Training algorithms for Structured Learning

Sunita Sarawagi
IIT Bombay
http://www.cse.iitb.ac.in/~sunita
Training

Given

- $N$ input output pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$
- Features $f(x, y) = \sum_c f(x, y_c, c)$
- Error of output: $E(y_i, y)$
  - (Use short form: $E(y_i, y) = E_i(y)$)
  - Also decomposes over smaller parts: $E_i(y) = \sum_{c \in C} E_{i,c}(y_c)$

Find $w$

- Small training error
- Generalizes to unseen instances
- Efficient for structured models
Outline

1. Likelihood based Training

2. Max-margin training
   - Decomposition-based approaches
   - Cutting-plane approaches
Probability distribution from scores

- Convert scores into a probability distribution

\[
\Pr(y|x) = \frac{1}{Z_w(x)} \exp(w.f(x, y))
\]

where \( Z_w(x) = \sum_{y'} \exp(w.f(x, y')) \)

These are called **Conditional Random Fields (CRFs)**.
Probability distribution from scores

- Convert scores into a probability distribution

\[
Pr(y|x) = \frac{1}{Z_w(x)} \exp(w.f(x, y))
\]

where \( Z_w(x) = \sum_{y'} \exp(w.f(x, y')) \)

- When \( y \) vector of variables, say \( y_1, \ldots, y_n \), and decomposition parts \( c \) are subsets of variables we get a graphical model.

\[
Pr(y|x) = \frac{1}{Z_w(x)} \exp(\sum_c w.f(x, y_c, c)) = \frac{1}{Z} \prod_c \psi_c(y_c)
\]

with clique potential \( \psi_c(y_c) = \exp(w.f(x, y_c, c)) \)
Probability distribution from scores

- Convert scores into a probability distribution

\[
Pr(y|x) = \frac{1}{Z_w(x)} \exp(w.f(x, y))
\]

where \( Z_w(x) = \sum_y \exp(w.f(x, y')) \)

- When \( y \) vector of variables, say \( y_1, \ldots, y_n \), and decomposition parts \( c \) are subsets of variables we get a graphical model.

\[
Pr(y|x) = \frac{1}{Z_w(x)} \exp(\sum_c w.f(x, y_c, c)) = \frac{1}{Z} \prod_c \psi_c(y_c)
\]

with clique potential \( \psi_c(y_c) = \exp(w.f(x, y_c, c)) \)

- These are called **Conditional Random Fields (CRFs)**.
Training via gradient descent

\[ L(w) = \sum_{\ell} \log \Pr(y_\ell| x_\ell, w) = \sum_{\ell} (w \cdot f(x_\ell, y_\ell) - \log Z_w(x_\ell)) \]

Add a regularizer to prevent over-fitting.

\[ \max_w \sum_{\ell} (w \cdot f(x_\ell, y_\ell) - \log Z_w(x_\ell)) - ||w||^2 / C \]

Concave in \( w \) \( \implies \) gradient descent methods will work.

Gradient:

\[ \nabla L(w) = \sum_{\ell} f(x_\ell, y_\ell) - \sum_{y'} \frac{f(y', x_\ell) \exp w \cdot f(x_\ell, y')}{Z_w(x_\ell)} - 2w / C \]

\[ = \sum_{\ell} f(x_\ell, y_\ell) - E_{Pr(y'|w)} f(x_\ell, y') - 2w / C \]
Training algorithm

1. Initialize $w^0 = 0$
Training algorithm

1. Initialize $w^0 = 0$
2. for $t = 1 \ldots T$ do
3. \hspace{1em} for $\ell = 1 \ldots N$ do
4. \hspace{2em} $g_{k,\ell} = f_k(x_\ell, y_\ell) - E_{\Pr(y'|w)} f_k(x_\ell, y')$ \hspace{1em} $k = 1 \ldots K$
5. \hspace{1em} end for
6. \hspace{1em} $g_k = \sum_\ell g_{k,\ell} \hspace{1em} k = 1 \ldots K$

