Reducing CRF training to a series of (possibly nonlinear) logistic regression problems.

Justin Domke

National ICT Australia / Australia National University
Outline

1. Overview
2. Loss Functions
3. Joint Learning/Inference
4. Logistic Regression
5. Experiments
   - Binary Denoising
   - Horses
6. Conclusions
Structured Prediction

\[ y^* = \arg \max_y F(x, y) \]
\[ x \in \mathbb{R}^N \]
\[ y \in \{1, \ldots, L\}^M \]
Structured Prediction

\[ y^* = \arg\max_y F(x, y) \]
\[ x \in \mathbb{R}^N \]
\[ y \in \{1, \ldots, L\}^M \]

Prototypically, \( x \) is an image, \( y \) is a discrete labeling.
Structured Prediction

Most commonly, $F$ can be written in linear form

$$F(x, y) = \sum_{\alpha} w^T \Phi_{\alpha}(x, y_{\alpha}),$$
Structured Prediction

Most commonly, $F$ can be written in linear form

$$F(x, y) = \sum_{\alpha} w^T \Phi_{\alpha}(x, y_{\alpha}),$$

$$\{\alpha\} = \{ \}$$
Structured Prediction

Most commonly, $F$ can be written in linear form

$$F(x, y) = \sum_{\alpha} w^T \Phi_{\alpha}(x, y_{\alpha}),$$

$$\{\alpha\} = \{\{1\}, \ldots, \{M\},$$
Structured Prediction

Most commonly, $F$ can be written in linear form

$$F(x, y) = \sum_{\alpha} w^T \Phi_\alpha(x, y_\alpha),$$

$$\{\alpha\} = \{\{1\}, ..., \{M\}, \{1, 2\}, ..., \{M - 1, M\}\}$$
Structured Learning

How to pick $F$?
Intuitively, given some dataset $(x^1, y^1), \ldots, (x^N, y^N)$, $F$ such that

$$y^k \approx \arg\max_y F(x^k, y).$$
Structured Learning

How to pick $F$?
Intuitively, given some dataset $(x^1, y^1), ..., (x^N, y^N)$, $F$ such that

$$y^k \approx \arg \max_y F(x^k, y).$$

Problem: when $F$ changes, maximizer changes
Linear vs. Nonlinear

Most commonly, $F$ can be written in linear form

$$F(x, y) = \sum_{\alpha} w^T \Phi_{\alpha}(x, y_{\alpha}),$$

with learning problem being to select $w \in \mathbb{R}^n$. 
Linear vs. Nonlinear

Most commonly, $F$ can be written in linear form

$$F(x, y) = \sum_{\alpha} w^T \Phi_\alpha(x, y_\alpha),$$

with learning problem being to select $w \in \mathbb{R}^n$.

This talk: allow $F$ to be some function like

$$F(x, y) = \sum_{\alpha} f_\alpha(x, y_\alpha),$$

with learning problem being to select $f_\alpha \in \mathcal{F}_\alpha$. 
Why Nonlinear?

Usual strategy for vision:

1. Fit some fancy classifier to univariate terms.
   1. Linear with highly engineered features
   2. Ensembles of trees (Shotton et al., 2009; Gould et al., 2008; Xiao and Quan, 2009; Ladicky et al., 2009; Winn and Shotton, 2006; Schroff et al., 2008)
   3. Neural Nets (He et al., 2004; Silberman and Fergus, 2011)

2. Freeze those, fit linear energy to pairwise terms
   1. Possibly use fancy classifier as a feature (Nowozin et al., 2011)
Main Result

Previous work: Alternate between:

1. Message passing updates $\{\lambda^k\}$

2. Gradient descent updates to $F \leftarrow$ assumed linear
Main Result

Previous work: Alternate between:
1. Message passing updates \( \{\lambda^k\} \)
2. Gradient descent updates to \( F \leftarrow \) assumed linear

What I will talk about:
1. Message passing updates to \( \{\lambda^k\} \)
2. Solve logistic regression problem to update \( F \)
   1. Include “bias” term that depends on \( \{\lambda^k\} \)
Main Result

Previous work: Alternate between:

1. Message passing updates $\{\lambda^k\}$
2. Gradient descent updates to $F \leftarrow$ assumed linear

What I will talk about:

1. Message passing updates to $\{\lambda^k\}$
2. Solve logistic regression problem to update $F$
   
   1. Include “bias” term that depends on $\{\lambda^k\}$

Pros:

1. Optimize $F$ “all the way” for fixed $\{\lambda^k\}$.
2. Use any function class we can fit a logistic loss over
Another Way of Looking at It

\[ y^* = \arg \max_y F(x, y), \quad F(x, y) = \sum_{\alpha} f_{\alpha}(x, y_{\alpha}). \]
Another Way of Looking at It

\[
y^* = \arg\max_y F(x, y), \quad F(x, y) = \sum_{\alpha} f_{\alpha}(x, y_{\alpha}).
\]

Piecewise training— ignore messages, train each \( f_{\alpha} \) as if the rest of the graph didn’t exist. (Sutton and McCallum, 2005)
Another Way of Looking at It

\[ y^* = \arg \max_y F(x, y), \quad F(x, y) = \sum_{\alpha} f_\alpha(x, y_\alpha). \]

Piecewise training—ignore messages, train each \( f_\alpha \) as if the rest of the graph didn’t exist. (Sutton and McCallum, 2005)

Can think of this talk as

1. Train piecewise style
2. Do message-passing inference
3. Bias piecewise problems using messages, repeat
1. Overview

2. Loss Functions

3. Joint Learning/Inference

4. Logistic Regression

5. Experiments
   - Binary Denoising
   - Horses

6. Conclusions
Given \{ (x^k, y^k) \} want to minimize \( R(F) = \sum_k l(x^k, y^k; F) \).
Structured SVM

Given \{ (x^k, y^k) \} want to minimize \( R(F) = \sum_k l(x^k, y^k; F) \).

Original (intractable) loss function

\[
l_0(x^k, y^k; F) = -F(x^k, y^k) + \max_{y \in Y} F(x^k, y)
\]
Structured SVM

Given \( \{(x^k, y^k)\} \) want to minimize \( R(F) = \sum_k l(x^k, y^k; F) \).

Original (intractable) loss function

\[
l_0(x^k, y^k; F) = -F(x^k, y^k) + \max_{y \in \mathcal{Y}} F(x^k, y) + \Delta(y^k, y)
\]
Structured SVM

Given \( \{(x^k, y^k)\} \) want to minimize \( R(F) = \sum_k l(x^k, y^k; F) \).

Original (intractable) loss function

\[
l_0(x^k, y^k; F) = -F(x^k, y^k) + \max_{y \in \mathcal{Y}} F(x^k, y) + \Delta(y^k, y)
\]

Trouble: \( \mathcal{Y} \) is big, discrete.
Structured SVM

Given \( \{(x^k, y^k)\} \) want to minimize \( R(F) = \sum_k l(x^k, y^k; F) \).

Original (intractable) loss function

\[
l_0(x^k, y^k; F) = -F(x^k, y^k) + \max_{y \in \mathcal{Y}} F(x^k, y) + \Delta(y^k, y)
\]

Trouble: \( \mathcal{Y} \) is big, discrete.

Solution: Relax \( \rightarrow \) Second (tractable) loss function

\[
l_1(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} F(x^k, \mu) + \Delta(y^k, \mu)
\]
Structured SVM

\[ l_0(x^k, y^k; F) = -F(x^k, y^k) + \max_{y \in \mathcal{Y}} F(x^k, y) + \Delta(y^k, y) \]

\[ l_1(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} F(x^k, \mu) + \Delta(y^k, \mu) \]

A couple details:
Structured SVM

\[ l_0(x^k, y^k; F) = -F(x^k, y^k) + \max_{y \in Y} F(x^k, y) + \Delta(y^k, y) \]

\[ l_1(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} F(x^k, \mu) + \Delta(y^k, \mu) \]

A couple details:

1. What is \( \mu \)?
Structured SVM

\[ l_0(x^k, y^k; F) = -F(x^k, y^k) + \max_{y \in \mathcal{Y}} F(x^k, y) + \Delta(y^k, y) \]

\[ l_1(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} F(x^k, \mu) + \Delta(y^k, \mu) \]

A couple details:

1. What is \( \mu \)? Answer \( \mu = \{\mu_\alpha(y_\alpha)\} \).
Structured SVM

\[ l_0(x^k, y^k; F) = -F(x^k, y^k) + \max_{y \in \mathcal{Y}} F(x^k, y) + \Delta(y^k, y) \]

\[ l_1(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} F(x^k, \mu) + \Delta(y^k, \mu) \]

