Accelerated Training of Conditional Random Fields with Stochastic Gradient Methods

S.V.N. Vishwanathan, Nicol N. Schraudolph, Mark Schmidt, Kevin Murphy
ICML 2006
Overview

• Conditional Random Fields
• Batch Learning Methods
• Stochastic Gradient Methods
• Stochastic Meta-Descent
• Automatic Differentiation
• Gradient Approximations
Conditional Random Fields

- Discriminative model for structured data
  - \( \mathbb{P}(Y|\mathbf{x}) \) modeled directly
- Log-Likelihood:
  \[
p(y|x; \theta) = \exp(\langle \phi(x, y), \theta \rangle - z(\theta|x))
\]
- Log-Partition Function:
  \[
z(\theta|x) := \ln \sum_y \exp(\langle \phi(x, y), \theta \rangle)
\]
CRF Properties

- Exponential Family
- Continuous, Twice-Differentiable
- Probabilistic Interpretation
- Negative log-likelihood is convex (worst initialization => best parameters)
- Log-partition function is cumulant generating
- Efficient Calculation of Objective and Gradient for ‘thin’ graph structures
Objective and Gradients

\[ \mathcal{L}(\theta) := \frac{\|\theta\|^2}{2\sigma^2} - \sum_{i=1}^{m} \left[ \langle \phi(x_i, y_i), \theta \rangle - z(\theta | x_i) \right] \]

\[ g(\theta) = \frac{\theta}{\sigma^2} - \sum_{i=1}^{m} \left[ \phi(x_i, y_i) - \mathbb{E}_p(y|x_i; \theta) \phi(x_i, y) \right] \]

\[ H(\theta) = \frac{1}{\sigma^2} + \sum_{i=1}^{m} \text{Cov}_p(y|x_i; \theta) \phi(x_i, y) \]
Overview

• Conditional Random Fields
  • Batch Learning Methods
• Stochastic Gradient Methods
• Stochastic Meta-Descent
• Automatic Differentiation
• Gradient Approximations
Parameter Estimation

- Each evaluation of objective/gradient requires *inference* on each training example.
- Chains/Trees: Belief Propagation
- Learning is an unconstrained convex optimization
- Current state of the art:
  - Generalized Iterative Scaling
  - Newton Methods
Newton’s Method

\( w = \text{smallRand}; \)

\([f,g,H] = \text{@gradientFunction}(w);\)

do

\( \text{stepDir} = H \backslash g \)

\( \text{stepLen} = \text{lineSearch}(w + \text{stepLen} \ast \text{stepDir}) \)

\( w = w + \text{stepLen} \ast \text{stepDir} \)

\([f,g,H] = \text{@gradientFunction}(x);\)

while \( \text{norm}(g) > \text{optTol} \)
Quasi-Newton Method

\[
\begin{align*}
w &= \text{smallRand}; \\
B &= \text{eye}; \\
[f,g] &= \text{@gradientFunction}; \\
\text{do} & \\
\quad \text{stepDir} &= B \backslash g \\
\quad \text{stepLen} &= \text{lineSearch}(w + \text{stepLen} \times \text{stepDir}) \\
\quad \text{update}(B) \text{ } & \\
\quad w &= w + \text{stepLen} \times \text{stepDir} \\
[f,g] &= \text{@gradientFunction} \\
\text{while } \text{norm}(g) > \text{optTol}
\end{align*}
\]
BFGS Update

- Broyden-Fletcher-Goldfarb-Shanno (BFGS) Update:

\[ B_{i+1} = B_i + \frac{yy^T}{y^Ts} - \frac{B_is s^T B_i}{s^T B_is} \]

- Update Factorization or inverse instead of inverting \( B \):

  \[ \text{[In Matlab: } R = \text{cholupdate(cholupdate(R,y/sqrt(y'*s)),R'*R*s/sqrt(s'*R'*R*s),'-'));\]  

- Under certain conditions (initial \( B \) is pd, function convex, twice-differentiable, \( \text{sum(norm(x_k-x^*)< inf, Hessian Lipschitz continuous at minimizer found, line search satisfies Wolfe conditions):} \)
  - BFGS leads to super-linear convergence to global minimum
L-BFGS Update

