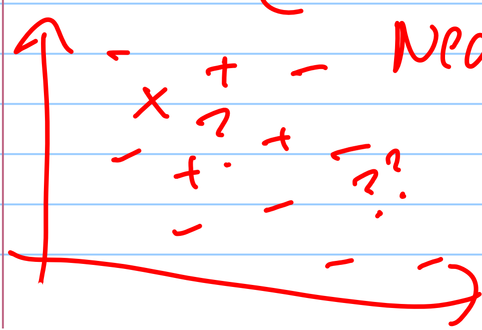


plane, Polynomial.



Radial Basis Function → For RBF, the classifier is the weighted sum of the neighbors fall off with distance.

$K(\underline{x}, \underline{x}') = e^{-|\underline{x} - \underline{x}'|^2}$



Nearest Neighbour

? → + , ?? → -

(15) ## Generalization to multiclass

Both classifiers are of form $\text{sign}\{\underline{\lambda} \cdot \underline{\varphi}(\underline{x})\}$

$$= \arg\max_{y \in \{\pm 1\}} \{\underline{\lambda} \cdot \underline{\varphi}(\underline{x}) y\}$$

$$= \arg\max_{y \in \{\pm 1\}} \{\underline{\lambda} \cdot \underline{\varphi}(\underline{x}, y)\}$$

with $\underline{\varphi}(\underline{x}, y) = \underline{\varphi}(\underline{x}) y$

Formally, define

$$\underline{\varphi}(\underline{x}, \underline{y}) = \{\varphi_\mu(\underline{x}, \underline{y}) : \mu = 1 \text{ to } M\}$$

class of features.

decision rule $\hat{y}(\underline{x}) = \arg\max_y (\underline{\lambda} \cdot \underline{\varphi}(\underline{x}, y))$

Convex upper bounds for errors + regularization.

Detecting and Reading Text in Natural Scenes

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

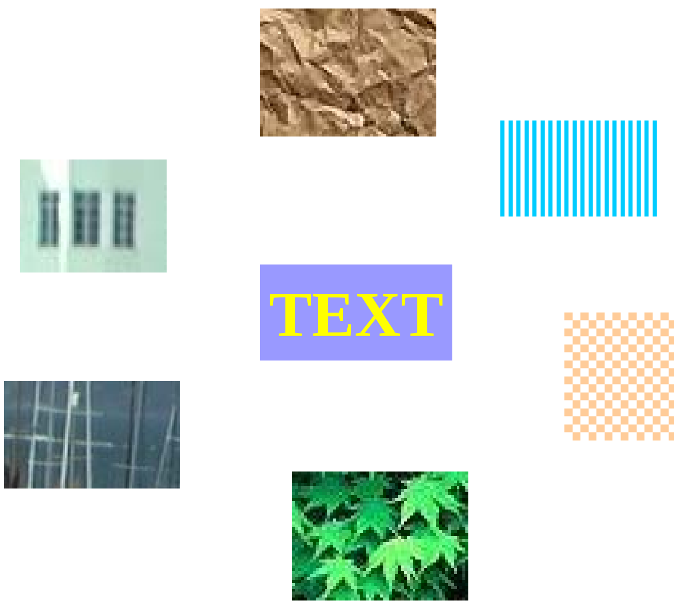
Statistics dept, UCLA

Outline

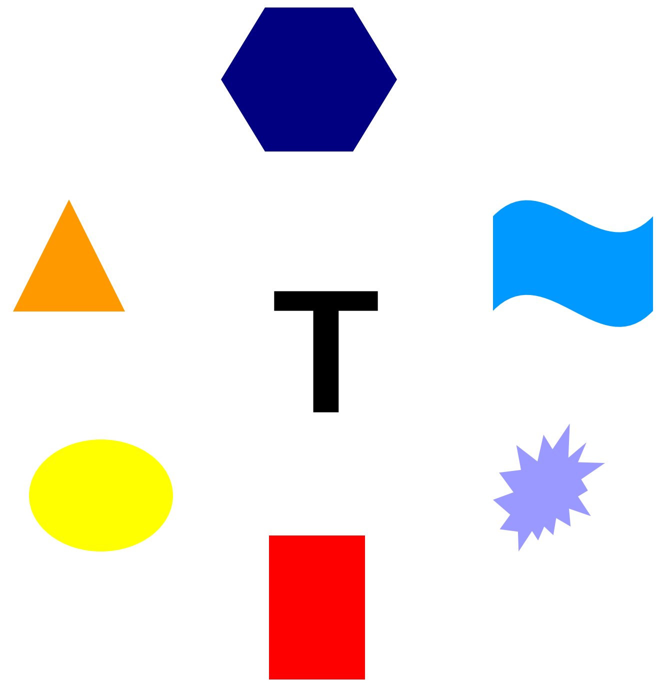
- Background
- Overview of our method
- Detecting text
- Reading text
- Experiments
- Summary