A couple details:

1. What is \( \mu \)? Answer \( \mu = \{\mu_\alpha(y_\alpha)\} \).
2. What is \( \mathcal{M} \)?
Structured SVM

\[ l_0(x^k, y^k; F) = -F(x^k, y^k) + \max_{y \in \mathcal{Y}} F(x^k, y) + \Delta(y^k, y) \]

\[ l_1(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} F(x^k, \mu) + \Delta(y^k, \mu) \]

A couple details:

1. What is \( \mu \)? Answer \( \mu = \{\mu_\alpha(y_\alpha)\} \).
2. What is \( \mathcal{M} \)?
   1. If \( \mathcal{M} = \{\mu : \mu_\alpha(y_\alpha) \in \{0, 1\}, \sum_{y_\alpha} \mu_\alpha(y_\alpha) = 1, \mu_{\alpha\beta}(x_\beta) = \mu_\beta(x_\beta)\} \), \( l_0 = l_1 \)

\[ \mu_{\alpha\beta}(x_\beta) = \sum_{x_\alpha \setminus \beta} \mu_\alpha(x_\alpha) \]
Structured SVM

\[
\begin{align*}
l_0(x^k, y^k; F) &= -F(x^k, y^k) + \max_{y \in \mathcal{Y}} F(x^k, y) + \Delta(y^k, y) \\
l_1(x^k, y^k; F) &= -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} F(x^k, \mu) + \Delta(y^k, \mu)
\end{align*}
\]

A couple details:

1. What is \( \mu \)? Answer: \( \mu = \{ \mu_\alpha(y_\alpha) \} \).

2. What is \( \mathcal{M} \)?

   1. If \( \mathcal{M} = \{ \mu : \mu_\alpha(y_\alpha) \in \{0, 1\}, \sum_{y_\alpha} \mu_\alpha(y_\alpha) = 1, \mu_{\alpha\beta}(x_\beta) = \mu_\beta(x_\beta) \}, l_0 = l_1 \)

   \[ \mu_{\alpha\beta}(x_\beta) = \sum_{x_\alpha \setminus \beta} \mu_\alpha(x_\alpha) \]

   2. Answer: Instead use a linear relaxation

\[
\mathcal{M} = \{ \mu | \mu_\alpha(y_\alpha) \in [0, 1], \sum_{y_\alpha} \mu_\alpha(y_\alpha) = 1, \mu_{\alpha\beta}(y_\beta) = \mu_\beta(y_\beta) \}.
\]
Structured SVM

\[ l_0(x^k, y^k; F) = -F(x^k, y^k) + \max_{y \in \mathcal{Y}} F(x^k, y) + \Delta(y^k, y) \]
\[ l_1(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} F(x^k, \mu) + \Delta(y^k, \mu) \]

A couple details:

1. What is \( \mu \)? Answer \( \mu = \{\mu_\alpha(y_\alpha)\} \).
2. What is \( \mathcal{M} \)?
   - If \( \mathcal{M} = \{\mu : \mu_\alpha(y_\alpha) \in \{0, 1\}, \sum_{y_\alpha} \mu_\alpha(y_\alpha) = 1, \mu_{\alpha\beta}(x_\beta) = \mu_\beta(x_\beta)\} \), \( l_0 = l_1 \)
   - \( \mu_{\alpha\beta}(x_\beta) = \sum_{x_\alpha \setminus \beta} \mu_\alpha(x_\alpha) \)

   Answer: Instead use a linear relaxation

   \[ \mathcal{M} = \{\mu | \mu_\alpha(y_\alpha) \in [0, 1], \sum_{y_\alpha} \mu_\alpha(y_\alpha) = 1, \mu_{\alpha\beta}(y_\beta) = \mu_\beta(y_\beta)\} \].
3. What is \( F(x^k, \mu), \Delta(y^k, \mu) \)?
Structured SVM

\[ l_0(x^k, y^k; F) = -F(x^k, y^k) + \max_{y \in Y} F(x^k, y) + \Delta(y^k, y) \]
\[ l_1(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} F(x^k, \mu) + \Delta(y^k, \mu) \]

