Faster Stochastic Variational Inference using Proximal-Gradient Methods with General Divergence Functions

Presented by Frank Ferraro, UMBC 691

Paper: Babanezhad, Lin, Schmidt and Sugiyama
UAI 2016
https://arxiv.org/abs/1511.00146

Adapted from slides made & presented by Frank in 2016
So far, in this class

• Probability and statistics
• Modeling considerations
  – Generative vs. conditional/discriminative
  – Directed vs. Undirected
• Loss functions
  – Classification & structured loss
  – Why MAP is the "right" thing to do
• Expectation maximization
  – Full vs. MAP
• Belief propagation
  – Exact inference:
    • forward/backward algorithm in HMMs
    • Sum-product for tree-structured graphs
  – Inexact:
    • Loopy BP for general graphs
• Variational inference
• Sampling
If you know me...

unsupervised learning of document semantics

Generative neural models for language grounding
Why This Paper?
Problem

1. Develop a model $p_\beta$

2. Define an objective $F(p_\beta)$

3. Optimize the objective
Problem

1. Develop a model $p_\beta$

2. Define an objective $F(p_\beta)$

3. Optimize the objective

\[ \beta^{(t+1)} = \beta^{(t)} + \rho * \nabla F(p_\beta) \]
Problem

1. Develop a model $p_\beta$

   What does $\theta$ represent?

   What restrictions are on $\theta$?

2. Define an objective $F(p_\beta)$

   $\beta^{(t+1)} = \beta^{(t)} + \rho \ast \nabla F(p_\beta)$

   In what sense is this step “good”?

3. Optimize the objective
Summary

Unified proximal-gradient framework that accounts for underlying characteristics of $\theta$

Stochastic ➔ run online on large corpora

Use MCMC to get closed-form updates for non-conjugate models
Proximal Gradient?

Updating parameters by (some) gradient and projecting on to a convex constraint
Proximal Gradient?

Updating parameters by (some) gradient and projecting on to a convex constraint

$$\max_{\phi} f(\phi) \text{ subject to } \phi_i \geq 0, \sum_i \phi_i = 1$$
Variational Inference

\[ \log p(y) \]

observed data (BOW counts)
Variational Inference

True objective: difficult to compute
Variational Inference

True objective: difficult to compute

Proxy objective: Easy(ier) to compute

Minimize the “difference”
Variational Inference

True objective: difficult to compute

Proxy objective: Easy(ier) to compute

Minimize the “difference”
Variational Inference

\[ \log p(y) = \log \int p_\beta(y, z) \, dz \]

- \( \beta \): hyperparameters
- \( z \): latent parameters

If \( p_\beta(z) \) is Gaussian,
\( \beta = \{\text{mean, variance}\} \)
Variational Inference

\[
\log p(y) = \log \int p_\beta(y, z) dz - q_\lambda(z)
\]

Let \( q \) be a separate distribution over \( z \)

\( \lambda \): hyperparameters for \( q \)
Variational Inference

\[ \log p(y) = \log \int p_\beta(y, z) \, dz \]

\[ = \log \int \frac{q_\lambda(z)}{q_\lambda(z)} p_\beta(y, z) \, dz \]

form of 1
Jensen’s Inequality

\[ f(\alpha v_1 + (1 - \alpha)v_2) \leq \alpha f(v_1) + (1 - \alpha)f(v_2) \]
Jensen’s Inequality

\[ f(\alpha v_1 + (1 - \alpha)v_2) \leq \alpha f(v_1) + (1 - \alpha)f(v_2) \]