• Re-write BFGS in terms of inverse:

\[ B_{i+1}^{-1} = (I - \frac{sy^T}{y^Ts})B_i^{-1}(I - \frac{ys^T}{y^Ts}) + \frac{ss^T}{y^Ts} \]

• Current Inverse Hessian can be computed recursively based on previous function and gradient values

• Limited Memory BFGS:
  • Compute B\(g\) without storing Hessian approximation
function [d] = lbfgs(s,y,g)
% [L-]BFGS Search Direction
%
% This function returns the (L-BFGS) approximate inverse Hessian,
% multiplied by the gradient
%
% If you pass in all previous parameter/gradient differences, it will be full BFGS
% If you truncate to the k most recent, it will be L-BFGS
%
% s - differences in parameters between last k steps (p by k)
% y - differences in gradient between last k steps(p by k)
% g - gradient (p by 1)

[p,k] = size(s);

for i = 1:k
    ro(i,1) = 1/(y(:,i)'*s(:,i));
end

q = zeros(p,k+1);
r = zeros(p,k+1);
al =zeros(k,1);
be =zeros(k,1);

q(:,k+1) = g;

for i = k:-1:1
    al(i) = ro(i)*s(:,i)'*q(:,i+1);
    q(:,i) = q(:,i+1)-al(i)*y(:,i);
end

r(:,1) = q(:,1);

for i = 1:k
    be(i) = ro(i)*y(:,i)'*r(:,i);
    r(:,i+1) = r(:,i) + s(:,i)*(al(i)-be(i));
end
d=r(:,k+1);
Overview

• Conditional Random Fields
• Batch Learning Methods
• **Stochastic Gradient Methods**
• Stochastic Meta-Descent
• Automatic Differentiation
• Gradient Approximations
CRF Parameter Learning

- Current Champ:
  - Quasi-Newton w/ [L-]BFGS Updating

- Challenger:
  - Stochastic Gradient
Stochastic Gradient

\[ w = \text{smallRand; } \]

\[ \text{for } i = 1:\text{maxIter} \]
\[ \quad \text{for } b = 1:\text{maxBatch} \]
\[ \quad \quad [f(b), g(b)] = @\text{gradientFunction}(b); \]
\[ \quad \quad w = w - \text{stepSize} \times g(b) \]
\[ \quad \text{end} \]

\[ \text{end} \]
CRF Parameter Learning

• Current Champ:
  • Quasi-Newton w/ [L-]BFGS Updating
  • Inference on all training examples

• Challenger:
  • Stochastic Gradient
  • Inference on batch of training examples
Experiment 1

- CoNLL-2000 Shared Word Chunking Task
- 8936 Sentences
- 330731 Features
- BFGS faster than NL-CG and GIS [3]
- Compare BFGS, Stochastic Gradient, Collin’s Perceptron (see Yann’s talk), SMD (later)
Learning vs. Optimization (revisited)

Figure 1. Left: F-scores on the CoNLL-2000 shared ... solution as BFGS (85.8%), significantly outperforming SGD (85.2%) and CP, whose oscillations are settling around 83%.
Learning vs. Optimization

Figure 1. Left: F-scores on the CoNLL-2000 shared task Development data for the "On"-Line solution as BFGS (85.8%), significantly outperforming SGD (85.2%) and CP, whose oscillations are settling around 83%.
Overview

• Conditional Random Fields
• Batch Learning Methods
• Stochastic Gradient Methods
• Stochastic Meta-Descent
• Automatic Differentiation
• Gradient Approximations
Disadvantage of Stochastic Gradient

• For a fixed step size:
  • May not converge
  • May converge too slowly

• For annealed step size:
  • Need to tune step size update

• Steepest Descent direction (batch case: sub-linear convergence, pathological cases converge in infinite number of steps)
SMD

- Stochastic Meta-Descent:
  - Attempt to translate non-linear CG to stochastic gradient learning
  - Adaptive Step Sizes for each dimension
  - Some 2nd-Order information provided through Hessian-Vector products
SMD

• Each parameter has its own gain:
  \[ \theta_{t+1} = \theta_t - \eta_t \cdot g_t. \]