A couple details:

1. What is \( \mu \)? Answer: \( \mu = \{ \mu_\alpha(y_\alpha) \} \).
2. What is \( \mathcal{M} \)?
   1. If \( \mathcal{M} = \{ \mu : \mu_\alpha(y_\alpha) \in \{0, 1\}, \sum_{y_\alpha} \mu_\alpha(y_\alpha) = 1, \mu_{\alpha\beta}(x_\beta) = \mu_{\beta}(x_\beta) \} \), \( l_0 = l_1 \)
   2. Answer: Instead use a linear relaxation
      \[ \mathcal{M} = \{ \mu | \mu_\alpha(y_\alpha) \in [0, 1], \sum_{y_\alpha} \mu_\alpha(y_\alpha) = 1, \mu_{\alpha\beta}(y_\beta) = \mu_{\beta}(y_\beta) \} \].
3. What is \( F(x^k, \mu), \Delta(y^k, \mu) \)? Answer:
   \[ F(x^k, \mu) = \sum_{\alpha} \sum_{y_\alpha} f(x^k, y_\alpha) \mu(y_\alpha), \Delta(y^k, \mu) = \sum_{\alpha} \sum_{y_\alpha} \Delta_\alpha(y^k_\alpha, y_\alpha) \mu(y_\alpha) \]
Change of Notation

Our relaxed loss:

\[
l_1(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} F(x^k, \mu) + \Delta(y^k, \mu)
\]

Can also be written as:

\[
l_1(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} \theta_F^k \cdot \mu
\]

\[
\theta_F^k(y_\alpha) = f_\alpha(x^k, y_\alpha) + \Delta_\alpha(y_\alpha^k, y_\alpha)
\]
Entropy Smoothing

\[ l_1(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} \theta^k_F \cdot \mu \]

\[ \theta^k_F(y_\alpha) = f_\alpha(x^k, y_\alpha) + \Delta_\alpha(y_\alpha^k, y_\alpha) \]

This is tractable (poly time via LP) to compute, but not smooth.
Entropy Smoothing

\[ l_1(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} \theta_F^k \cdot \mu \]

\[ \theta_F^k(y_\alpha) = f_\alpha(x^k, y_\alpha) + \Delta_\alpha(y_\alpha^k, y_\alpha) \]

This is tractable (poly time via LP) to compute, but not smooth. Final approximate loss: add entropy smoothing (Meshi et al., 2012)

\[ l_2(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} \left( \theta_F^k \cdot \mu + \epsilon \sum_\alpha H(\mu_\alpha) \right) \]
Entropy Smoothing

\[ l_1(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} \theta^k_F \cdot \mu \]

\[ \theta^k_F(y_\alpha) = f_\alpha(x^k, y_\alpha) + \Delta_\alpha(y^k_\alpha, y_\alpha) \]

This is tractable (poly time via LP) to compute, but not smooth. Final approximate loss: add entropy smoothing (Meshi et al., 2012)

\[ l_2(x^k, y^k; F) = -F(x^k, y^k) + \max_{\mu \in \mathcal{M}} \left( \theta^k_F \cdot \mu + \epsilon \sum_\alpha H(\mu_\alpha) \right) \]

Bounded error:

\[ l_1(x, y, F) \leq l_2(x, y, F) \leq l_1(x, y, F) + \epsilon H_{\text{max}}, \quad H_{\text{max}} = \sum_\alpha |y_\alpha| \log |y_\alpha|. \]
Outline

1. Overview
2. Loss Functions
3. Joint Learning/Inference
4. Logistic Regression
5. Experiments
   - Binary Denoising
   - Horses
6. Conclusions
Saddle-Point Situation

\[
\min_F R(F) = \min_F \sum_k \left[ -F(x^k, y^k) + A(\theta^k_F) \right]
\]

\[
A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_\alpha H(\mu_\alpha).
\]
Saddle-Point Situation

\[ \min_{F} R(F) = \min_{F} \sum_{k} [-F(x^k, y^k) + A(\theta^k_F)] \]

\[ A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_\alpha). \]

Saddle-point!
Saddle-Point Situation

\[
\min_{F} R(F) = \min_{F} \sum_{k} \left[ -F(x^k, y^k) + A(\theta^k_F) \right]
\]

\[
A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_{\alpha}).
\]

Saddle-point! But what if dual \( A(\theta) = \min_{\lambda} A(\lambda, \theta) \) exists?
Saddle-Point Situation

\[
\min_F R(F) = \min_F \sum_k \left[ -F(x^k, y^k) + A(\theta^k_F) \right]
\]

\[
A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_\alpha).
\]

Saddle-point! But what if dual \( A(\theta) = \min_\lambda A(\lambda, \theta) \) exists?

