# 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
  - $\mathbb{P}(Y|\boldsymbol{x})$ modeled directly
- Log-Likelihood:

$$p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) = \exp(\langle\phi(\boldsymbol{x},\boldsymbol{y}),\boldsymbol{\theta}\rangle - z(\boldsymbol{\theta}|\boldsymbol{x}))$$

- Log-Partition Function:

$$z(\boldsymbol{\theta}|\boldsymbol{x}) := \ln\sum_{\boldsymbol{y}}\exp(\langle\phi(\boldsymbol{x},\boldsymbol{y}),\boldsymbol{\theta}\rangle)$$

# 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

$$\mathcal{L}(\boldsymbol{\theta}) := \frac{||\boldsymbol{\theta}||^2}{2\sigma^2} - \sum_{i=1}^{m} [\langle \phi(\boldsymbol{x}_i, \boldsymbol{y}_i), \boldsymbol{\theta} \rangle - z(\boldsymbol{\theta}|\boldsymbol{x}_i)]$$

$$\boldsymbol{g}(\boldsymbol{\theta}) = \frac{\boldsymbol{\theta}}{\sigma^2} - \sum_{i=1}^{m} \left[\phi(\boldsymbol{x}_i, \boldsymbol{y}_i) - \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x}_i;\boldsymbol{\theta})}[\phi(\boldsymbol{x}_i, \boldsymbol{y})]\right]$$

$$\cancel{\boldsymbol{H}(\boldsymbol{\theta}) = \frac{\mathrm{I}}{\sigma^2} + \sum_{i=1}^{m} \mathrm{Cov}_{p(\boldsymbol{y}|\boldsymbol{x}_i;\boldsymbol{\theta})} \phi(\boldsymbol{x}_i, \boldsymbol{y})}$$

# Overview

- Conditional Random Fields
- $\boxed{\text{Batch Learning Methods}}$
- 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 = smallRand;
[f,g,H] = @gradientFunction(w);
do
   stepDir = H \ g
   stepLen = lineSearch(w + stepLen*stepDir)
   w = w + stepLen*stepDir
   [f,g,H] = @gradientFunction(x);
while norm(g) > optTol
```

# Quasi-Newton Method

```
w = smallRand;
B = eye;
[f,g] = @gradientFunction;
do
    stepDir = B \ g
    stepLen = lineSearch(w + stepLen*stepDir)
    update(B)
    w = w + stepLen*stepDir
    [f,g] = @gradientFunction
while norm(g) > optTol
```

# BFGS Update

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

$$B_{i+1} = B_i + \frac{yy^T}{y^T s} - \frac{B_i s s^T B_i}{s^T B_i s}$$

- Update Factorization or inverse instead of inverting B:

```
[In Matlab: R= 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, 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:

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

- Current Inverse Hessian can be computed recursively based on previous function and gradient values
- Limited Memory BFGS:
  - Compute B\g without storing Hessian approximation

```matlab
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 = smallRand;
for i = 1:maxIter
   for b = 1:maxBatch
      [f(b),g(b)] = @gradientFunction(b);
      w = w - stepSize*g(b)
   end
end
```