Binary expectation

https://alliance.seas.upenn.edu/~cis520/dynamic/2016/wiki/uploads/Lectures/jensen.png
Variational Inference

\[ \log p(y) = \log \int p_\beta(y, z) \, dz \]

\[ = \log \int \frac{q_\lambda(z)}{q_\lambda(z)} p_\beta(y, z) \, dz \]

\[ \geq \int \log \frac{p_\beta(y, z)}{q_\lambda(z)} q_\lambda(z) \, dz \]
Variational Inference

\[ \log p(y) = \log \int p_\beta(y, z) dz \]

\[ = \log \int \frac{q_\lambda(z)}{q_\lambda(z)} p_\beta(y, z) dz \]

\[ \geq \int \log \frac{p_\beta(y, z)}{q_\lambda(z)} q_\lambda(z) dz \]

\[ = \mathbb{E}_q[\log \frac{p_\beta(y, z)}{q_\lambda(z)}] \]
Variational Inference

\[
\log p(y) = \log \int p_\beta(y, z)dz \\
= \log \int \frac{q_\lambda(z)}{q_\lambda(z)} p_\beta(y, z)dz \\
\geq \int \log \frac{p_\beta(y, z)}{q_\lambda(z)} q_\lambda(z)dz \\
= \mathbb{E}_q[\log \frac{p_\beta(y, z)}{q_\lambda(z)}] \\
= \mathcal{L}(\lambda) = \text{ELBO}
\]
Variational Inference

\[ \mathcal{L}(\lambda) = \mathbb{E}_q \left[ \log \frac{p_\beta(y, z)}{q_\lambda(z)} \right] \leq \log p(y) \]

Change the bound by changing \( \lambda \)

Maximize \( \mathcal{L}(\lambda) \)

Original model \( p_\beta \) acts only as a guide
Take 1: Gradient Ascent

\[ \lambda^{(t+1)} = \lambda^{(t)} + \rho \cdot \nabla_\lambda \mathcal{L}(\lambda) \]
Take 1: Gradient Ascent

\[ \lambda^{(t+1)} = \lambda^{(t)} + \rho \cdot \nabla_\lambda \mathcal{L}(\lambda) \]

View it as approximation of quadratic Taylor approximation

\[ \mathcal{L}(\lambda^{(t+1)}) \approx \mathcal{L}(\lambda^{(t)}) + \nabla_\lambda \mathcal{L}(\lambda)^T (\lambda^{(t+1)} - \lambda^{(t)}) + (\lambda^{(t+1)} - \lambda^{(t)})^T \frac{1}{2} \nabla^2_\lambda \mathcal{L}(\lambda) (\lambda^{(t+1)} - \lambda^{(t)}) \]
Take 1: Gradient Ascent

\[ \lambda^{(t+1)} = \lambda^{(t)} + \rho \cdot \nabla_{\lambda} \mathcal{L}(\lambda) \]

View it as approximation of quadratic Taylor approximation

\[ \mathcal{L}(\lambda^{(t+1)}) \approx \mathcal{L}(\lambda^{(t)}) + \nabla_{\lambda} \mathcal{L}(\lambda)^T (\lambda^{(t+1)} - \lambda^{(t)}) + (\lambda^{(t+1)} - \lambda^{(t)})^T \frac{1}{2} \nabla_{\lambda} \mathcal{L}(\lambda) (\lambda^{(t+1)} - \lambda^{(t)}) - \frac{1}{2\rho} \| \lambda^{(t+1)} - \lambda^{(t)} \|_2^2 \]
Take 1: Gradient Ascent

\[ \lambda^{(t+1)} = \lambda^{(t)} + \rho \cdot \nabla_{\lambda} \mathcal{L}(\lambda) \]

View it as approximation of quadratic Taylor approximation

\[ \lambda^{(t+1)} = \arg\max_{\lambda} \left[ \lambda^T \nabla \mathcal{L}(\lambda^{(t)}) - \frac{1}{2\rho} \| \lambda - \lambda^{(t)} \|_2^2 \right] \]
Take 1: Gradient Descent

\[ \lambda^{(t+1)} = \lambda^{(t)} - \rho \nabla \mathcal{L}(\lambda^{(t)}) \]