Running time of the algorithm is $O(INn(m^2 + K))$ where $I$ is the total number of iterations.
Training algorithm

1. Initialize $w^0 = 0$
2. for $t = 1 \ldots T$ do
3. for $\ell = 1 \ldots N$ do
4. 
   $g_{k,\ell} = f_k(x_\ell, y_\ell) - E_{Pr(y'|w)} f_k(x_\ell, y')$ $k = 1 \ldots K$
5. end for
6. $g_k = \sum_{\ell} g_{k,\ell}$ $k = 1 \ldots K$
7. $w^t_k = w^{t-1}_k + \gamma_t (g_k - 2w^{t-1}_k / C)$
8. Exit if $\|g\| \approx \text{zero}$
9. end for

Running time of the algorithm is $O(INn(m^2 + K))$ where $I$ is the total number of iterations.
Training algorithm

1. Initialize $w^0 = 0$
2. for $t = 1 \ldots T$ do
3. for $\ell = 1 \ldots N$ do
4. $g_{k,\ell} = f_k(x_\ell, y_\ell) - \mathbb{E}_{\Pr(y'|w)} f_k(x_\ell, y')$ $k = 1 \ldots K$
5. end for
6. $g_k = \sum_\ell g_{k,\ell}$ $k = 1 \ldots K$
7. $w_k^t = w_k^{t-1} + \gamma_t (g_k - 2w_k^{t-1}/C)$
8. Exit if $\|g\| \approx zero$
9. end for

Running time of the algorithm is $O(INn(m^2 + K))$ where $I$ is the total number of iterations.
Calculating $E_{Pr(y'|w)} f_k(x_\ell, y')$ using inference.
Likelihood-based trainer

1. Penalizes all wrong $y$s the same way, does not exploit $E_i(y)$
2. Requires the computation of sum-marginals, not possible in all kinds of structured learning.

   1. Collective extraction
   2. Sentence Alignment
   3. Ranking
Outline

1. Likelihood based Training

2. Max-margin training
   - Decomposition-based approaches
   - Cutting-plane approaches
Two formulations

Margin scaling

\[
\min_{\mathbf{w}, \xi} \frac{1}{2} ||\mathbf{w}||^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i \\
\text{s.t. } \mathbf{w}^T \mathbf{f}(x_i, y_i) - \mathbf{w}^T \mathbf{f}(x_i, y) \geq E_i(y) - \xi_i \quad \forall y, i
\]
Two formulations

1. Margin scaling

\[
\min_{w, \xi} \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i \\
\text{s.t. } w^T f(x_i, y_i) - w^T f(x_i, y) \geq E_i(y) - \xi_i \quad \forall y, i
\]

2. Slack scaling

\[
\min_{w, \xi} \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i \\
\text{s.t. } w^T f(x_i, y_i) - w^T f(x_i, y) \geq 1 - \frac{\xi_i}{E_i(y)} \quad \forall y, i
\]
Max-margin loss surrogates

True error \( E_i(\text{argmax}_y w.f(x_i, y)) \)

Let \( w.\delta f(x_i, y) = w.f(x_i, y_i) - w.f(x_i, y) \)

1. Margin Loss
\[
\max_y [E_i(y) - w.\delta f(x_i, y)]_+
\]

2. Slack Loss
\[
\max_y E_i(y)[1 - w.\delta f(x_i, y)]_+
\]
Max-margin training: margin-scaling

The Primal (P):

\[
\min_{\mathbf{w}, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i \\
\text{s.t. } \mathbf{w}^T \delta f_i(\mathbf{y}) \geq E_i(\mathbf{y}) - \xi_i \quad \forall \mathbf{y}, i : 1 \ldots N
\]
Max-margin training: margin-scaling