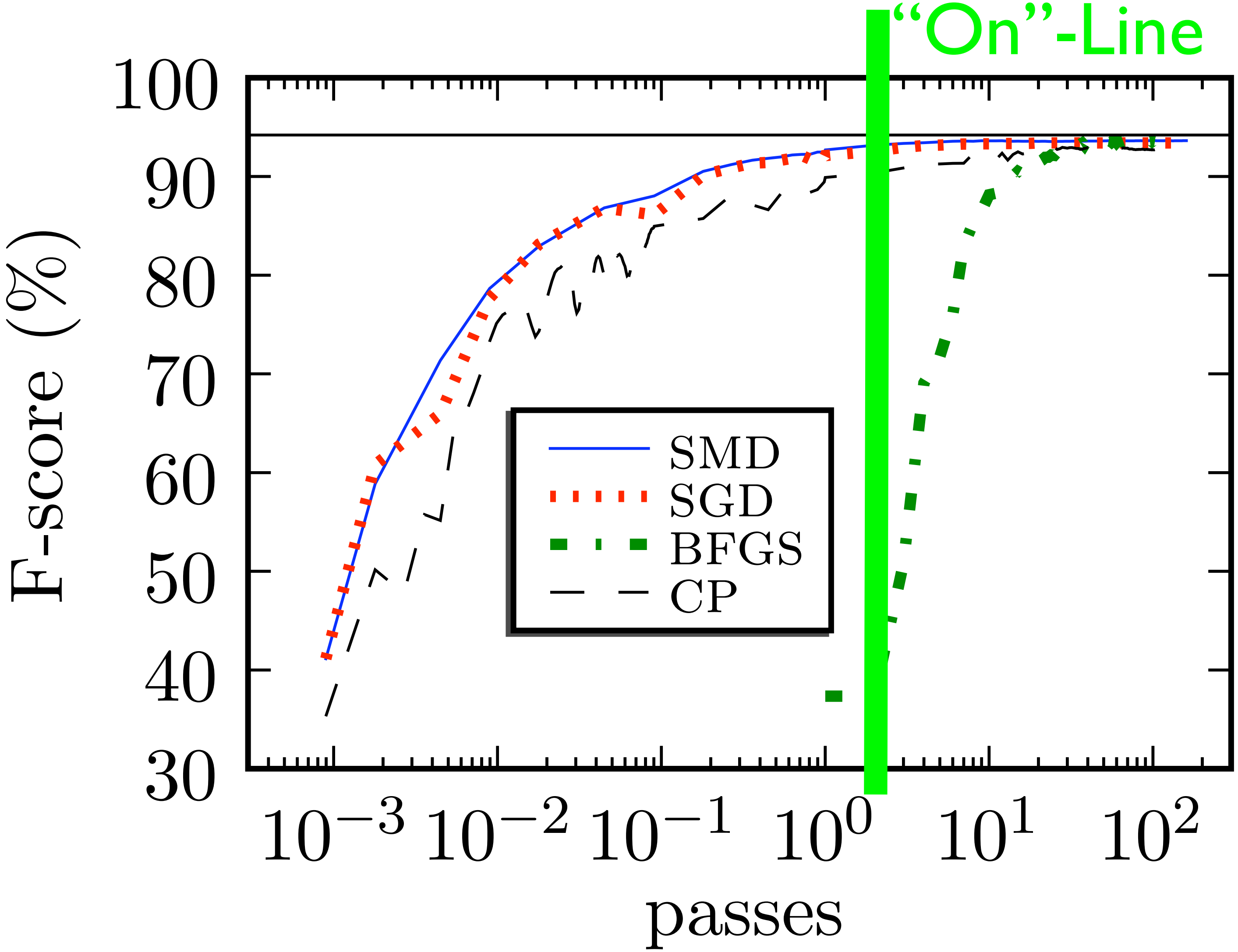
# CRF Parameter Learning

- Current Champ:
  - Quasi-Newton w/ [L-]BFGS Updating
  - Inference on all training examples
- Challenger:
  - Stochastic Gradient
  - Inference on batch of training examples