Joint minimization

\[
\min_F \min_{\lambda^k} \sum_k \left[ -F(x^k, y^k) + A(\lambda^k, \theta^k_F) \right].
\]
Saddle-Point Situation

\[
\min_F R(F) = \min_F \sum_k \left[ -F(x^k, y^k) + A(\theta^k_F) \right]
\]

\[
A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_\alpha H(\mu_\alpha).
\]

Saddle-point! But what if dual \( A(\theta) = \min_\lambda A(\lambda, \theta) \) exists?

Joint minimization

\[
\min_F \min_{\{\lambda^k\}} \sum_k \left[ -F(x^k, y^k) + A(\lambda^k, \theta^k_F) \right].
\]

Existing works (Meshi et al., 2010; Hazan and Urtasun, 2012) alternate:

1. Message-passing updates to \( \{\lambda^k\} \).
2. Gradient updates to \( F \) (linear)
Saddle-Point Situation

\[ \min_F R(F) = \min_F \sum_k \left[ -F(x^k, y^k) + A(\theta^k_F) \right] \]

\[ A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_{\alpha}) . \]

Saddle-point! But what if dual \( A(\theta) = \min_\lambda A(\lambda, \theta) \) exists?

Joint minimization

\[ \min_F \min_{\{\lambda^k\}} \sum_k \left[ -F(x^k, y^k) + A(\lambda^k, \theta^k_F) \right] . \]

Existing works (Meshi et al., 2010; Hazan and Urtasun, 2012) alternate:

1. Message-passing updates to \( \{\lambda^k\} \).
2. Gradient updates to \( F \) (linear)

This talk:

1. Message passing updates to \( \{\lambda^k\} \)
2. Solve logistic regression problem(s) to get new \( F \)
Message - Passing Updates

\[ A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_\alpha). \]

How to get/solve dual \( A(\theta) = \min_{\lambda} A(\lambda, \theta) \)?
Message - Passing Updates

\[ A(\theta) = \max_{\mu \in M} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_\alpha). \]

How to get/solve dual \( A(\theta) = \min_\lambda A(\lambda, \theta) \)? Variant of Heskes (2006).
Message - Passing Updates

\[ A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_\alpha). \]

How to get/solve dual \( A(\theta) = \min_\lambda A(\lambda, \theta) \)? Variant of Heskes (2006).

- Create a Lagrangian \( \mathcal{L}(\lambda, \theta, \mu) \)
Message - Passing Updates

\[ A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_\alpha). \]

How to get/solve dual \( A(\theta) = \min_{\lambda} A(\lambda, \theta) \)? Variant of Heskes (2006).

- Create a Lagrangian \( \mathcal{L}(\lambda, \theta, \mu) \)
- Multiplier \( \lambda_\alpha(x_\beta) \) enforces that \( \mu_{\alpha\beta}(x_\beta) = \mu_\beta(x_\beta) \).
Message - Passing Updates

\[ A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_\alpha). \]

How to get/solve dual \( A(\theta) = \min_\lambda A(\lambda, \theta) \)? Variant of Heskes (2006).

- Create a Lagrangian \( \mathcal{L}(\lambda, \theta, \mu) \)
- Multiplier \( \lambda_\alpha(x_\beta) \) enforces that \( \mu_{\alpha\beta}(x_\beta) = \mu_\beta(x_\beta) \).
- Given \( \lambda \), maximizing \( \mu \) is

\[ \mu_\alpha(y_\alpha) = \frac{1}{Z_\alpha} \exp \left( \frac{1}{\epsilon} \left( \theta(y_\alpha) + \sum_{\beta \subset \alpha} \lambda_\alpha(y_\beta) - \sum_{\gamma \supset \alpha} \lambda_\gamma(y_\alpha) \right) \right). \]
Message - Passing Updates

\[ A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_{\alpha}). \]

How to get/solve dual \( A(\theta) = \min_\lambda A(\lambda, \theta) \) ? Variant of Heskes (2006).

