Discriminative Learning via Semidefinite Probabilistic Models

K. Crammer & A. Globerson, UAI’06

Presented by Kevin Duh
UW Graphical Models Reading Group
Feb 14, 2007
Motivation

• Combine max margin & probablistic outputs:
  • Max margin (e.g. SVM)
    → good generalization
  • Probabilistic output (e.g. logistic regression)
    → confidence estimation, system integration

• Note: prob-output SVM do exist
  • Wahba, Vapnik, Hastie & Tibshirani, Platt, etc.

• Additional motivation:
  • Higher entropy probability outputs
Outline

• Probabilistic Model
• Learning
  • Max margin and max likelihood objective
  • Loss functions
• Semidefinite Programming
  • Overview
  • Implementation
• Results
Probabilistic Model (1/2)

- Assume: classes reside in linear subspace
  - \( \mathbf{x}^d : \mathbb{R}^d \rightarrow \) feature vector
  - \( y = \{1, \ldots, k\} \rightarrow k \) classes
  - \( S_i : \) subspace of class \( y \)
    - \( S_i \) and \( S_j \) are orthogonal for \( i \neq j \)
    - \( \{S_i\}_{1:k} \) span \( \mathbb{R}^d \)
- Projection operator \( A_j \) onto \( S_j \)
  - \( A_j \) is symmetric idempotent (\( A_j^2 = A_j, A_j^T = A_j \))
  - \( ||A_j\mathbf{x}||^2 \) is a natural distance measure of \( \mathbf{x} \) to class \( y \)
  - \( ||A_j\mathbf{x}||^2 = \mathbf{x}^T A_j^T A_j \mathbf{x} = \mathbf{x}^T A_j A_j \mathbf{x} = \mathbf{x}^T A_j^2 \mathbf{x} = \mathbf{x}^T A_j \mathbf{x} \)
Idempotent matrices

- Eigenvalues of $A$ are $\{0,1\}$
- $A$ is idempotent $\rightarrow$ $(I-A)$ is idempotent
  - Note $A$ and $(I-A)$ is also orthogonal
- Examples:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
Ay = \begin{pmatrix}
\frac{1}{n} & \cdots & \frac{1}{n} \\
\cdots & \cdots & \cdots \\
\frac{1}{n} & \cdots & \frac{1}{n}
\end{pmatrix}
\begin{pmatrix}
y_1 \\
y_2 \\
y_n
\end{pmatrix}
= \begin{pmatrix}
y \\
y \\
y
\end{pmatrix}.
\]

\[
(I - A)z = \begin{pmatrix}
1 - \frac{1}{n} & \cdots & -\frac{1}{n} \\
\cdots & \cdots & \cdots \\
-\frac{1}{n} & \cdots & 1 - \frac{1}{n}
\end{pmatrix}
\begin{pmatrix}
z_1 \\
z_2 \\
z_n
\end{pmatrix}
= \begin{pmatrix}
z_1 - \overline{z} \\
z_2 - \overline{z} \\
z_n - \overline{z}
\end{pmatrix}.
\]
Probabilistic Model (2/2)

Original multi-class probabilistic model:

\[ p(y|x) = \frac{1}{x^T (\sum_y A_y)x} x^T A_y x \]

But: \[ \|x\|^2 = 1 \quad \sum_y A_y = I \]

So:

\[ p(y|x) = x^T A_y x \]

Furthermore: relax \( A \) to be positive-semidefinite

\[ \lambda \in \{0, 1\} \rightarrow \lambda \in [0, 1] \]
Learning: Margin-based

• Setup: Given training data, learn set of A’s to maximize margin
  \[ m_i = p(y_i|x_i) - \max_{z \neq y_i} p(z|x_i) \]

• Objective:

\[
\begin{align*}
\max & \quad \eta - \beta \sum_i \xi_i \\
\text{s.t.} & \quad p(y_i|x_i) - p(z|x_i) \geq \eta - \xi_i \quad \forall i, \quad z \neq y_i \\
& \quad \sum_y A_y = I \\
& \quad A_y \geq 0, \quad \xi_i \geq 0
\end{align*}
\]
Learning: Likelihood-based

**Maximum likelihood:**

\[
\text{max} \sum_i \log p(y_i|x_i) \\
\text{s.t.} \sum_y A_y = I \\
A_y \geq 0
\]

- \( p(y|x) \) is linear in \( A \), \( \log \) is concave
- objective is non-linear (non-standard SDP)

**Optimal Bayes Loss:**

\[
\text{max} \sum_i p(y_i|x_i) \\
\text{s.t.} \sum_y A_y = I \\
A_y \geq 0
\]

- optimal Bayes loss given that \( p(y|x) \) is true distribution
Solution of Optimal Bayes Loss (binary case)