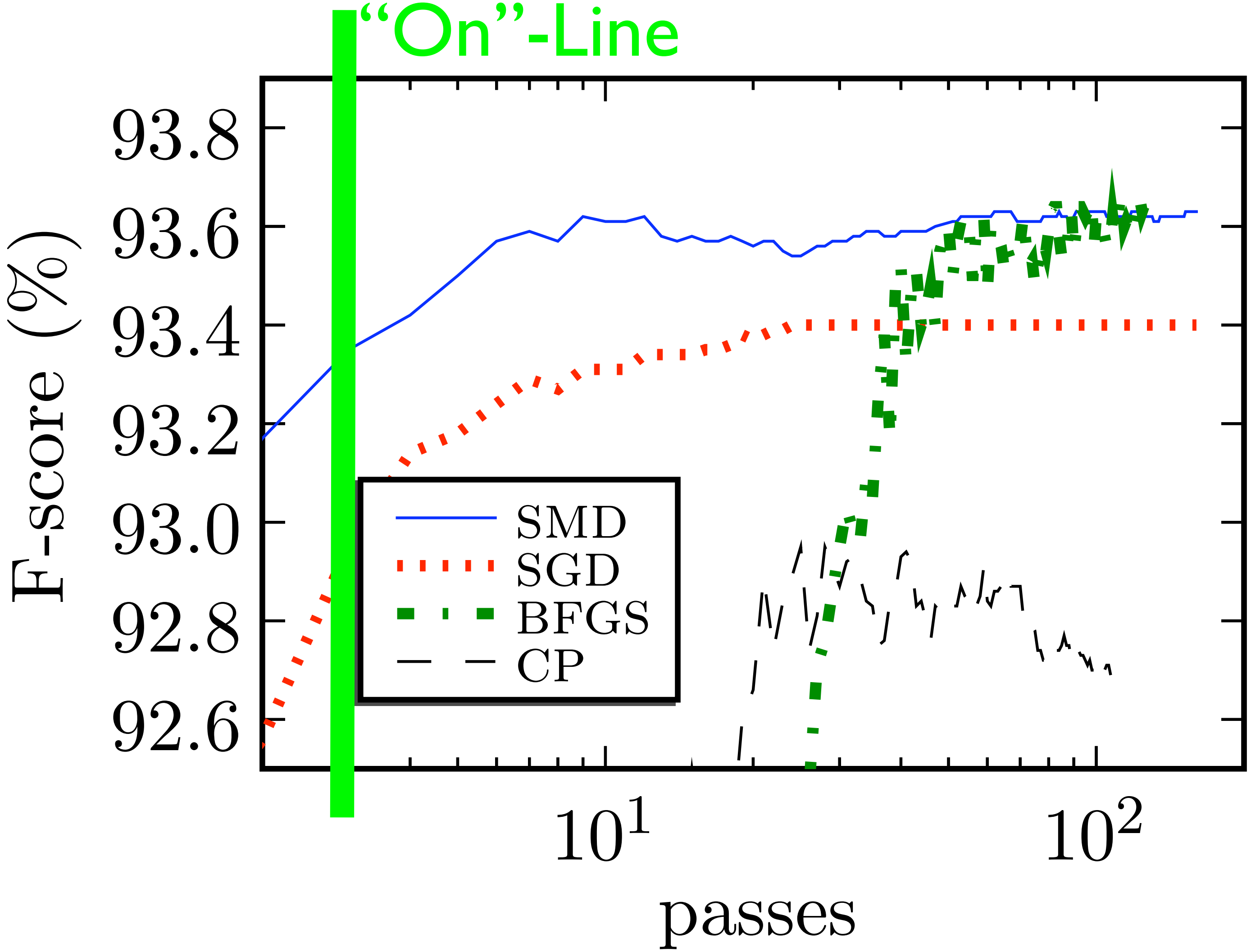
# Experiment 1

- 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)



# Learning vs. Optimization



# 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:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{\eta}_t \cdot \boldsymbol{g}_t,$$

- Update the gain multiplicatively by meta-gain (mu):

$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t \cdot \max(\tfrac{1}{2}, 1 - \mu\, \boldsymbol{g}_{t+1} \cdot \boldsymbol{v}_{t+1})$$

- Update the long-term 2nd-order dependence w/ memory (lambda):

$$\boldsymbol{v}_{t+1} = \lambda \boldsymbol{v}_t - \boldsymbol{\eta}_t \cdot (\boldsymbol{g}_t + \lambda \boldsymbol{H}_t \boldsymbol{v}_t)$$