Text detection methods

Text as texture



Text as connected component



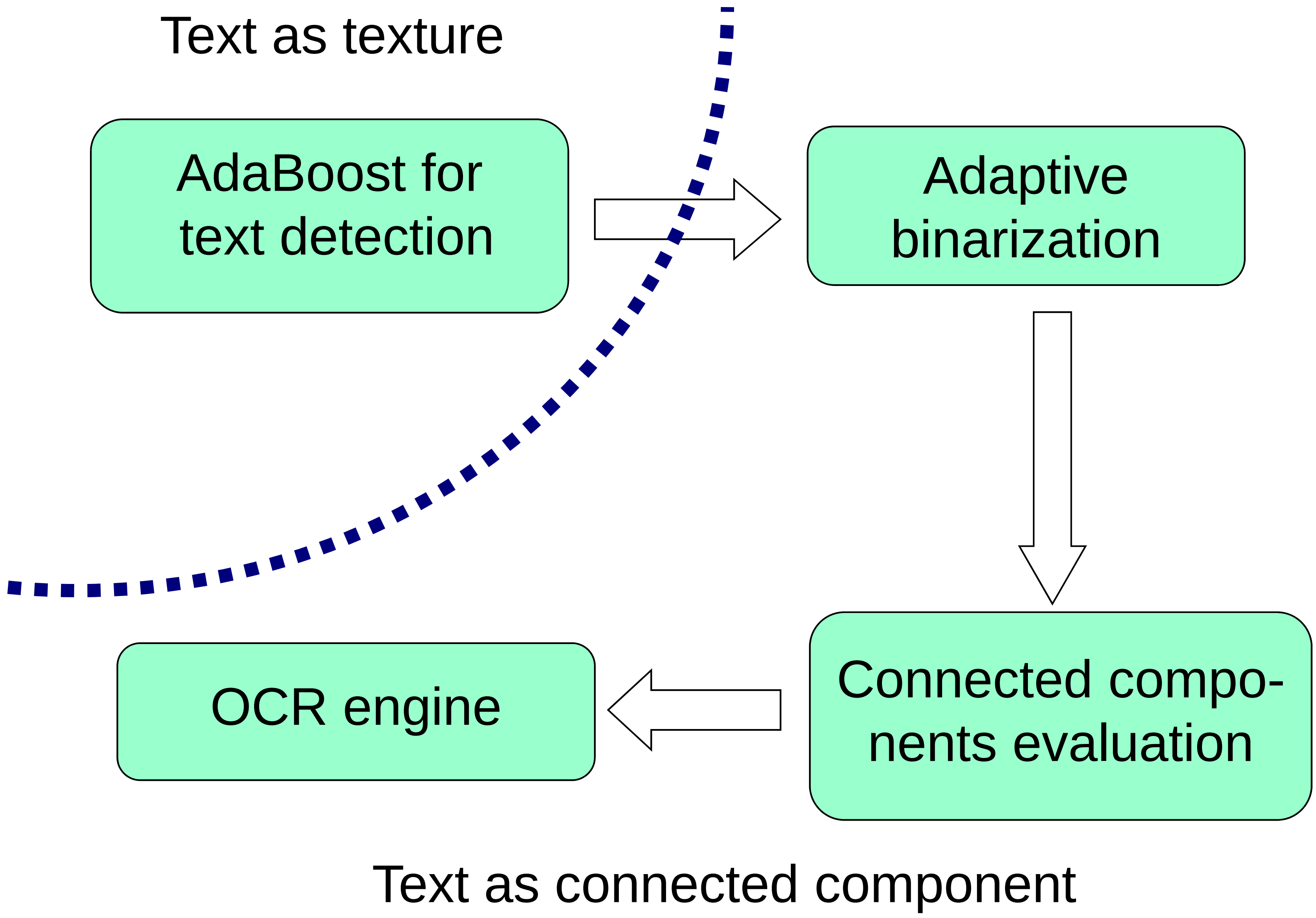
Comparison

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

Combination

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

Proposed method

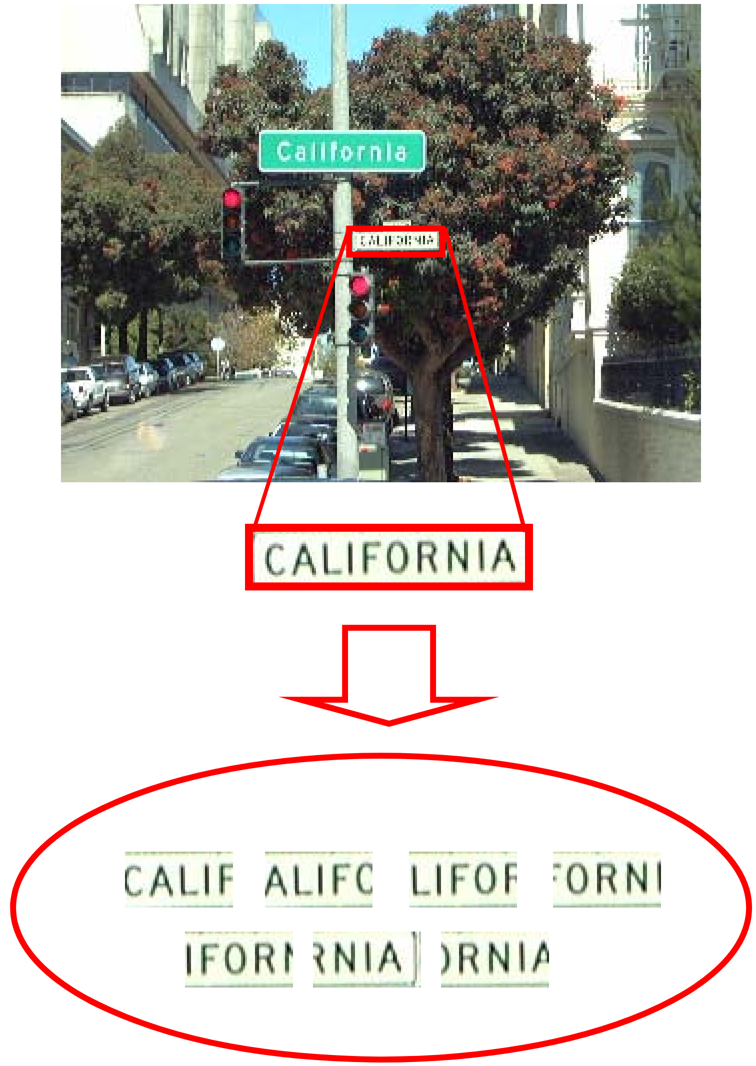


Why using AdaBoost

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