View it as approximation of quadratic Taylor approximation

\[ \lambda^{(t+1)} = \arg\min_{\lambda} \left[ -\lambda^T \nabla \mathcal{L}(\lambda^{(t)}) + \frac{1}{2\rho} \| \lambda - \lambda^{(t)} \|_2^2 \right] \]
Euclidean Distance May Not be Best

$N(0, 0.01)$ vs. $N(0.1, 0.01)$

Derived from Hoffman et al. (2013)
Euclidean Distance May Not be Best

\( N(0, 0.01) \) vs. \( N(0.1, 0.01) \)  

\( N(0, 50) \) vs. \( N(10, 50) \)

Derived from Hoffman et al. (2013)
Take 2: Natural Gradient Descent

Replace Euclidean proximity constraint with probability similarity constraint
Take 2: Natural Gradient Descent

Replace Euclidean proximity constraint with probability similarity constraint

\[
\frac{1}{2\rho} \| \lambda - \lambda^{(t)} \|_2^2
\]

\[
\frac{1}{\rho} \left[ D_{KL} \left( q(\cdot \mid \lambda) \| q(\cdot \mid \lambda^{(t)}) \right) + D_{KL} \left( q(\cdot \mid \lambda^{(t)}) \| q(\cdot \mid \lambda) \right) \right]
\]
Take 2: Natural Gradient Descent

Replace Euclidean proximity constraint with probability similarity constraint

\[
\lambda^{(t+1)} = \arg\min_{\lambda} \left[ -\lambda^T \nabla \mathcal{L}(\lambda^{(t)}) + \frac{1}{\rho} \mathcal{D}_{\text{symKL}} \left( q(\cdot \mid \lambda), q(\cdot \mid \lambda^{(t)}) \right) \right]
\]
Take 2: Natural Gradient Descent

Replace Euclidean proximity constraint with probability similarity constraint

\[ \lambda^{(t+1)} = \arg \min_{\lambda} \left[ -\lambda^T \nabla \mathcal{L}(\lambda^{(t)}) + \frac{1}{\rho} \mathcal{D}_{\text{symKL}} \left( q(\cdot \mid \lambda), q(\cdot \mid \lambda^{(t)}) \right) \right] \]

Information geometry (Amari, 1985)

\[ \lambda^{(t+1)} = \lambda^{(t)} - \rho \left[ \underbrace{\mathcal{I}(\lambda)^{-1}}_{\text{inverse Fisher information matrix}} \underbrace{\nabla \mathcal{L}(\lambda^{(t)})}_{\text{Euclidean gradient}} \right] \]
Take 2: Natural Gradient Descent

\[ \lambda^{(t+1)} = \lambda^{(t)} - \rho \begin{bmatrix} \text{inverse Fisher information matrix} \\
\mathcal{I}(\lambda)^{-1} \\
\nabla \mathcal{L}(\lambda^{(t)}) \end{bmatrix} \]

very nice cancelations/simplifications in conditionally conjugate exponential family models (e.g., LDA)

Empirically, faster and more stable convergence than gradient descent

Conditionally conjugate models: closed form updates

Conjugacy requirements can be limiting
(Euclidean) Gradient Descent

$$\lambda^{(t+1)} = \lambda^{(t)} - \rho \nabla \mathcal{L}(\lambda^{(t)})$$

Natural Gradient Descent

$$\lambda^{(t+1)} = \lambda^{(t)} - \rho \left[ \mathcal{I}(\lambda)^{-1} \nabla \mathcal{L}(\lambda^{(t)}) \right]$$
Take 3: “Trust Regions”  
(Theis and Hoffman, 2015)

Replace Euclidean proximity constraint with (asymmetric) probability-based loss
Take 3: “Trust Regions”  
(Theis and Hoffman, 2015)  
Replace Euclidean proximity constraint with (asymmetric) probability-based loss

\[
\frac{1}{2\rho} \| \lambda - \lambda^{(t)} \|_2^2
\]

\[
\frac{1}{\rho} \mathcal{D}_{KL} \left( q(\cdot \mid \lambda) \| q(\cdot \mid \lambda^{(t)}) \right)
\]
Take 3: “Trust Regions”  
(Theis and Hoffman, 2015)