# 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:

$$\mathrm{d}\boldsymbol{g}(\boldsymbol{\theta}) = \boldsymbol{H}(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} \qquad d\mathbf{g}(\theta) \approx \frac{\mathbf{g}(\theta + \epsilon\mathbf{d}) - \mathbf{g}(\theta)}{\epsilon}$$

- Algorithmic Differentiation: Under arithmetic assumption about gradient evaluation, can use 1 gradient evaluation and complex perturbation:

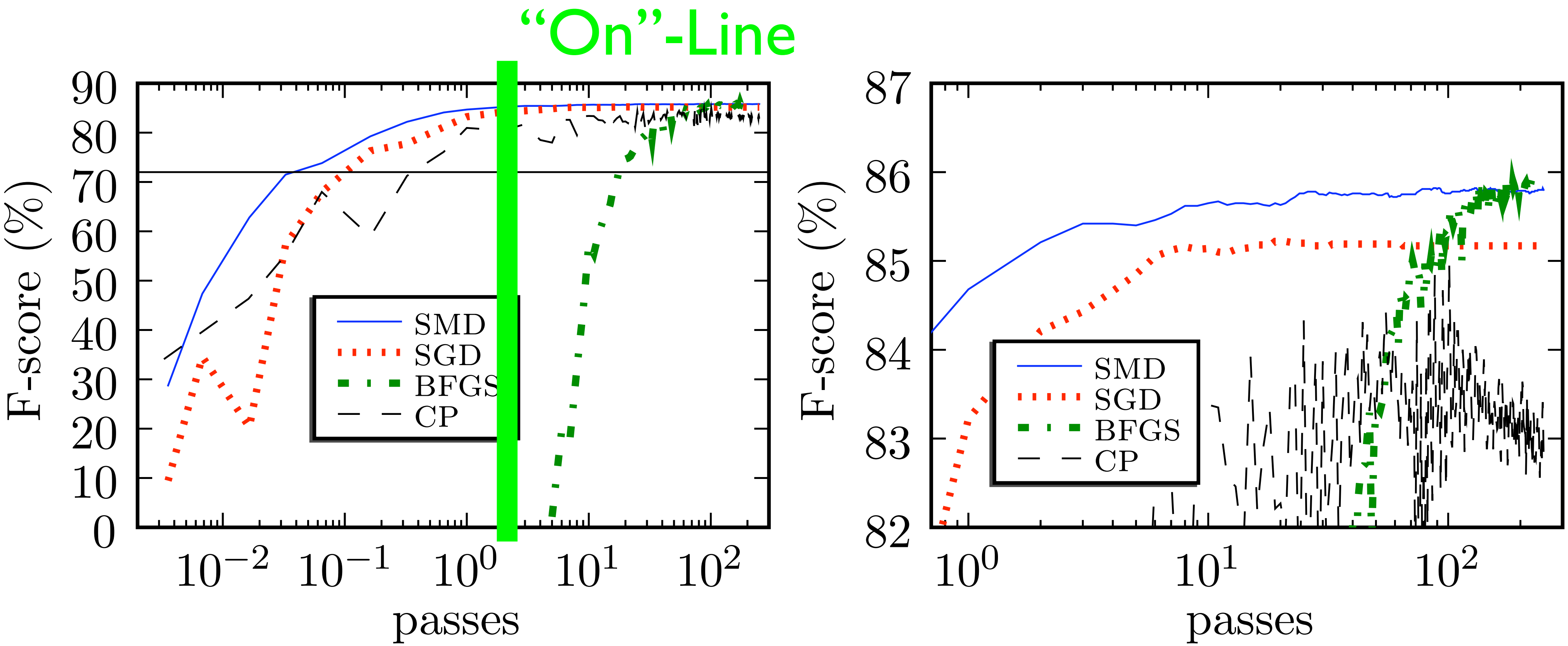
$$\boldsymbol{g}(\boldsymbol{\theta} + i\,\epsilon\,\mathrm{d}\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta}) + O(\epsilon^2) + i\,\epsilon\,\mathrm{d}\boldsymbol{g}(\boldsymbol{\theta})$$