The Primal (P):

$$\min_{w, \xi} \frac{1}{2} ||w||^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i$$

s.t. $w^T \delta f_i(y) \geq E_i(y) - \xi_i \quad \forall y, i : 1 \ldots N$

- Good news: Convex in $w, \xi$
- Bad news: exponential number of constraints
- Two main lines of attacks
  1. Decomposition: polynomial-sized rewrite of objective in terms of parts of $y$
  2. Cutting-plane: generate constraints on the fly.
1 The Primal (P):

\[
\min_{w, \xi} \frac{1}{2} ||w||^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i \\
\text{s.t. } w^T \delta f_i(y) \geq E_i(y) - \xi_i \quad \forall y, i: 1 \ldots N
\]
1. The Primal (P):
\[
\min_{\mathbf{w}, \xi} \frac{1}{2} \| \mathbf{w} \|^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i \\
\text{s.t. } \mathbf{w}^T \delta \mathbf{f}_i(y) \geq E_i(y) - \xi_i \quad \forall y, i : 1 \ldots N
\]

2. The Dual (D) of (P)
\[
\max_{\alpha_i(y)} -\frac{1}{2} \sum_{i,y} \alpha_i(y) \delta \mathbf{f}_i(y) \sum_{j,y'} \alpha_j(y') \delta \mathbf{f}_j(y') + \sum E_i(y) \alpha_i(y) \\
\text{s.t. } \sum_{y} \alpha_i(y) = \frac{C}{N} \\
\alpha_i(y) \geq 0 \quad i : 1 \ldots N
\]
Properties of Dual

1. Strong duality holds: Primal (P) solution = Dual (D) solution.

2. \( w = \sum_{i,y} \alpha_i(y) \delta f_i(y) \)

3. Dual (D) is concave in \( \alpha \), constraints are simpler.

4. Size of \( \alpha \) is still intractably large \( \Rightarrow \) cannot solve via standard libraries.
Decomposition-based approaches

1. \( \delta f_i(y) = \sum_c \delta f_{i,c}(y_c) \)
2. \( E_i(y) = \sum_c E_{i,c}(y_c) \)
Decomposition-based approaches

1. \[ \delta \mathbf{f}_i (\mathbf{y}) = \sum_c \delta \mathbf{f}_{i, c} (\mathbf{y}_c) \]
2. \[ E_i (\mathbf{y}) = \sum_c E_{i, c} (\mathbf{y}_c) \]

Rewrite the dual as

\[
\max_{\mu_{i, c} (\mathbf{y}_c)} \frac{1}{2} \sum_{i, c, \mathbf{y}_c} \delta \mathbf{f}_{i, c} (\mathbf{y}_c) \mu_{i, c} (\mathbf{y}_c) \sum_{j, d, \mathbf{y}_d'} \delta \mathbf{f}_{j, d} (\mathbf{y}_d') \mu_{j, d} (\mathbf{y}_d') \\
+ \sum_{i, c, \mathbf{y}_c} E_{i, c} (\mathbf{y}_c) \mu_{i, c} (\mathbf{y}_c)
\]

s.t. \[ \sum_{\mathbf{y}} \mu_{i, c} (\mathbf{y}_c) = \sum_{\mathbf{y} \sim \mathbf{y}_c} \alpha_i (\mathbf{y}) \]
\[ \sum_{\mathbf{y}} \alpha_i (\mathbf{y}) = \frac{C}{N}, \alpha_i (\mathbf{y}) \geq 0 \quad i : 1 \ldots N \]
\( \alpha \)s as probabilities

Scale \( \alpha \)s with \( \frac{C}{N} \).

\[
\max_{\mu_{i,c}(y_c)} \left\{ \frac{C}{2N} \sum_{i,c,y_c} \delta f_{i,c}(y_c) \mu_{i,c}(y_c) \sum_{j,d,y'_d} \delta f_{j,d}(y'_d) \mu_{j,d}(y'_d) \right. \\
+ \sum_{i,c,y_c} E_{i,c}(y_c) \mu_{i,c}(y_c) \\
\text{s.t. } \mu_{i,c}(y_c) \in \text{Marginals of any valid distribution}
\]
$\alpha$s as probabilities

Scale $\alpha$s with $\frac{C}{N}$.