- \( A_1 \) and \( A_2 = I - A_1 \) are the projection matrices

- Objective:
  \[
  \max_{s.t} \sum_i p(y_i | x_i) \sum_y A_y = I \quad A_y \geq 0
  \rightarrow \sum_{i:y_i=1} \text{tr}(A_1 x_i x_i^T) + \sum_{i:y_i=2} \text{tr}((I - A_1) x_i x_i^T)
  \rightarrow \text{tr} \left( A_1 \left( \sum_{i:y_i=1} x_i x_i^T - \sum_{i:y_i=2} x_i x_i^T \right) \right) = \sum_i \lambda_i d_i
  \]

- Solution:
  - Compute covariance difference and its eigenvalues
  - Assign eigenvectors to \( A_1 \) depending on sign of eigenvalue
Convex Bounds on 0-1 Loss

\[
l_{\text{ML}}(x, y, p) = -\log_2(p(y|x))
\]
\[
l_{\text{Bayes}}(x, y, p) = 2(1 - p(y|x))
\]
\[
l_{\text{Marg}}(x, y, p, \eta) = \max\{0, 1 + \frac{1}{\eta} - \frac{2}{\eta}p(y|x)\}
\]

Smaller margin \(\rightarrow\) steeper hinge
Semidefinite Programming

• Max margin objective here:

\[
\begin{align*}
\max & \quad \eta - \beta \sum_i \xi_i \\
\text{s.t.} & \quad p(y_i|\mathbf{x}_i) - p(z|\mathbf{x}_i) \geq \eta - \xi_i \quad \forall i, \quad z \neq y_i \\
& \quad \sum_y A_y = I \\
& \quad A_y \succeq 0, \xi_i \geq 0
\end{align*}
\]

• SDP: linear objective, convex constraint
  • Generalization of linear & quadratic programming…
  • Linear programming w/ infinite constraints

Alternative implementation: Projected sub-gradient algo (1/3)

• Rather than using SDP solver…
• Re-write max margin objective:

\[
\begin{align*}
\max & \quad \eta - \beta \sum_i \left[ \eta - p(y_i|x_i) + \max_{z \neq y_i} p(z|x_i) \right]_+ \\
\text{s.t} & \quad \sum_y A_y = I \\
& \quad A_y \geq 0
\end{align*}
\]

• This is not differentiable, so use sub-gradient method:

\textbf{subgradient method} is simple algorithm to minimize nondifferentiable convex function \( f \)

\[
x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}
\]

• \( x^{(k)} \) is the \( k \)th iterate
• \( g^{(k)} \) is any subgradient of \( f \) at \( x^{(k)} \)
• \( \alpha_k > 0 \) is the \( k \)th step size

Reference: http://www.stanford.edu/class/ee364b/
Alternative implementation: Projected sub-gradient algo (2/3)

recall basic inequality for convex differentiable $f$:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

- first-order approximation of $f$ at $x$ is global underestimator

$g$ is a subgradient of $f$ (not necessarily convex) at $x$ if

$$f(y) \geq f(x) + g^T (y - x) \quad \text{for all $y$}$$
Alternative implementation: Projected sub-gradient algo (3/3)

Projected subgradient method is given by

\[ x^{(k+1)} = P(x^{(k)} - \alpha_k g^{(k)}) , \]

- Key: projection onto constraint set doesn’t increase distance to optimal \( x \)

\[
S_{\text{norm}} = \{ A_y : \sum_y A_y = I \}
\]

\[
S_{\text{pos}} = \{ A_y : A_y \geq 0 \}
\]

\[
S = S_{\text{norm}} \cap S_{\text{pos}}
\]

- For binary class case, this projection can be found analytically
- For multiclass, use Dykstra’s alternating projection algorithm
Relation to 2\textsuperscript{nd} order kernel method

- \( x^TA_yx = \text{tr}(A_yxx^T) \)
  - Dot product between A and xx': second order kernel
  - This paper’s method is more constrained (due to A)
- Example: A is diagonal

\[
p(y \mid x) = x^TA_yx = \text{tr}(A_yx^Tx) = \sum_i \text{diag}_i(A_y) * x_i^2
\]

- Diag(A) is bounded [0,1], which implies that weight vectors have box constrains

\[
p(y \mid x) = \sum_i w_i x_i^2, w \in [0,1]
\]
Experiments on USPS digit recognition

- Compare: SVM, Bayes, MaxMargin
- 45 binary problems:
  - 300 training samples each, validation performed
- Result: MaxMargin > SVM > Bayes
Analysis: MaxMargin vs. Bayes

Figure 4: Fraction of examples in test set which the difference in probability $|p(3|\mathbf{x}) - p(5|\mathbf{x})|$ is below a threshold set by a value in the x-axis.
Discussion

• What does it mean to have well-calibrated probability outputs?
  • Is there a way to judge which calibration is better (e.g. SVM+Platt scaling vs. this paper)

• SVM vs. MaxMargin in this paper
  • MaxMargin is more constrained problem
    → lower generalization error variance?
  • Parameterized hinge loss → Is this useful in general?

• What problems fit the class-comes-from-linear subspace assumption?
  • Can we apply more kernels?
  • This doesn’t seem hard to implement (at least for binary case). Which of our problems might benefit from it?