• Update the gain multiplicatively by meta-gain (mu):
  \[ \eta_{t+1} = \eta_t \cdot \max\left(\frac{1}{2}, 1 - \mu g_{t+1} \cdot v_{t+1}\right) \]

• Update the long-term 2nd-order dependence w/ memory (lambda):
  \[ v_{t+1} = \lambda v_t - \eta_t \cdot (g_t + \lambda H_t v_t) \]
Overview

• Conditional Random Fields
• Batch Learning Methods
• Stochastic Gradient Methods
• Stochastic Meta-Descent
• Automatic Differentiation
• Gradient Approximations
Hessian-Vector Products

- Finite Differencing: For any $d$, can compute Hessian-Vector product using 2 gradient evaluations:
  \[
  dg(\theta) = H(\theta) \, d\theta \quad \text{and} \quad dg(\theta) \approx \frac{g(\theta + \epsilon d) - g(\theta)}{\epsilon}
  \]

- Algorithmic Differentiation: Under arithmetic assumption about gradient evaluation, can use 1 gradient evaluation and complex perturbation:
  \[
  g(\theta + i \epsilon \, d\theta) = g(\theta) + O(\epsilon^2) + i \epsilon \, dg(\theta)
  \]
for i = 2:T
    for b=1:Nbatches
        batchNdx = batchIndices{b};
        % Nic's code - uses complex number trick
        [f(b),g] = feval(gradient, w + ii*v, batchNdx, gradArgs{:});
        eta = eta.*max(1/2,1+mu*v.*real(g));
        w = w - eta.*real(g);
        v = lambda*v+eta.*(real(g)-lambda*imag(g)*1e150);
Experiment 2

- BioNLP/NLPBA-2004 Shared Task:
  - Biomedical Named Entity recognition on GENIA corpus
- 18546 Sentences
- 106583 Features
“On”-Line

![Graph showing F-score (%) vs passes for different optimization methods: SMD, SGD, BFGS, and CP. The graph compares the performance of these methods under varying conditions, highlighting the effectiveness of the optimization techniques in achieving higher F-scores.](image-url)
Overview

- Conditional Random Fields
- Batch Learning Methods
- Stochastic Gradient Methods
- Stochastic Meta-Descent
- Automatic Differentiation
- Gradient Approximations
General Graphs

- In General Graphs, Inference may be intractable
- Batch Models: need to approximate log(Z) in objective and marginals in gradient
- Stochastic Approaches: need marginals, but no log(Z)
Pseudo-likelihood

\[
S \leftarrow y_i w^T x_i + \sum_{j \in nei(y_i)} y_i y_j v^T x_{ij} \\
- \log(1 + \exp(S)) \\
-y_i [x_i; \sum_{j \in nei(y_i)} [y_j x_{ij}]] \sigma(S)
\]
Experiment 3

- Man-Made Structure Detection
- Images divided into 16x24 patches
- 108 training Images
- 35 Features (we used ‘full’ BFGS)
Variational Approximations

- Mean Field:

\[ F_{MF}(b_i) = - \sum_{i,j} \sum_{x_i,x_j} b_i(x_i)b_j(x_j) \log \psi_i,j(x_i,x_j) + \sum_i \sum_{x_i} b_i(x_i)[\log b_i(x_i) - \log \psi_i(x_i)] \]

- Bethe:

\[ F_\beta(b_i, b_j) = \sum_{i,j} \sum_{x_i,x_j} b_{i,j}(x_i,x_j)[\log b_{i,j}(x_i,x_j) - \log \psi_{i,j}(x_i,x_j)] \]
\[ - \sum_i (d_i - 1) \sum_{x_i} b_i(x_i)[\log b_i(x_i) - \psi_i(x_i)] \]

- Not convex, may not give descent direction

- Can SG methods escape bad gradient or local min?
Accelerated Training of CRFs with Stochastic Gradient Methods

Winkler, G. (1995). Image Analysis, Random Fields and

...
Final Notes

• For large data sets and well-behaved functions, SG methods can significantly improve training time

• SMD has better convergence properties than SG in these cases

• Reproducible Research:
  • Matlab code/data for replicating 2D experiments on-line (including mex code for MF/LBP)