$$\max_{\mu_{i,c}(y_c)} \frac{C}{2N} \sum_{i,c,y_c} \delta f_{i,c}(y_c) \mu_{i,c}(y_c) \sum_{j,d,y'_d} \delta f_{j,d}(y'_d) \mu_{j,d}(y'_d)$$

$$+ \sum_{i,c,y_c} E_{i,c}(y_c) \mu_{i,c}(y_c)$$

s.t. $\mu_{i,c}(y_c) \in \text{Marginals of any valid distribution}$

Solve via the exponentiated gradient method.
Exponentiated gradient algorithm

1. Initially $\mu_{i,c}(y_{i,c}) = 1$, for $y_{i,c} \neq y_c$, $\mu_{i,c}(y_c) = 0$
2. For $t = 1, \ldots, T$
   1. Choose a $i$ from $1, \ldots, N$.
Exponentiated gradient algorithm

1. Initially $\mu_{i,c}(y_{i,c}) = 1$, for $y_{i,c} \neq y_c$, $\mu_{i,c}(y_c) = 0$

2. For $t = 1, \ldots, T$
   1. Choose a $i$ from $1, \ldots, N$.
   2. Ignore constraints and perform a gradient-based update:
      \[ s_{i,c} = \mu_{i,c}^t + \eta(E_{i,c} - w^t \delta f_{i,c}(y_c)) \]
      where $w^t = \sum_{i,c,y_c} \mu_{i,c}^t(y_c) \delta f_{i,c}(y_c)$
Exponentiated gradient algorithm

1. Initially $\mu_{i,c}(y_{i,c}) = 1$, for $y_{i,c} \neq y_c$, $\mu_{i,c}(y_c) = 0$

2. For $t = 1, \ldots, T$
   - Choose a $i$ from $1, \ldots, N$.
   - Ignore constraints and perform a gradient-based update:
     $$s_{i,c} = \mu_{i,c}^t + \eta(E_{i,c} - w^t \delta f_i(y_c))$$
     where $w^t = \sum_{i,c,y_c} \mu_{i,c}^t(y_c) \delta f_{i,c}(y_c)$

3. Define a distribution $\alpha$ by exponentiating the updates:
   $$\alpha_i(y)^{t+1} = \frac{1}{Z} \exp(\sum_c s_{i,c}(y_c))$$
   where $Z = \sum_y \exp(\sum_c s_{i,c}(y_c))$
Exponentiated gradient algorithm

1. Initially $\mu_{i,c}(y_{i,c}) = 1$, for $y_{i,c} \neq y_c$, $\mu_{i,c}(y_c) = 0$
2. For $t = 1, \ldots, T$
   1. Choose an $i$ from $1, \ldots, N$.
   2. Ignore constraints and perform a gradient-based update:
      
      $$s_{i,c} = \mu_{i,c}^t + \eta (E_{i,c} - w^t \delta f_i(y_c))$$

      where $w^t = \sum_{i,c,y_c} \mu_{i,c}^t (y_c) \delta f_i, c(y_c)$
   3. Define a distribution $\alpha$ by exponentiating the updates:
      
      $$\alpha_{i}(y)^{t+1} = \frac{1}{Z} \exp(\sum_{c} s_{i,c}(y_c))$$

      where $Z = \sum_{y} \exp(\sum_{c} s_{i,c}(y_c))$
   4. New feasible values are marginals of $\alpha$
      
      $$\mu_{i,c}^{t+1}(y_c) = \sum_{y \sim y_c} \alpha_{i}(y)^{t+1}$$
Convergence results

Theorem

\[ J(\alpha^{t+1}) - J(\alpha^t) \geq \frac{1}{\eta} KL(\alpha^t, \alpha^{t+1}) \]

where \( \eta \leq \frac{1}{nR^2} \) where

\[ R = \max \delta f_i(y) \delta f_j(y') \]
Convergence results

Theorem

\[ J(\alpha^{t+1}) - J(\alpha^t) \geq \frac{1}{\eta} KL(\alpha^t, \alpha^{t+1}) \]

where \( \eta \leq \frac{1}{nR^2} \)

where \( R = \max \delta f_i(y) \delta f_j(y') \)