- Create a Lagrangian \( \mathcal{L}(\lambda, \theta, \mu) \)
- Multiplier \( \lambda_\alpha(x_\beta) \) enforces that \( \mu_{\alpha\beta}(x_\beta) = \mu_\beta(x_\beta) \).
- Given \( \lambda \), maximizing \( \mu \) is

\[
\mu_\alpha(y_\alpha) = \frac{1}{Z_\alpha} \exp \left( \frac{1}{\epsilon} \left( \theta(y_\alpha) + \sum_{\beta \subset \alpha} \lambda_\alpha(y_\beta) - \sum_{\gamma \supset \alpha} \lambda_\gamma(y_\alpha) \right) \right).
\]

- Block coord. descent to minimize \( A(\lambda, \theta) = \max_\mu \mathcal{L}(\lambda, \theta, \mu) \).
Message - Passing Updates

\[ A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_\alpha). \]

How to get/solve dual \( A(\theta) = \min_\lambda A(\lambda, \theta) \)? Variant of Heskes (2006).

- Create a Lagrangian \( \mathcal{L}(\lambda, \theta, \mu) \)
- Multiplier \( \lambda_\alpha(x_\beta) \) enforces that \( \mu_{\alpha\beta}(x_\beta) = \mu_\beta(x_\beta) \).
- Given \( \lambda \), maximizing \( \mu \) is

\[
\mu_\alpha(y_\alpha) = \frac{1}{Z_\alpha} \exp \left( \frac{1}{\epsilon} \left( \theta(y_\alpha) + \sum_{\beta \subset \alpha} \lambda_\alpha(y_\beta) - \sum_{\gamma \supset \alpha} \lambda_\gamma(y_\alpha) \right) \right).
\]

- Block coord. descent to minimize \( A(\lambda, \theta) = \max_\mu \mathcal{L}(\lambda, \theta, \mu) \). For blocks \( \{ \lambda_\alpha(x_\nu) | \alpha \supset \nu \} \):
Message - Passing Updates

\[ A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_{\alpha}). \]

How to get/solve dual \( A(\theta) = \min_\lambda A(\lambda, \theta) \)? Variant of Heskes (2006).

- Create a Lagrangian \( \mathcal{L}(\lambda, \theta, \mu) \)
- Multiplier \( \lambda_{\alpha}(x_{\beta}) \) enforces that \( \mu_{\alpha\beta}(x_{\beta}) = \mu_{\beta}(x_{\beta}) \).
- Given \( \lambda \), maximizing \( \mu \) is

\[
\mu_{\alpha}(y_{\alpha}) = \frac{1}{Z_{\alpha}} \exp \left( \frac{1}{\epsilon} \left( \theta(y_{\alpha}) + \sum_{\beta \subset \alpha} \lambda_{\alpha}(y_{\beta}) - \sum_{\gamma \supset \alpha} \lambda_{\gamma}(y_{\alpha}) \right) \right).
\]

- Block coord. descent to minimize \( A(\lambda, \theta) = \max_\mu \mathcal{L}(\lambda, \theta, \mu) \). For blocks \( \{ \lambda_{\alpha}(x_{\nu}) | \alpha \supset \nu \} \):

\[
\lambda'_\alpha(y_{\nu}) \leftarrow \lambda_\alpha(y_{\nu}) + \frac{\epsilon}{1 + N_{\nu}} \left( \log \mu_{\nu}(y_{\nu}) + \sum_{\alpha' \supset \nu} \log \mu_{\alpha'}(y_{\nu}) \right) - \epsilon \log \mu_{\alpha}(y_{\nu}),
\]
The Talk So Far
The Talk So Far

- Want to do learning by minimizing

\[
\sum_k \left[ -F(x^k, y^k) + A(\theta^k_F) \right]
\]

\[
A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_\alpha).
\]
The Talk So Far

• Want to do learning by minimizing

$$\sum_k \left[ -F(x^k, y^k) + A(\theta^k_F) \right]$$

$$A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_\alpha H(\mu_\alpha).$$

• Can transform this problem into the joint optimization

$$\min_F \min_{\{\lambda^k\}} \sum_k \left[ -F(x^k, y^k) + A(\lambda^k, \theta^k_F) \right].$$
The Talk So Far