Replace Euclidean proximity constraint with probability similarity constraint

$$
\lambda^{(t+1)} = \arg\min_{\lambda} \left[ -\lambda^T \nabla \mathcal{L}(\lambda^{(t)}) + \frac{1}{\rho} \mathcal{D}_{KL} \left( q(\cdot | \lambda) \| q(\cdot | \lambda^{(t)}) \right) \right]
$$
Take 3: “Trust Regions”  
(Theis and Hoffman, 2015)

Replace Euclidean proximity constraint with probability similarity constraint

\[ \lambda^{(t+1)} = \arg\min_{\lambda} \left[ -\lambda^T \nabla \mathcal{L}(\lambda^{(t)}) + \frac{1}{\rho} \mathcal{D}_{KL} \left( q(\cdot | \lambda) \| q(\cdot | \lambda^{(t)}) \right) \right] \]

Empirically: faster and more stable convergence than 
**natural** gradient descent

Theoretically: generalizes natural gradient descent

Numerical optimization generally required
(Euclidean) Gradient Descent

\[ \lambda^{(t+1)} = \lambda^{(t)} - \rho \nabla \mathcal{L}(\lambda^{(t)}) \]

Natural Gradient Descent

\[ \lambda^{(t+1)} = \lambda^{(t)} - \rho \left[ \mathcal{I}(\lambda)^{-1} \nabla \mathcal{L}(\lambda^{(t)}) \right] \]

Trust Region Descent

\[ \lambda^{(t+1)} = \arg\min_{\lambda} \left[ -\nabla \mathcal{L}(\lambda^{(t)}) + \frac{1}{\rho} \mathcal{D}_{\text{KL}}(q(\cdot \mid \lambda) \| q(\cdot \mid \lambda^{(t)})) \right] \]
Take 3.5: Splitting of Trust Regions
(Khan et al., 2015)

Treat “easy” and “hard” parts of objective differently

\[-\mathcal{L}(\lambda) = \mathbb{E} \left[ \log \frac{p}{q} \right] \]
\[= \mathbb{E} [\tilde{r}_{\text{hard}} + \tilde{r}_{\text{easy}}] \]
\[= \mathbb{E} [\tilde{r}_{\text{hard}}] + \mathbb{E} [\tilde{r}_{\text{easy}}] \]
\[= f(\lambda) + h(\lambda) \]
Take 3.5: Splitting of Trust Regions
(Khan et al., 2015)

Treat “easy” and “hard” parts of objective differently

\[-\mathcal{L}(\lambda) = \mathbb{E}\left[ \log \frac{p}{q} \right] \]
\[= \mathbb{E}[\tilde{r}_{\text{hard}} + \tilde{r}_{\text{easy}}] \]
\[= \mathbb{E}[\tilde{r}_{\text{hard}}] + \mathbb{E}[\tilde{r}_{\text{easy}}] \]

\[f(\lambda) \quad h(\lambda)\]

differentiable, with bounded gradient changes

convex
Take 3.5: Splitting of Trust Regions (Khan et al., 2015)

Treat “easy” and “hard” parts of objective differently... and linearly approximate the hard part (expectation and all)

\[-\mathcal{L}(\lambda) \approx f(\lambda^{(t)}) + \nabla_\lambda f(\lambda^{(t)})^T \left( \lambda - \lambda^{(t)} \right) + h(\lambda)\]
Take 3.5: Splitting of Trust Regions (Khan et al., 2015)