Theorem

Let \( \alpha^* = \text{dual optimal}. \) Then at the \( T \)th iteration.

\[ J(\alpha^*) - \frac{1}{T \eta} KL(\alpha^*; \alpha^0) \leq J(\alpha^{T+1}) \leq J(\alpha^*) \]
## Convergence results

**Theorem**

\[ J(\alpha^{t+1}) - J(\alpha^t) \geq \frac{1}{\eta} KL(\alpha^t, \alpha^{t+1}) \]  
where \( \eta \leq \frac{1}{nR^2} \) where \( R = \max \delta f_i(y) \delta f_j(y') \)

**Theorem**

Let \( \alpha^* = \) dual optimal. Then at the \( T \)th iteration.

\[ J(\alpha^*) - \frac{1}{T \eta} KL(\alpha^*; \alpha^0) \leq J(\alpha^{T+1}) \leq J(\alpha^*) \]

**Theorem**

The number of iterations of the algorithm is at most

\[ \frac{N^2}{\epsilon} R^2 KL(\alpha^*; \alpha^0) \]
Cutting plane method

1. Exponentiated gradient approach requires computation of sum-marginals and decomposable losses.

2. Cutting plane — a more general approach that just requires MAP
Cutting-plane algorithm [TJHA05]

1. Initialize $w^0 = 0$, Active constraints=Empty.
Cutting-plane algorithm [TJHA05]

1: Initialize \( w^0 = 0 \), Active constraints=Empty.
2: for \( t = 1 \ldots T \) do
3:   for \( \ell = 1 \ldots N \) do
4:     \( \hat{y} = \arg\max_y \left( E_\ell(y) + w^t \cdot f(x_\ell, y) \right) \)
5:     if \( w^t \cdot \delta f_\ell(\hat{y}) < E_\ell(\hat{y}) - \xi_\ell - \epsilon \) then
6:       Add \((x_\ell, \hat{y})\) to set of active constraints.
7:       \( w^t, \xi^t \) = solve QP with active constraints.
8:     end if
9:   end for
10:  Exit if no new constraint added.
11: end for
Efficient solution in the dual space

Solve QP in the dual space.

1. Initially $\alpha_{y_i}^t = 0$, $\forall y \neq y_i$, $\alpha_{y_i}^t = \frac{C}{N}$

2. For $t = 1, \ldots, T$
   1. Choose an $i$ from $1, \ldots, N$.
   2. $\hat{y} = \operatorname{argmax}_y (E_\ell(y) + w^t \cdot f(x_\ell, y))$ where $w^t = \sum_{i,y} \alpha_i(y) \delta f_i(y)$
   3. $\alpha_i(\hat{y}) = \text{coordinate with highest gradient}$.
   4. Optimize $J(\alpha)$ over set of $y$s in the active set (SMO applicable here).
Convergence results

Let $R^2 = \max \delta f_i(y) \delta f_j(y')$, $\Delta = \max_{i,y} E_i(y)$

**Theorem**

$$J(\alpha^{t+1}) - J(\alpha^t) \geq \min\left(\frac{C \epsilon}{2N}, \frac{\epsilon^2}{8R^2}\right)$$

**Theorem**

The number of constraints that the cutting plane algorithm adds is at most
$$\max\left(\frac{2N \Delta}{\epsilon}, \frac{8C \Delta R^2}{\epsilon^2}\right)$$
Theorem

The number of constraints that the cutting plane algorithm adds in the single slack formulation is at most

\[
\max \left( \log \frac{C \Delta}{4R^2C^2}, \frac{16CR^2}{\varepsilon} \right)
\]
Summary

1. Two very efficient algorithms for training structured models that avoids the problem of exponential output space.

2. Other alternatives
   1. Online training, example MIRA and Collins Trainer
   3. Local training: SEARN

3. Extension to Slack-scaling and other loss functions.
Peter L. Bartlett, Michael Collins, Ben Taskar, and David McAllester.

Exponentiated gradient algorithms for large-margin structured classification.


Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks.


Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu.

Cutting-plane training of structural SVMs.

Large margin methods for structured and interdependent output variables.