Training algorithms for Structured Learning

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

Training

Given

- N input output pairs $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$
- Features $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_c \mathbf{f}(\mathbf{x}, \mathbf{y}_c, c)$
- Error of output : $E(\mathbf{y}_i, \mathbf{y})$
  - (Use short form: $E(\mathbf{y}_i, \mathbf{y}) = E_i(\mathbf{y})$)
  - Also decomposes over smaller parts: $E_i(\mathbf{y}) = \sum_{c \in C} E_{i,c}(\mathbf{y}_c)$

Find $\mathbf{w}$

- 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

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{w}}(\mathbf{x})} \exp(\mathbf{w}.\mathbf{f}(\mathbf{x}, \mathbf{y}))$$

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

Probability distribution from scores

- Convert scores into a probability distribution

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{w}}(\mathbf{x})} \exp(\mathbf{w}.\mathbf{f}(\mathbf{x}, \mathbf{y}))$$

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

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

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{w}}(\mathbf{x})} \exp(\sum_c \mathbf{w}.\mathbf{f}(\mathbf{x}, \mathbf{y}_c, c)) = \frac{1}{Z} \prod_c \psi_c(\mathbf{y}_c)$$

with clique potential $\psi_c(\mathbf{y}_c) = \exp(\mathbf{w}.\mathbf{f}(\mathbf{x}, \mathbf{y}_c, c))$

Probability distribution from scores

- Convert scores into a probability distribution

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{w}}(\mathbf{x})} \exp(\mathbf{w}.\mathbf{f}(\mathbf{x}, \mathbf{y}))$$

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

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

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{w}}(\mathbf{x})} \exp(\sum_c \mathbf{w}.\mathbf{f}(\mathbf{x}, \mathbf{y}_c, c)) = \frac{1}{Z} \prod_c \psi_c(\mathbf{y}_c)$$

with clique potential $\psi_c(\mathbf{y}_c) = \exp(\mathbf{w}.\mathbf{f}(\mathbf{x}, \mathbf{y}_c, c))$

- These are called Conditional Random Fields (CRFs).

Training via gradient descent

$$L(\mathbf{w}) = \sum_{\ell} \log \Pr(\mathbf{y}_\ell | \mathbf{x}_\ell, \mathbf{w}) = \sum_{\ell} (\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_\ell, \mathbf{y}_\ell) - \log Z_{\mathbf{w}}(\mathbf{x}_\ell))$$

Add a regularizer to prevent over-fitting.

$$\max_{\mathbf{w}} \sum_{\ell} (\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_\ell, \mathbf{y}_\ell) - \log Z_{\mathbf{w}}(\mathbf{x}_\ell)) - ||\mathbf{w}||^2 / C$$

Concave in $\mathbf{w}$ $\implies$ gradient descent methods will work.
Gradient:

$$\begin{aligned}
\nabla L(\mathbf{w}) &= \sum_{\ell} \mathbf{f}(\mathbf{x}_\ell, \mathbf{y}_\ell) - \frac{\sum_{\mathbf{y}'} \mathbf{f}(\mathbf{y}', \mathbf{x}_\ell) \exp \mathbf{w} \cdot \mathbf{f}(\mathbf{x}_\ell, \mathbf{y}')}{Z_{\mathbf{w}}(\mathbf{x}_\ell)} - 2\mathbf{w}/C \\
&= \sum_{\ell} \mathbf{f}(\mathbf{x}_\ell, \mathbf{y}_\ell) - E_{\Pr(\mathbf{y}'|\mathbf{w})} \mathbf{f}(\mathbf{x}_\ell, \mathbf{y}') - 2\mathbf{w}/C
\end{aligned}$$