Training data

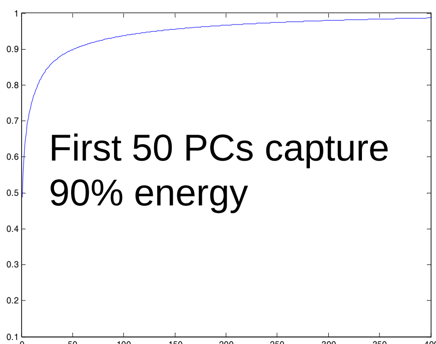
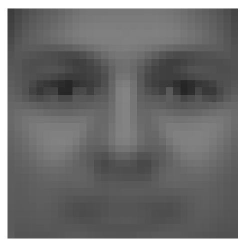
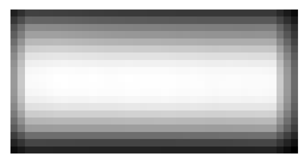
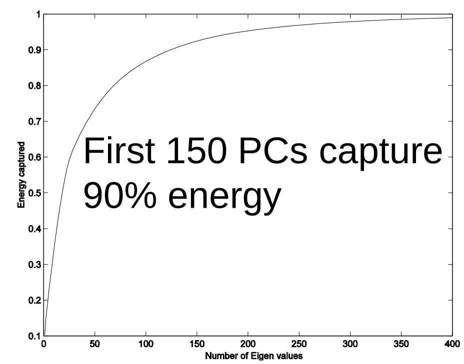
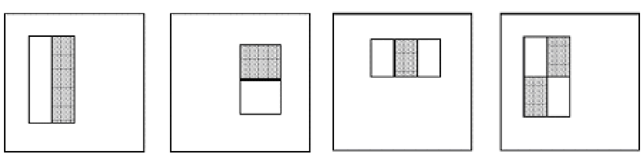
- 162 Source images by normal and blind people
- Manually label text regions
- Cut the text regions into overlapped training samples with fixed width-to-height ratio, 2:1