# SMD:

```
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
F-score (%)
passes
SMD
SGD
BFGS
CP
F-score (%)
passes
SMD
SGD
BFGS
CP

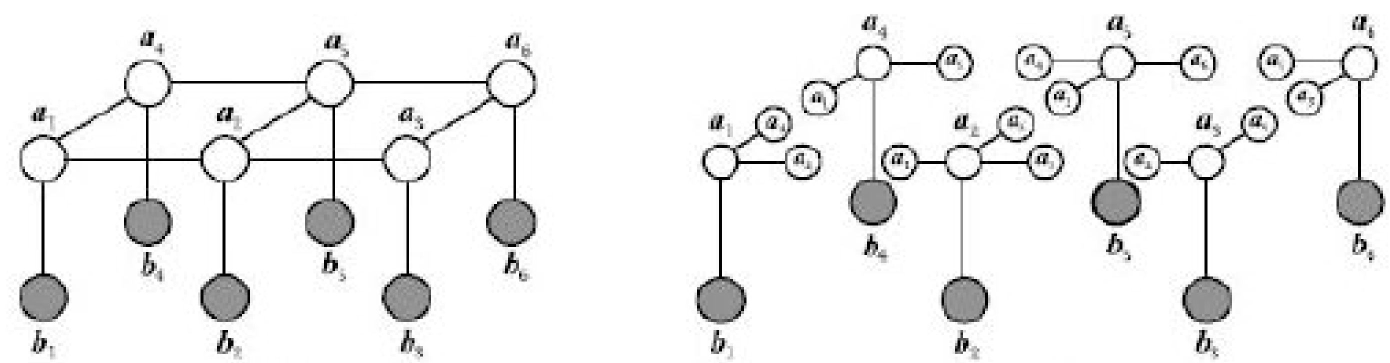
# Overview

- 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



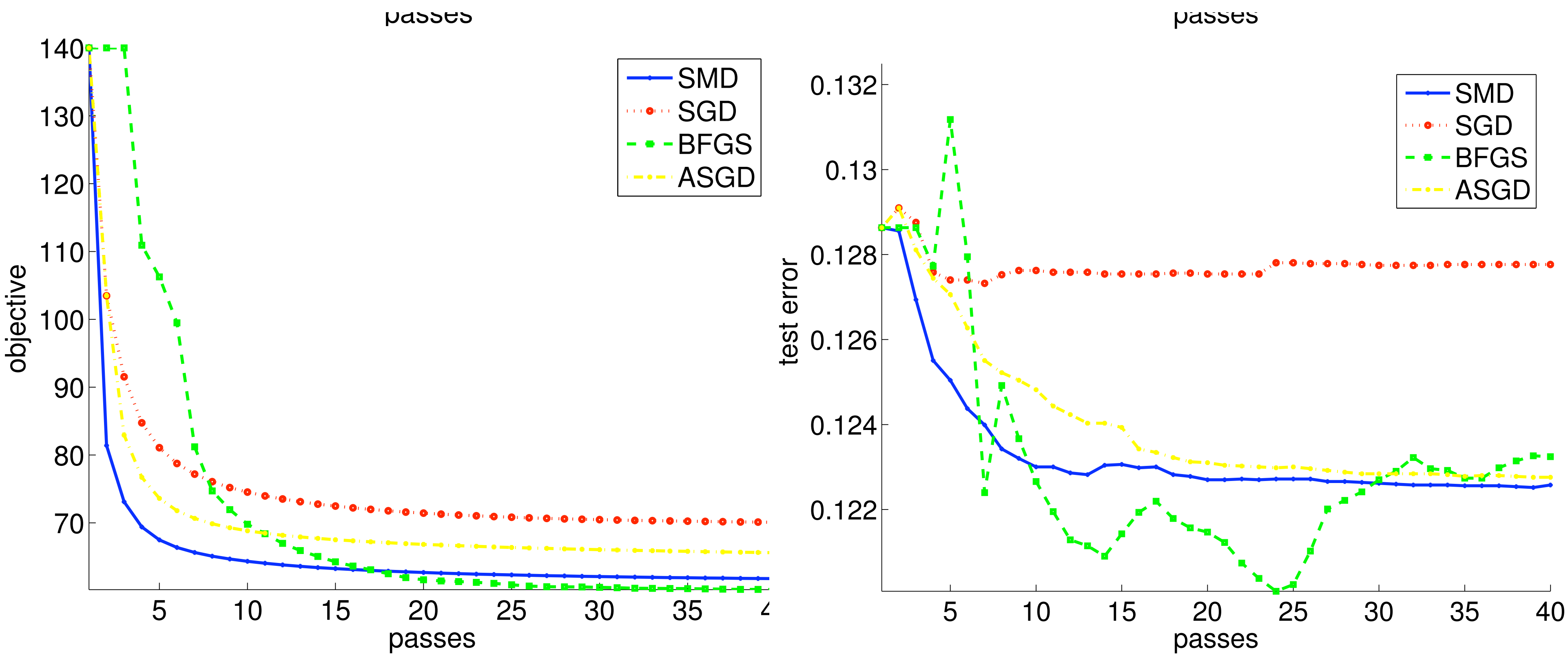
$$S \leftarrow y_i w^T x_i + \sum_{j \in nei(y_i)} y_i y_j v^T x_{ij}$$

$$-\log(1 + \exp(S))$$

$$-y_i[x_i; \sum_{j \in nei(y_i)} [y_j x_{ij}]]\sigma(S)$$

# Experiment 3

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


passes
objective