• Want to do learning by minimizing

\[
\sum_k \left[ -F(x^k, y^k) + A(\theta^k_F) \right]
\]

\[
A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_{\alpha} H(\mu_{\alpha}).
\]

• Can transform this problem into the joint optimization

\[
\min_{\mathcal{F}} \min_{\{\lambda^k\}} \sum_k \left[ -F(x^k, y^k) + A(\lambda^k, \theta^k_F) \right].
\]

• For fixed \( F \), can do \( \min_{\lambda^k} A(\lambda^k, \theta^k_F) \) through some interesting coordinate descent / message-passing algorithms.
The Talk So Far

- Want to do learning by minimizing

\[
\sum_k [ -F(x^k, y^k) + A(\theta^k_F) ] 
\]

\[
A(\theta) = \max_{\mu \in \mathcal{M}} \theta \cdot \mu + \epsilon \sum_\alpha H(\mu_\alpha). 
\]

- Can transform this problem into the joint optimization

\[
\min_F \min_{\lambda} \sum_k [ -F(x^k, y^k) + A(\lambda^k, \theta^k_F) ] . 
\]

- For fixed \( F \), can do \( \min_{\lambda} A(\lambda^k, \theta^k_F) \) through some interesting coordinate descent / message-passing algorithms.

**Obvious question**: For fixed \( \{\lambda^k\} \), how do we minimize with respect to \( F \)?
Outline

1. Overview
2. Loss Functions
3. Joint Learning/Inference
4. Logistic Regression
5. Experiments
   - Binary Denoising
   - Horses
6. Conclusions
Logistic Regression

Normal logistic regression: Given dataset \((x^1, y^1), \ldots, (x^N, y^N)\)

\[ \max_W \sum_k \left[ (Wx^k)_y - \log \sum_y \exp(Wx^k)_y \right]. \]

\[ x^k \in \mathbb{R}^M \]

\[ y^k \in \{1, 2, \ldots, L\} \]
Logistic Regression

Normal logistic regression: Given dataset \((x^1, y^1), ..., (x^N, y^N)\)

\[
\max_W \sum_k \left[ (Wx^k)_{y^k} - \log \sum_y \exp(Wx^k)_y \right].
\]

\(x^k \in \mathbb{R}^M\)

\(y^k \in \{1, 2, ..., L\}\)

Generalize to a set of functions \(F\). (Same for \(f(x, y) = (Wx)_y\).)

\[
\max_{f \in F} \sum_k \left[ f(x^k, y^k) - \log \sum_y \exp f(x^k, y) \right]
\]
Logistic Regression

Normal logistic regression: Given dataset \((x^1, y^1), ..., (x^N, y^N)\)

\[
\max_W \sum_k \left[ (Wx^k)_y - \log \sum_y \exp(Wx^k)_y \right].
\]

\(x^k \in \mathbb{R}^M\)

\(y^k \in \{1, 2, ..., L\}\)

Generalize to a set of functions \(\mathcal{F}\). (Same for \(f(x, y) = (Wx)_y\).)

\[
\max_{f \in \mathcal{F}} \sum_k \left[ f(x^k, y^k) - \log \sum_y \exp f(x^k, y) \right]
\]

Generalize again, adding a “bias” \(b^k\) for each \((x^k, y^k)\).

\[
\max_{f \in \mathcal{F}} \sum_k \left[ (f(x^k, y^k) + b^k(y^k)) - \log \sum_y \exp (f(x^k, y) + b^k(y)) \right]
\]
Logistic Regression

\[
\max_{f \in \mathcal{F}} \sum_k \left[ (f(x^k, y^k) + b^k(y^k)) - \log \sum_y \exp (f(x^k, y) + b^k(y)) \right]
\]

What sets of functions can we solve this for?
Logistic Regression

$$\max_{f \in \mathcal{F}} \sum_k \left[ (f(x^k, y^k) + b^k(y^k)) - \log \sum_y \exp (f(x^k, y) + b^k(y)) \right]$$

What sets of functions can we solve this for?

- Linear Functions, $f(x, y) = (Wx)_y$
Logistic Regression

\[
\max_{f \in \mathcal{F}} \sum_k \left[ (f(x^k, y^k) + b^k(y^k)) - \log \sum_y \exp (f(x^k, y) + b^k(y)) \right]
\]

What sets of functions can we solve this for?