Features – Criterion

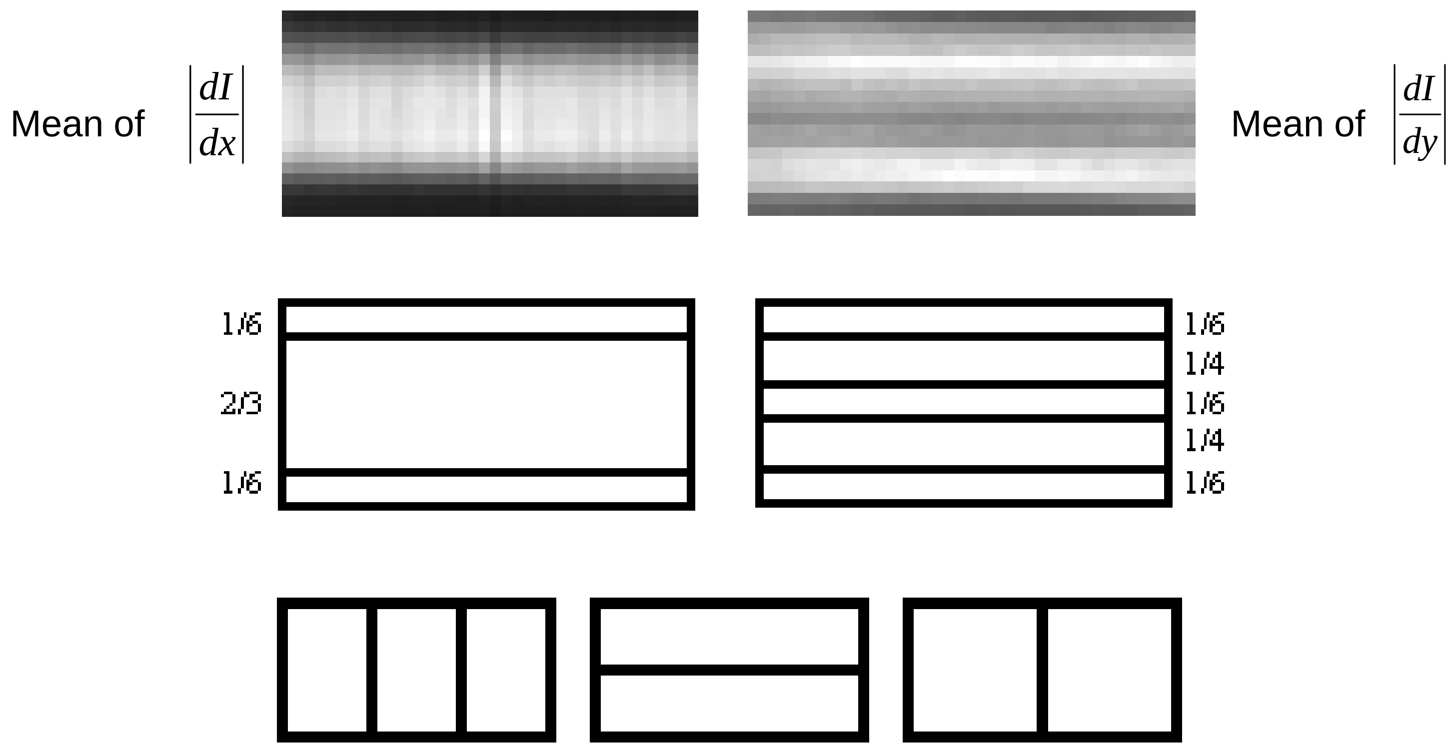
- Informative
  - Invariant for text regions
  - Discriminating between text and non-text regions
- Cost
  - Computation

Features-Training samples

	Face	Text
Raw data		
Align, Crop & Scale	 4,000 faces 32×32	4,000 patches 20×40 
PCA	 First 50 PCs capture 90% energy Mean face 	Mean patch   First 150 PCs capture 90% energy
Features		?