140
130
120
110
100
90
80
70
5
10
15
20
25
30
35
4
passes
SMD
SGD
BFGS
ASGD
passes
test error
0.132
0.13
0.128
0.126
0.124
0.122
5
10
15
20
25
30
35
40
passes
SMD
SGD
BFGS
ASGD

# Variational Approximations

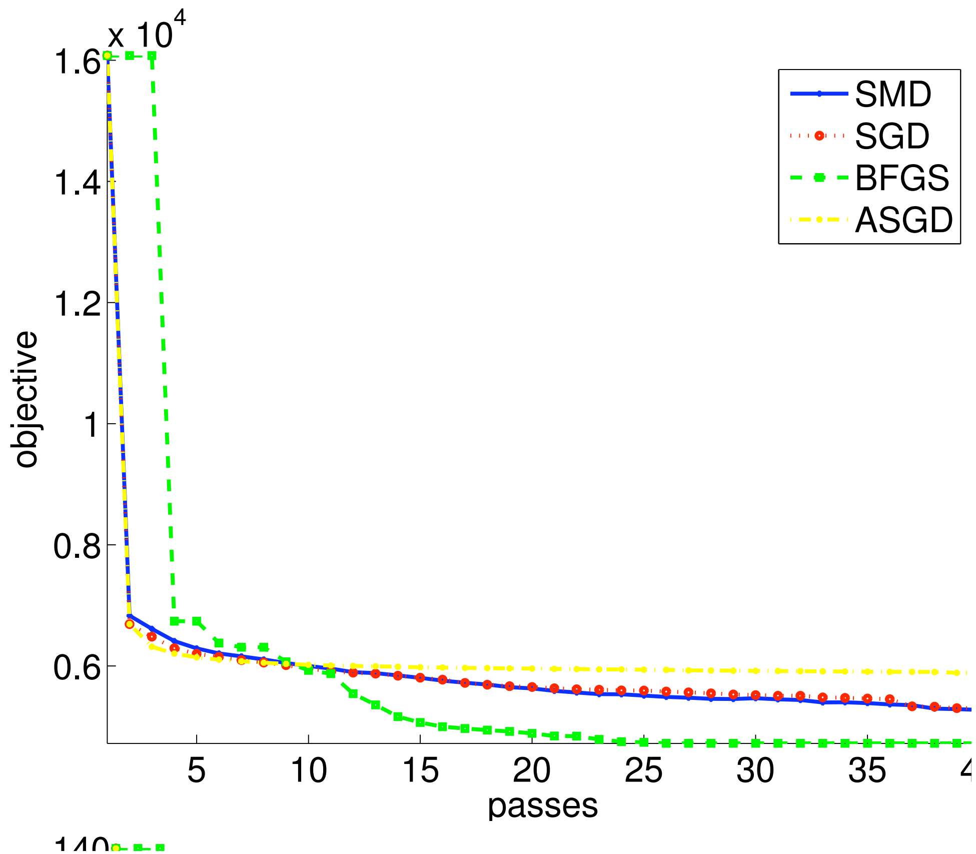
- Mean Field:

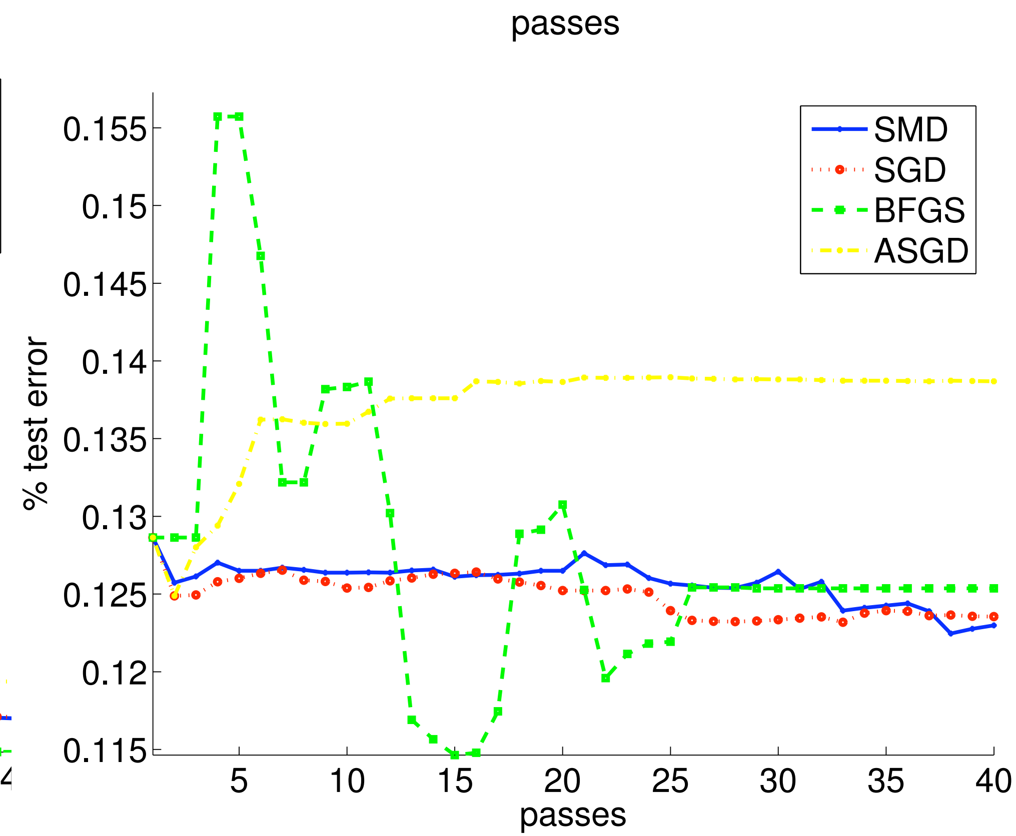
$$F_{MF}(b_i) = -\sum_{i,j} \sum_{x_i,x_j} b_i(x_i) b_j(x_j) \log \psi_{i,j}(x_i, x_j) + \sum_i \sum_{x_i} b_i(x_i)[\log b_i(x_i) - \log \psi_i(x_i)]$$

- Bethe:

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