Training algorithm

1: Initialize $\mathbf{w}^0 = \mathbf{0}$

Training algorithm

Initialize $\mathbf{w}^0 = \mathbf{0}$
for $t = 1 \ldots \mathrm{T}$ **do**
  for $\ell = 1 \ldots N$ **do**
    $g_{k,\ell} = f_k(\mathbf{x}_\ell, \mathbf{y}_\ell) - E_{\Pr(\mathbf{y}'|\mathbf{w})} f_k(\mathbf{x}_\ell, \mathbf{y}') \quad k = 1 \ldots K$
  end for
  $g_k = \sum_\ell g_{k,\ell} \quad k = 1 \ldots K$

Training algorithm

Initialize $\mathbf{w}^0 = \mathbf{0}$
for $t = 1 \ldots \mathrm{T}$ **do**
  for $\ell = 1 \ldots N$ **do**
    $g_{k,\ell} = f_k(\mathbf{x}_\ell, \mathbf{y}_\ell) - E_{\Pr(\mathbf{y}'|\mathbf{w})} f_k(\mathbf{x}_\ell, \mathbf{y}') \quad k = 1 \ldots K$
  end for
  $g_k = \sum_\ell g_{k,\ell} \quad k = 1 \ldots K$
  $w_k^t = w_k^{t-1} + \gamma_t (g_k - 2 w_k^{t-1} / C)$
  Exit if $||\mathbf{g}|| \approx$ *zero*
end for

Training algorithm

```
Initialize w^0 = 0
for t = 1 ... T do
    for ℓ = 1 ... N do
        g_{k,ℓ} = f_k(x_ℓ, y_ℓ) − E_{Pr(y'|w)} f_k(x_ℓ, y')    k = 1 ... K
    end for
    g_k = Σ_ℓ g_{k,ℓ}    k = 1 ... K
    w_k^t = w_k^{t−1} + γ_t(g_k − 2w_k^{t−1}/C)
    Exit if ||g|| ≈ zero
end for
```

$$
\begin{array}{l}
\text{Initialize } \mathbf{w}^0 = \mathbf{0} \\
\textbf{for } t = 1 \ldots \mathrm{T} \textbf{ do} \\
\quad \textbf{for } \ell = 1 \ldots N \textbf{ do} \\
\quad\quad g_{k,\ell} = f_k(\mathbf{x}_\ell, \mathbf{y}_\ell) - E_{\Pr(\mathbf{y}'|\mathbf{w})} f_k(\mathbf{x}_\ell, \mathbf{y}') \quad k = 1 \ldots K \\
\quad \textbf{end for} \\
\quad g_k = \sum_\ell g_{k,\ell} \quad k = 1 \ldots K \\
\quad w_k^t = w_k^{t-1} + \gamma_t(g_k - 2w_k^{t-1}/C) \\
\quad \textbf{Exit} \text{ if } ||\mathbf{g}|| \approx \textit{zero} \\
\textbf{end for}
\end{array}
$$

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

Calculating $E_{\Pr(\mathbf{y}'|\mathbf{w})} f_k(\mathbf{x}_\ell, \mathbf{y}')$ using inference.

Likelihood-based trainer

1. Penalizes all wrong **y**s the same way, does not exploit $E_i(\mathbf{y})$
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

1. Margin scaling

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

$$\text{s.t. } \mathbf{w}^T\mathbf{f}(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{w}^T\mathbf{f}(\mathbf{x}_i, \mathbf{y}) \geq E_i(\mathbf{y}) - \xi_i \quad \forall \mathbf{y}, i$$

Two formulations

1. Margin scaling

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

$$\text{s.t. } \mathbf{w}^T\mathbf{f}(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{w}^T\mathbf{f}(\mathbf{x}_i, \mathbf{y}) \geq E_i(\mathbf{y}) - \xi_i \quad \forall \mathbf{y}, i$$

2. Slack scaling

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

$$\text{s.t. } \mathbf{w}^T\mathbf{f}(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{w}^T\mathbf{f}(\mathbf{x}_i, \mathbf{y}) \geq 1 - \frac{\xi_i}{E_i(\mathbf{y})} \quad \forall \mathbf{y}, i$$