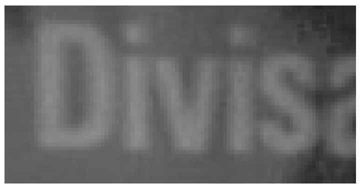
Features – Set I

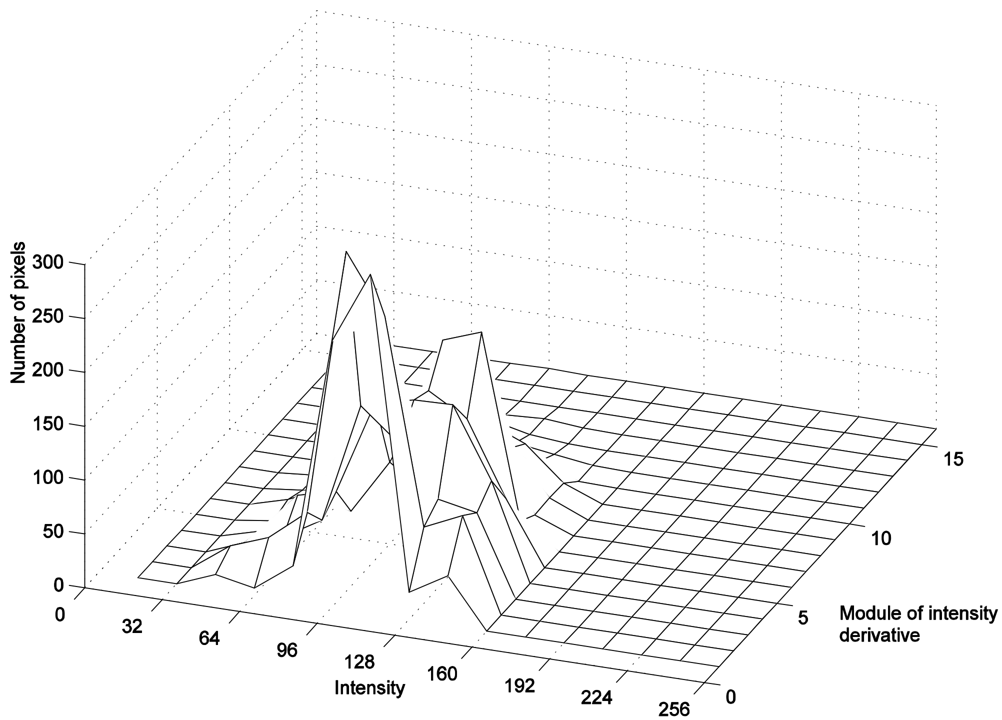
- 1st order derivatives

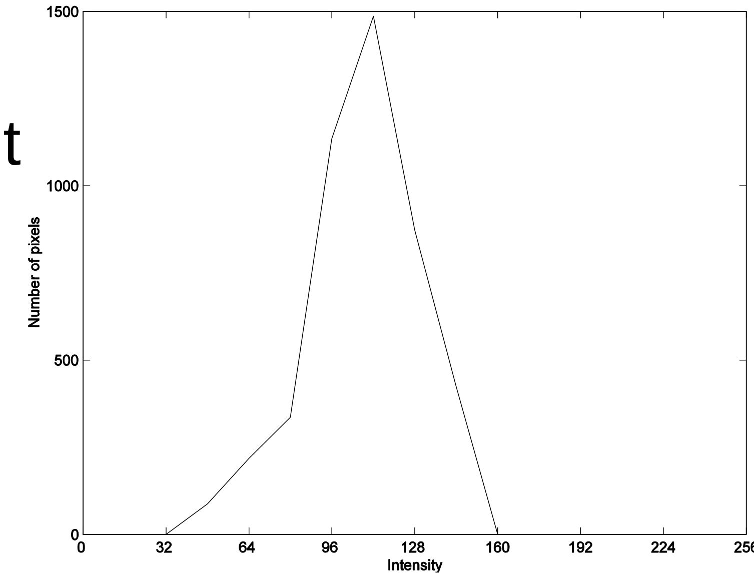


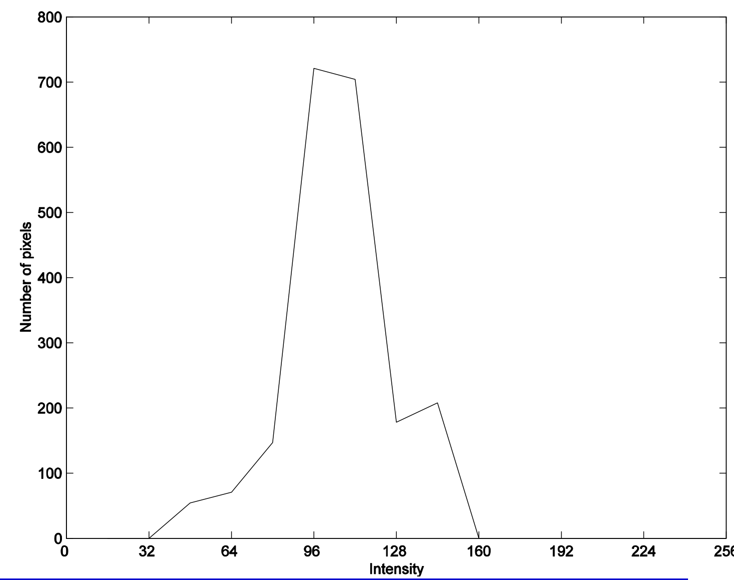
Features – Set II

- Histogram of Intensity and gradient









Features – Set III

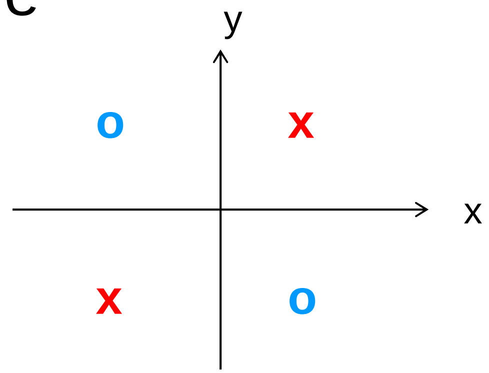