- Not convex, may not give descent direction
- Can SG methods escape bad gradient or local min?

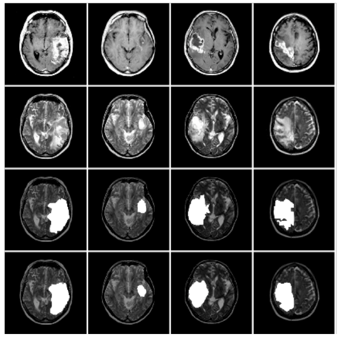

x 10^4
1.6
1.4
1.2
1
0.8
0.6
objective
5
10
15
20
25
30
35
4
passes
SMD
SGD
BFGS
ASGD
140


passes
0.155
0.15
0.145
0.14
0.135
0.13
0.125
0.12
0.115
% test error
5
10
15
20
25
30
35
40
passes
SMD
SGD
BFGS
ASGD

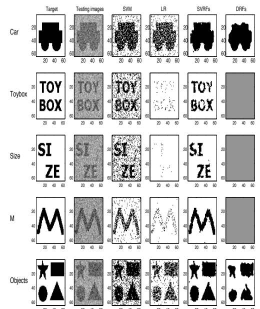
# Final Notes

- 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)

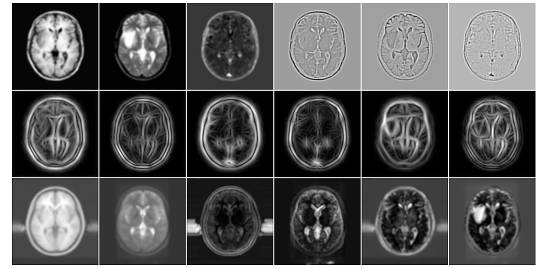
Automatic Brain Tumor Segmentation