Max-margin loss surrogates

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

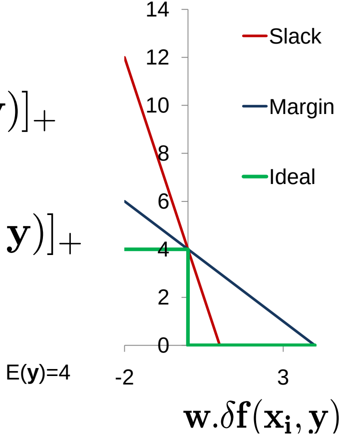
Let $\mathbf{w}.\delta\mathbf{f}(\mathbf{x_i}, \mathbf{y}) = \mathbf{w}.\mathbf{f}(\mathbf{x_i}, \mathbf{y_i}) - \mathbf{w}.\mathbf{f}(\mathbf{x_i}, \mathbf{y})$

1. Margin Loss

$$\max_{\mathbf{y}} [E_i(\mathbf{y}) - \mathbf{w}.\delta\mathbf{f}(\mathbf{x_i}, \mathbf{y})]_+$$

2. Slack Loss

$$\max_{\mathbf{y}} E_i(\mathbf{y})[1 - \mathbf{w}.\delta\mathbf{f}(\mathbf{x_i}, \mathbf{y})]_+$$



Max-margin training: margin-scaling

The Primal (P):

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

$$\text{s.t. } \mathbf{w}^T\delta\mathbf{f}_i(\mathbf{y}) \geq E_i(\mathbf{y}) - \xi_i \quad \forall \mathbf{y}, i : 1 \dots N$$

Max-margin training: margin-scaling

The Primal (P):

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

$$\text{s.t. } \mathbf{w}^T\delta\mathbf{f}_i(\mathbf{y}) \geq E_i(\mathbf{y}) - \xi_i \quad \forall \mathbf{y}, i : 1 \ldots N$$

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

1. The Primal (P):

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

1. The Primal (P):

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

2. The Dual (D) of (P)

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

Properties of Dual

1. Strong duality holds: Primal (P) solution $=$ Dual (D) solution.
2. $\mathbf{w} = \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y}) \delta \mathbf{f}_i(\mathbf{y})$
3. Dual (D) is concave in α, constraints are simpler.
4. Size of α is still intractably large $\implies$ cannot solve via standard libraries.

Decomposition-based approaches

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

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

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

$$+ \sum_{i,c,\mathbf{y}_c} E_{i,c}(\mathbf{y}_c) \mu_{i,c}(\mathbf{y}_c)$$

$$\text{s.t. } \sum_{\mathbf{y}} \mu_{i,c}(\mathbf{y}_c) = \sum_{\mathbf{y} \sim \mathbf{y}_c} \alpha_i(\mathbf{y})$$

$$\sum_{\mathbf{y}} \alpha_i(\mathbf{y}) = \frac{C}{N}, \alpha_i(\mathbf{y}) \geq 0 \quad i : 1 \ldots N$$

αs as probabilities

Scale αs with $\frac{C}{N}$.

$$\max_{\mu_{i,c}(\mathbf{y}_c)} -\frac{C}{2N} \sum_{i,c,\mathbf{y}_c} \delta\mathbf{f}_{i,c}(\mathbf{y}_c)\mu_{i,c}(\mathbf{y}_c) \sum_{j,d,\mathbf{y}'_d} \delta\mathbf{f}_{j,d}(\mathbf{y}'_d)\mu_{j,d}(\mathbf{y}'_d)$$

$$+ \sum_{i,c,\mathbf{y}_c} E_{i,c}(\mathbf{y}_c)\mu_{i,c}(\mathbf{y}_c)$$

s.t. $\mu_{i,c}(\mathbf{y}_c) \in$ Marginals of any valid distribution

αs as probabilities

Scale αs with $\frac{C}{N}$.