- Edge linking features

  edge map → thinning → linking

  Using statistics of the length of the linked edges

Weak learners

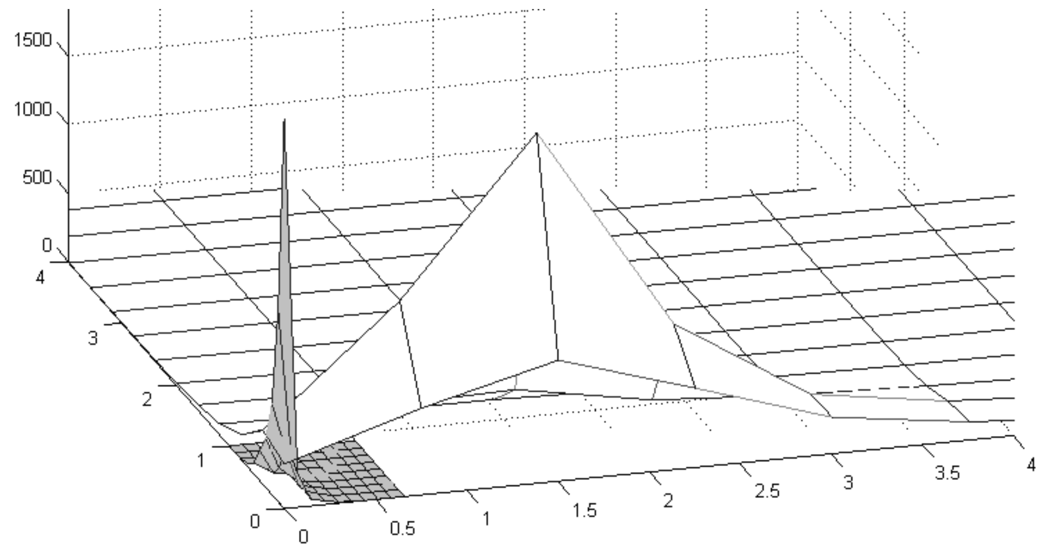
- Ability of the strong classifier is determined by the ability of the weak learners
- Strong classifier with 1D stub weak learners can't deal with the example