Support Vector Random Fields




Database Textons

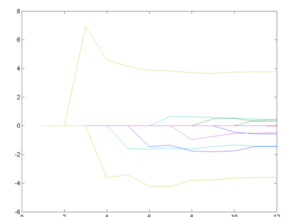


UNIVERSITY OF ALBERTA

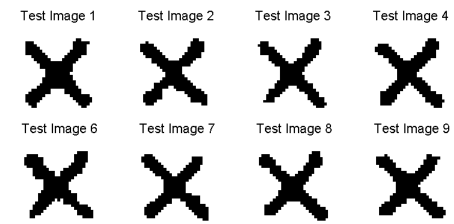
ALBERTA INGENUITY FUND

UBC

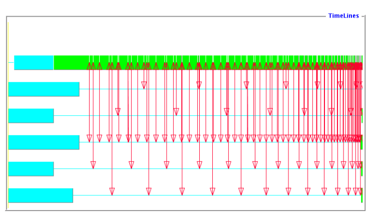
L1-Penalized Optimization



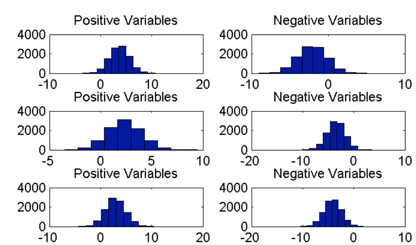
CRF Toolbox




Parallel Algorithms



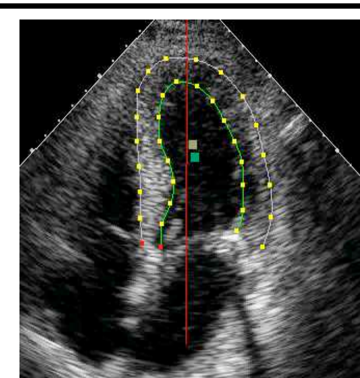

Bayesian Sparse Logistic Regression




Graphical Model Structure Learning

SIEMENS

Automated Heart Wall Abnormality Detection



"Non-Patented" Stuff