$$\max_{\mu_{i,c}(\mathbf{y}_c)} -\frac{C}{2N} \sum_{i,c,\mathbf{y}_c} \delta\mathbf{f}_{i,c}(\mathbf{y}_c)\mu_{i,c}(\mathbf{y}_c) \sum_{j,d,\mathbf{y}'_d} \delta\mathbf{f}_{j,d}(\mathbf{y}'_d)\mu_{j,d}(\mathbf{y}'_d)$$

$$+ \sum_{i,c,\mathbf{y}_c} E_{i,c}(\mathbf{y}_c)\mu_{i,c}(\mathbf{y}_c)$$

s.t. $\mu_{i,c}(\mathbf{y}_c) \in$ Marginals of any valid distribution

Solve via the exponentiated gradient method.

Exponentiated gradient algorithm

1. Initially $\mu_{i,c}(\mathbf{y}_{i,c}) = 1$, for $\mathbf{y}_{i,c} \neq \mathbf{y}_c$, $\mu_{i,c}(\mathbf{y}_c) = 0$
2. For $t = 1, \ldots, T$
    1. Choose a i from $1, \ldots, N$.

Exponentiated gradient algorithm

1. Initially $\mu_{i,c}(\mathbf{y}_{i,c}) = 1$, for $\mathbf{y}_{i,c} \neq \mathbf{y}_c$, $\mu_{i,c}(\mathbf{y}_c) = 0$
2. For $t = 1, \ldots, T$
   1. Choose a i from $1, \ldots, N$.
   2. Ignore constraints and perform a gradient-based update:

$$s_{i,c} = \mu^t_{i,c} + \eta(E_{i,c} - \mathbf{w}^t \delta \mathbf{f}_i(\mathbf{y}_c))$$

where $\mathbf{w}^t = \sum_{i,c,\mathbf{y}_c} \mu^t_{i,c}(\mathbf{y}_c) \delta \mathbf{f}_{i,c}(\mathbf{y}_c)$

Exponentiated gradient algorithm

1. Initially $\mu_{i,c}(\mathbf{y}_{i,c}) = 1$, for $\mathbf{y}_{i,c} \neq \mathbf{y}_c$, $\mu_{i,c}(\mathbf{y}_c) = 0$
2. For $t = 1, \ldots, T$
    1. Choose a i from $1, \ldots, N$.
    2. Ignore constraints and perform a gradient-based update:
    $$s_{i,c} = \mu_{i,c}^t + \eta(E_{i,c} - \mathbf{w}^t \delta \mathbf{f}_i(\mathbf{y}_c))$$
    where $\mathbf{w}^t = \sum_{i,c,\mathbf{y}_c} \mu_{i,c}^t(\mathbf{y}_c)\delta \mathbf{f}_{i,c}(\mathbf{y}_c)$
    3. Define a distribution α by exponentiating the updates:
    $$\alpha_i(\mathbf{y})^{t+1} = \frac{1}{Z}\exp(\sum_c s_{i,c}(\mathbf{y}_c))$$
    where $Z = \sum_{\mathbf{y}} \exp(\sum_c s_{i,c}(\mathbf{y}_c))$

Exponentiated gradient algorithm

1. Initially $\mu_{i,c}(\mathbf{y}_{i,c}) = 1$, for $\mathbf{y}_{i,c} \neq \mathbf{y}_c$, $\mu_{i,c}(\mathbf{y}_c) = 0$
2. For $t = 1, \ldots, T$
   1. Choose a i from $1, \ldots, N$.
   2. Ignore constraints and perform a gradient-based update:
      $$s_{i,c} = \mu_{i,c}^t + \eta(E_{i,c} - \mathbf{w}^t \delta \mathbf{f}_i(\mathbf{y}_c))$$
      where $\mathbf{w}^t = \sum_{i,c,\mathbf{y}_c} \mu_{i,c}^t(\mathbf{y}_c) \delta \mathbf{f}_{i,c}(\mathbf{y}_c)$
   3. Define a distribution α by exponentiating the updates:
      $$\alpha_i(\mathbf{y})^{t+1} = \frac{1}{Z} \exp\Big(\sum_c s_{i,c}(\mathbf{y}_c)\Big)$$
      where $Z = \sum_{\mathbf{y}} \exp(\sum_c s_{i,c}(\mathbf{y}_c))$
   4. New feasible values are marginals of α
      $$\mu_{i,c}^{t+1}(\mathbf{y}_c) = \sum_{\mathbf{y} \sim \mathbf{y}_c} \alpha_i(\mathbf{y})^{t+1}$$