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

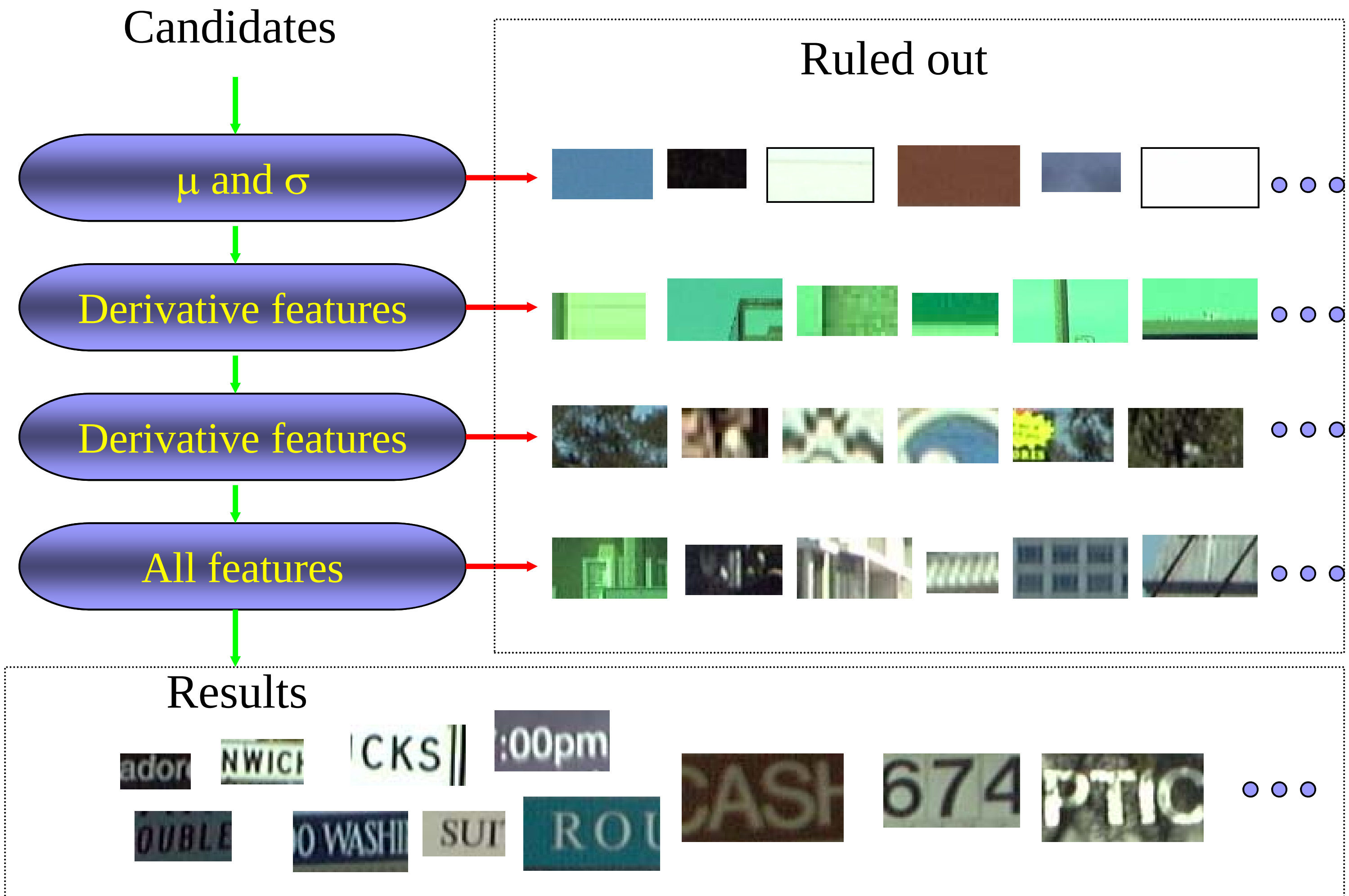
An example of Weak learners

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

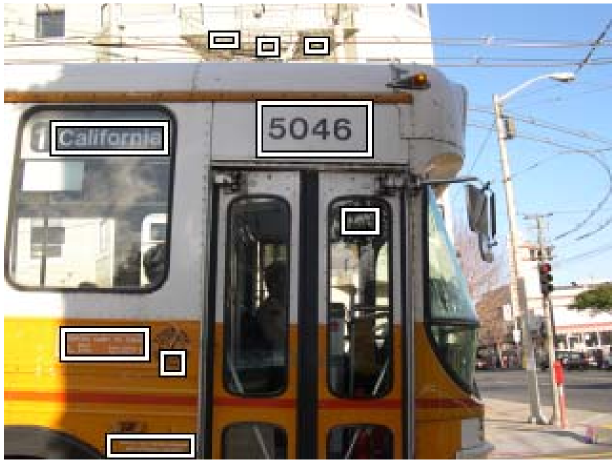


Text distribution is shaded.