Treat “easy” and “hard” parts of objective differently

\[
\lambda^{(t+1)} = \arg\min_\lambda [\lambda^T \nabla f(\lambda^{(t)}) + h(\lambda) \frac{1}{\rho} \mathcal{D}_{KL}(q(\cdot \mid \lambda) \| q(\cdot \mid \lambda^{(t)}))] 
\]

Able to handle non-conjugate models

Numerical optimization generally required
(Euclidean) Gradient Descent

$$\lambda^{(t+1)} = \lambda^{(t)} - \rho \nabla \mathcal{L}(\lambda^{(t)})$$

Natural Gradient Descent

$$\lambda^{(t+1)} = \lambda^{(t)} - \rho \left[ \mathcal{I}(\lambda)^{-1} \nabla \mathcal{L}(\lambda^{(t)}) \right]$$

Trust Region Descent

$$\lambda^{(t+1)} = \arg\min_{\lambda} \left[ -\lambda^\top \nabla \mathcal{L}(\lambda^{(t)}) + \frac{1}{\rho} \mathcal{D}_{KL}(q(\cdot \mid \lambda) \| q(\cdot \mid \lambda^{(t)})) \right]$$

Linearized Trust Region Descent

$$\lambda^{(t+1)} = \arg\min_{\lambda} \left[ \lambda^\top \nabla f(\lambda^{(t)}) + h(\lambda) + \frac{1}{\rho} \mathcal{D}_{KL}(q(\cdot \mid \lambda) \| q(\cdot \mid \lambda^{(t)})) \right]$$
Take 4: Mirror Descent/Bregman Divergence

Replace Euclidean proximity constraint with arbitrary (convex, non-negative, ...) Bregman divergences $G$

$$\lambda^{(t+1)} = \arg\min_\lambda \left[ -\lambda^T \nabla \mathcal{L}(\lambda^{(t)}) + \frac{1}{\rho} \mathcal{D}_G \left( q(\cdot \mid \lambda) \| q(\cdot \mid \lambda^{(t)}) \right) \right]$$
Bregman Divergence

Convex, non-negative generating function $G \Rightarrow$ divergence $D_G$

Basic idea: Evaluate a first-order Taylor approximation of $G$ around $y$ at $x$
Bregman Divergence

Convex, non-negative generating function $G \rightarrow$ divergence $D_G$

Basic idea: Evaluate a first-order Taylor approximation of $G$
around $y$ at $x$

$$D_G(x, y) = G(x) - G(y) - \langle \nabla G(y), x - y \rangle$$
Bregman Divergence

Convex, non-negative generating function $G \Rightarrow$ divergence $D_G$

$$D_G(x, y) = G(x) - G(y) - \langle \nabla G(y), x - y \rangle$$

Euclidean distance

$$G(x) = x^2 \Rightarrow D_G(x, y) = \| x - y \|_2^2$$

Generalized KL divergence

$$G(x) = x^T (\log x - 1) \Rightarrow D_G(x, y) = x^T \log \frac{x}{y} - 1^T (y - x)$$
(Euclidean) Gradient Descent

\[
\lambda^{(t+1)} = \lambda^{(t)} - \rho \nabla \mathcal{L}(\lambda^{(t)})
\]

Natural Gradient Descent

\[
\lambda^{(t+1)} = \lambda^{(t)} - \rho \left[ \mathcal{I}(\lambda)^{-1} \nabla \mathcal{L}(\lambda^{(t)}) \right]
\]

Trust Region Descent

\[
\lambda^{(t+1)} = \arg \min_{\lambda} \left[ -\lambda^\top \nabla \mathcal{L}(\lambda^{(t)}) + \frac{1}{\rho} \mathcal{D}_{\text{KL}} (q(\cdot | \lambda) \| q(\cdot | \lambda^{(t)})) \right]
\]

Linearized Trust Region Descent

\[
\lambda^{(t+1)} = \arg \min_{\lambda} \left[ \lambda^\top \nabla f(\lambda^{(t)}) + h(\lambda) + \frac{1}{\rho} \mathcal{D}_{\text{KL}} (q(\cdot | \lambda) \| q(\cdot | \lambda^{(t)})) \right]
\]