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* $R = \max \delta \mathbf{f}_i(\mathbf{y}) \delta \mathbf{f}_j(\mathbf{y}')$

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* $R = \max \delta \mathbf{f}_i(\mathbf{y}) \delta \mathbf{f}_j(\mathbf{y}')$

Theorem

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

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

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* $R = \max \delta \mathbf{f}_i(\mathbf{y}) \delta \mathbf{f}_j(\mathbf{y}')$

Theorem

Let α^* = *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]

Initialize $\mathbf{w}^0 = \mathbf{0}$, Active constraints=Empty.

Cutting-plane algorithm [TJHA05]

Initialize $\mathbf{w}^0 = \mathbf{0}$, Active constraints=Empty.
for $t = 1 \ldots \mathrm{T}$ **do**
  for $\ell = 1 \ldots N$ **do**
    $\hat{\mathbf{y}} = \mathrm{argmax}_{\mathbf{y}}(E_\ell(\mathbf{y}) + \mathbf{w}^t \cdot \mathbf{f}(\mathbf{x}_\ell, \mathbf{y}))$
    if $\mathbf{w}^t . \delta \mathbf{f}_\ell(\hat{\mathbf{y}}) < E_\ell(\hat{\mathbf{y}}) - \xi_\ell^t - \epsilon$ **then**
      Add $(\mathbf{x}_\ell, \hat{\mathbf{y}})$ to set of active constraints.
      $\mathbf{w}^t, \xi^t$=solve QP with active constraints.
    end if
  end for
  Exit if no new constraint added.
end for

Efficient solution in the dual space

Solve QP in the dual space.

1. Initially $\alpha_i^t(\mathbf{y}) = 0, \forall \mathbf{y} \neq \mathbf{y}_i, \alpha_i^t(\mathbf{y}_i) = \frac{C}{N}$
2. For $t = 1, \ldots, T$
   1. Choose a i from $1, \ldots, N$.
   2. $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}}(E_\ell(\mathbf{y}) + \mathbf{w}^t \cdot \mathbf{f}(\mathbf{x}_\ell, \mathbf{y}))$ where $\mathbf{w}^t = \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\delta\mathbf{f}_i(\mathbf{y})$
   3. $\alpha_i(\hat{\mathbf{y}}) =$ coordinate with highest gradient.
   4. Optimize $J(\alpha)$ over set of $\mathbf{y}$s in the active set (SMO applicable here).

Convergence results

Let $R^2 = \max \delta \mathbf{f}_i(\mathbf{y}) \delta \mathbf{f}_j(\mathbf{y}'), \Delta = \max_{i,\mathbf{y}} E_i(\mathbf{y})$

Theorem

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

Theorem

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

Single slack formulation [JFY09]

Theorem

The number of constraints that the cutting plane algorithm adds in the single slack formulation is at most $\max(\log \frac{C\Delta}{4R^2C^2}, \frac{16CR^2}{\epsilon})$

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
   2. Stochastic trainers: LARank.
   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.

In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 113–120. MIT Press, Cambridge, MA, 2005.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett.

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

J. Mach. Learn. Res., 9:1775–1822, 2008.

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

Cutting-plane training of structural svms.

Machine Learning, 77(1):27–59, 2009.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun.

Large margin methods for structured and interdependent output variables.

Journal of Machine Learning Research (JMLR), 6(Sep):1453–1484, 2005.