Cascade of strong classifiers

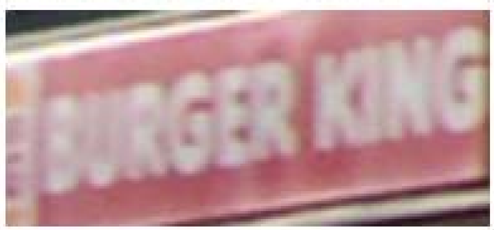


Text detection examples



Fail to detect

- Vertically aligned text
- Individual letters
- Extreme cases



Adaptive binarization

- Ni'Black's method

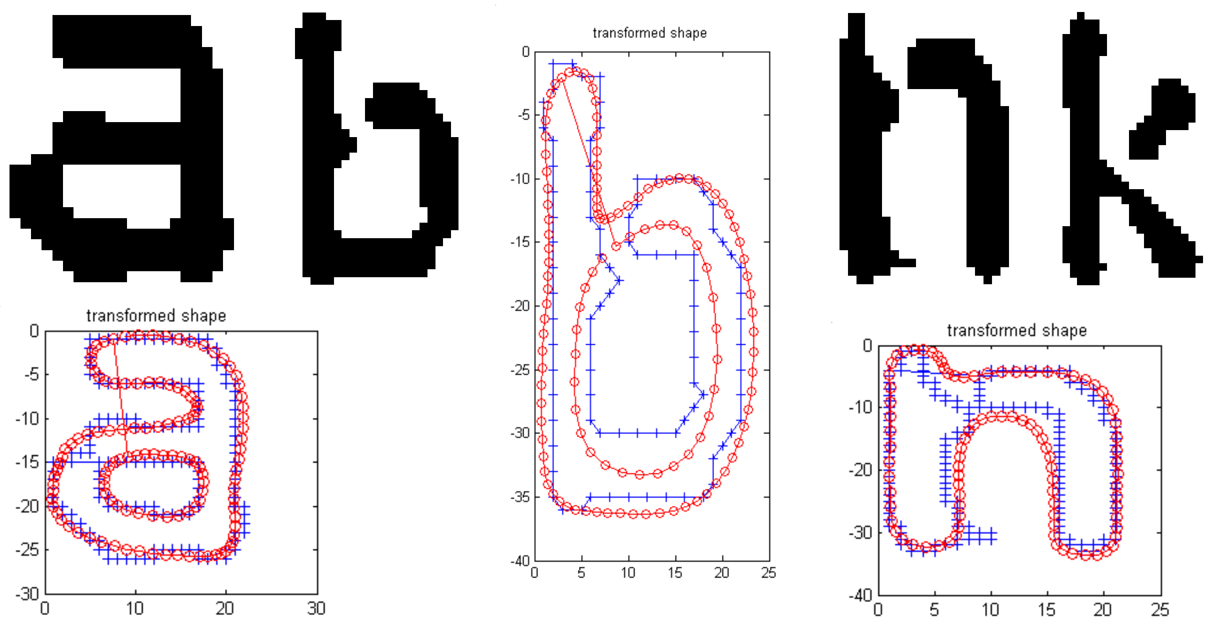
$$T_r(x) = \mu_r(x) + k \cdot \sigma_r(x)$$

- 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

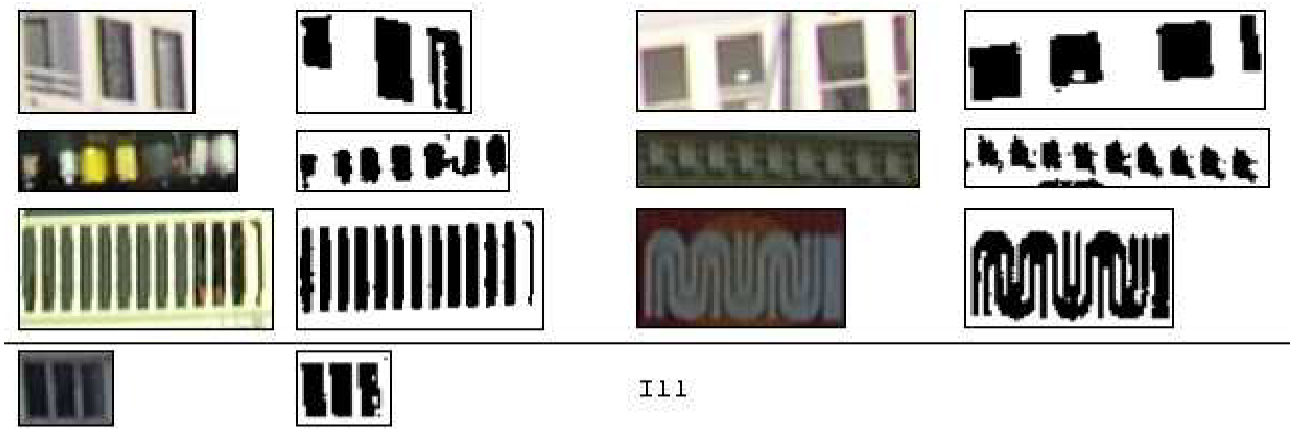
- Currently we use a commercial OCR engine
- A generative model for reading text is under developing



Text reading examples

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