Bregman Divergence/Mirror Descent

\[
\lambda^{(t+1)} = \arg \min_{\lambda} \left[ -\lambda^\top \nabla \mathcal{L}(\lambda^{(t)}) + \frac{1}{\rho} \mathcal{D}_G (q(\cdot | \lambda) \| q(\cdot | \lambda^{(t)})) \right]
\]
Take 5: Proximal-Gradient SVI

Treat “easy” and “hard” parts of objective differently...
in an online (stochastic) setting...
with different geometries

\[-\mathcal{L}(\lambda) \approx f(\lambda^{(t)}) + \nabla_{\lambda} f(\lambda^{(t)})^T \left( \lambda - \lambda^{(t)} \right) + h(\lambda)\]

\[\lambda^{(t+1)} = \operatorname{argmin}_\lambda \left[ \lambda^T \nabla f(\lambda^{(t)}) + h(\lambda) + \frac{1}{\rho} \mathcal{D} \left( q(\cdot | \lambda) \| q(\cdot | \lambda^{(t)}) \right) \right] \]
Take 5: Proximal-Gradient SVI

Treat “easy” and “hard” parts of objective differently...
   in an online (stochastic) setting...
   with different geometries...
   in non-conjugate models

\[-\mathcal{L}(\lambda) \approx f(\lambda^{(t)}) + \nabla_{\lambda} f(\lambda^{(t)})^T \left( \lambda - \lambda^{(t)} \right) + h(\lambda)\]

\[\lambda^{(t+1)} = \arg\min_{\lambda} \left[ \lambda^T \nabla \hat{f}(\lambda^{(t)}) + h(\lambda) + \frac{1}{\rho} \mathcal{D} \left( q(\cdot | \lambda) \| q(\cdot | \lambda^{(t)}) \right) \right]\]

sample (and hope variance isn’t too bad)
Take 5: Proximal-Gradient SVI

$$\lambda^{(t+1)} = \arg\min_\lambda [\lambda^T \nabla \hat{f}(\lambda^{(t)}) + h(\lambda) + \frac{1}{\rho} \mathcal{D} \left( q(\cdot \mid \lambda) \| q(\cdot \mid \lambda^{(t)}) \right)]$$

If $h = 0$, $G(x) = x^2 \Rightarrow$ (stochastic) gradient descent

If $h = 0$, $D =$ symmetric KL $\Rightarrow$ (stochastic) natural gradient descent

If $f = 0$, $D = \text{KL} \Rightarrow$ Theis and Hoffman’s “trust region” (for certain $q$)
Requirements on Divergences

\[ \mathcal{D} \left( q(\cdot \mid \lambda) \| q(\cdot \mid \lambda^{(t)}) \right) \geq 0 \]

\[ (\lambda - \lambda^{(t)})^T \nabla_\lambda \mathcal{D} \left( q(\cdot \mid \lambda) \| q(\cdot \mid \lambda^{(t)}) \right) \geq \alpha \| \lambda - \lambda^{(t)} \|^2 \]

changes in the divergence adequately reflect changes in the inputs (divergence must be strongly convex)
Results

1. (binary) Gaussian process classification on three datasets

2. (marginalized) topic model
GP Classification

GD = batch gradient descent
PG = batch prox-gradient
PG-MC = batch prox-gradient w/ approximated gradients
PG-SVI = stochastic gradients
GP Classification
Correlated Topic Model

NIPS 10 topics

AP 10 topics

Test Log-Loss vs Seconds

PG-SVI
PG-SVI-MF
Delta
MF
Laplace
Summary

Unified proximal-gradient framework that accounts for underlying characteristics of $\theta$

Stochastic $\Rightarrow$ run online on large corpora

Use MCMC to get closed-form updates for non-conjugate models