Conditional Random Fields (CRFs)

CRFs model probability of output \( y \in Y \) given input \( x \in X \) and features \( F(x, y) \) using

\[
p(y | x, w) = \exp(w^T F(x, y)) \sum_y \exp(w^T F(x, y))
\]

Given training examples \( \{(x_i, y_i)\}, \) standard approach is minimizing \( \ell_2\)-regularized NLL:

\[
\min_w \frac{1}{n} \sum_{i=1}^{n} \log p(y_i | x_i, w) + \lambda \frac{1}{2} \|w\|^2.
\]

Evaluating each \( \log p(y_i | x_i, w) \) is expensive due to sum over \( Y \).

Related Work on Deterministic, Stochastic, and Hybrid Methods

- Deterministic gradient methods like L-BFGS [Wallach et al., 2008, Spa and Pereira, 2003]: Requires \( O(\log(1/\epsilon)) \) iterations but 1 gradient evaluation per iteration.
- Stochastic gradient methods [Vishwanathan et al., 2006, Finkel et al., 2008]: Requires \( O(1/\epsilon) \) iterations but only 1 gradient evaluation per iteration.
- Online exponentiated gradient [Collins et al., 2008]: Requires \( O(\log(1/\epsilon)) \) iterations in terms of dual and 1 gradient evaluation per iteration.
- Hybrid deterministic-stochastic [Friedlander & Schmidt, 2012]: Requires \( O(\log(1/\epsilon)) \) iterations and growing number of gradient evaluations per iteration.

Stochastic Average Gradient (SAG) for CRFs

- Stochastic average gradient [Le Roux et al., 2012]: Requires \( O(\log(1/\epsilon)) \) iterations and 1 gradient evaluation per iteration.
- SAG uses the iteration

\[
w^{t+1} = w^t - \frac{\alpha}{n} \sum_{i=1}^{n} s_i^t,
\]

where we set \( s_i^t = -\nabla \log p(y_i | x_i, w^t) + \lambda w^t \) for one randomly-chosen training example.

- Challenge is the memory required for storing the \( s_i^t \):
  - \( \nabla \log p(y_i | x_i, w) \) often sparse but depends on number of features.
  - \( \lambda w \) is typically dense.

Implementation issues for CRFs:
- Sparse trick 1: to avoid storing \( \lambda w \), use the exact gradient of the regularizer.

\[
w^{t+1} = (1-\alpha)w^t - \frac{\alpha}{n} \sum_{i=1}^{n} s_i^t
\]

where we set \( s_i^t = -\nabla \log p(y_i | x_i, w^t) \) for one randomly-chosen example.

- Sparse trick 2: use the representation \( w^t - \frac{\alpha}{n} \sum_{i=1}^{n} s_i^t \) and ‘easy updates’ to avoid dense vector operations.
- Step size: set \( \alpha = 1/L \) with \( L = 4\lambda + \lambda \), and double approximation \( L \) when

\[
\ell(w - (1/\alpha)\hat{w}) > \ell(w) - \frac{\ell(w)}{2\alpha^2} \tag{1}
\]

but we multiply \( L \) by \( 2^{1/4} \) after every iteration to slowly increase the step size.

- Convergence: we can stop if \( \|w^t - \frac{1}{n} \sum_{i=1}^{n} s_i^t\| \) is sufficiently small.
- Reducing the memory: for ‘part-based’ features, \( F(x, y) = F(x) |_{y = \hat{y}} \), the gradient has the form

\[
\nabla \log p(y_i | x_i, w) = \frac{F(x) |_{y = \hat{y}} - F(x) |_{y = y_i}}{p(y_i | x_i, w) - p(y_i | x_i, \hat{y})}
\]

and SAG update depends on approximations in gradients.

\[
\nabla \log p(y_i | x_i, w) - \nabla \log p(y_i | x_i, \hat{y}) = F(x) |_{y = \hat{y}} - F(x) |_{y = y_i}
\]

so we only need to store marginals \( p(y_i | x_i, \hat{y}) \) that are shared across features that depend on \( y_i = \hat{y} \).

SAG with Practical Non-Uniform Sampling (NUS) Strategy

- Assume each gradient has its own Lipschitz constant \( L_i \), a value such that

\[
\|\nabla f_i(w) - \nabla f_i(v)\| \leq L_i \|w - v\|.
\]

- Key idea behind NUS: bias sampling probability \( p_i \), towards Lipschitz constant \( L_i \):
  - Gradients that can change more quickly get updated more often.
  - Convergence rate depends on \( \frac{1}{L_i} \text{mean}(L_i) \) instead of \( \frac{1}{\text{max}(L_i)} \).
  - Practical ‘partially-based’ strategy:
    - With probability \( 1/2 \) choose \( i \) uniformly.
    - Use a larger step-size of \( \alpha = \frac{1}{2} \frac{1}{L_1 + 1/2} \)
    - Initialize with \( L_i = L \) the first time an example is chosen.
    - Each time \( i \) is chosen, set \( L_i = 0.9L_i \) then double it \( w \) times.
  - If \( \{i\} \) holds \( 2 \) times (without backtracking), do not change \( L_i \) for 2\( t \) next times \( \hat{i} \) is sampled.


Convergence Analysis for SAGA with Non-Uniform Sampling

- We analyze a NUS extension of SAGA, which has similar performance but easier analysis.
  - Let the sequences \( \{w_i\} \) and \( \{x_i\} \) be defined by

\[
\begin{align*}
w_i^{t+1} & = w_i^t - \frac{\alpha}{n} \sum_{i=1}^{n} s_i^t, \\
\end{align*}
\]

where \( j \) is chosen with probability \( p_j \).

- (a) \( p_j = \frac{1}{n} \) and \( \alpha = \frac{1}{8} \) we have

\[
\mathbb{E}[\|w_i - w^*\|^2] \leq (1 - 1/n^2) [\|x^* - x_i\|^2 + C_1],
\]

where

\[
C_1 = \frac{2(1+1/n)}{(1-1/n^2) \beta^2} \mathbb{E}[\|\nabla f_i(x_i)\|^2].
\]

- (b) \( p_j = \frac{1}{n} \) and \( \alpha = \frac{1}{8} \) we have

\[
\mathbb{E}[\|w_i - w^*\|^2] \leq (1 - 1/n^2) \left[ \frac{1}{2 \beta^2} \mathbb{E}[\|\nabla f_i(x_i)\|^2] \right] [\|x^* - x_i\|^2 + C_1],
\]

where

\[
C_1 = \frac{n}{2 \beta^2} \mathbb{E}[\|\nabla f_i(x_i)\|^2] - \mathbb{E}[\nabla f(x_i)]^2.
\]

(a) SAGA has a linear convergence rate wherever \( p_i \gg 1 \) for all \( i \).

(b) SAGA has a faster rate with \( p_i \) proportional to \( L_i \) and generating a uniform sample.

Experiment Results

- Figure: Training objective sub-optimality against effective number of passes for OCR, CONLL-2000, POS.

- Figure: Test error against effective number of passes, for OCR, CONLL-2000, POS. (Dashed lines are stochastic with sub-optimal step-size.

Discussion

- If memory requirements prohibitive, use mini-batches or SVRG [Johnson & Zhang, 2013].
- Could use \( \ell_2 \)-regularization with proximal versions [Difazio et al., 2014].
- Algorithms applies to any graph structure and approximate inference could be used.
- Could use multi-core computation and distributed parallel implementations.