- Linear Functions, \( f(x, y) = (Wx)_y \)
- Things reducible to linear (polynomials, splines, etc)
Logistic Regression

$$\max_{f \in F} \sum_k \left[ (f(x^k, y^k) + b^k(y^k)) - \log \sum_y \exp (f(x^k, y) + b^k(y)) \right]$$

What sets of functions can we solve this for?

- Linear Functions, $f(x, y) = (Wx)_y$
- Things reducible to linear (polynomials, splines, etc)
- Multi-Layer Perceptrons, $f(x, y) = (W \sigma(Ux))_y$
Logistic Regression

\[
\max_{f \in F} \sum_k \left[ (f(x^k, y^k) + b^k(y^k)) - \log \sum_y \exp (f(x^k, y) + b^k(y)) \right]
\]

What sets of functions can we solve this for?

- Linear Functions, \( f(x, y) = (Wx)_y \)
- Things reducible to linear (polynomials, splines, etc)
- Multi-Layer Perceptrons, \( f(x, y) = (W \sigma(Ux))_y \)
- Trees
Logistic Regression

\[
\max_{f \in \mathcal{F}} \sum_k \left[ \left( f(x^k, y^k) + b^k(y^k) \right) - \log \sum_y \exp \left( f(x^k, y) + b^k(y) \right) \right]
\]

What sets of functions can we solve this for?

- Linear Functions, \( f(x, y) = (Wx)_y \)
- Things reducible to linear (polynomials, splines, etc)
- Multi-Layer Perceptrons, \( f(x, y) = (W\sigma(Ux))_y \)
- Trees
- Ensembles of Trees
Logistic Regression

$$\max_{f \in \mathcal{F}} \sum_k \left[ (f(x^k, y^k) + b^k(y^k)) - \log \sum_y \exp (f(x^k, y) + b^k(y)) \right]$$

What sets of functions can we solve this for?

- Linear Functions, $f(x, y) = (Wx)_y$
- Things reducible to linear (polynomials, splines, etc)
- Multi-Layer Perceptrons, $f(x, y) = (W\sigma(Ux))_y$
- Trees
- Ensembles of Trees
- Others?
The Reduction

\[ f^*_\alpha = \arg \min_{f\in F_\alpha} \sum_k \left[ -F(x^k, y^k) + A(\lambda^k, \theta_F^k) \right], \quad F(x, y) = \sum_\alpha f_\alpha(x, y_\alpha). \]
The Reduction

\[ f^*_\alpha = \arg \min_{f_\alpha \in F_\alpha} \sum_k \left[ -F(x^k, y^k) + A(\lambda^k, \theta^k_F) \right], \quad F(x, y) = \sum_\alpha f_\alpha(x, y_\alpha). \]

Claim

\[ f^*_\alpha = \epsilon \arg \max_{f_\alpha \in F_\alpha} \sum_k \left[ (f_\alpha(x^k, y^k_\alpha) + b^k_\alpha(y^k_\alpha)) \right. \]
\[ \left. - \log \sum_{y_\alpha} \exp \left( f_\alpha(x^k, y_\alpha) + b^k_\alpha(y_\alpha) \right) \right] \]

\[ b^k_\alpha(y_\alpha) = \frac{1}{\epsilon} \left( \Delta(y^k_\alpha, y_\alpha) + \sum_{\beta \subset \alpha} \lambda_\beta(y_\beta) - \sum_{\gamma \supset \alpha} \lambda_\gamma(y_\alpha) \right) \]
Algorithm

Initialize $\lambda^k \leftarrow 0$. Repeat until convergence:

- Set biases to

$$b^k_{\alpha}(y_{\alpha}) \leftarrow \frac{1}{\varepsilon} \left( \Delta(y^k_{\alpha}, y_{\alpha}) + \sum_{\beta \subset \alpha} \lambda^k_{\alpha}(y_{\beta}) - \sum_{\gamma \supset \alpha} \lambda^k_{\gamma}(y_{\alpha}) \right).$$

- Solve the logistic regression problems

$$f_{\alpha} \leftarrow \arg \max_{f_{\alpha} \in F_{\alpha}} \sum_{k=1}^{K} \left[ \left( f_{\alpha}(x^k, y^k_{\alpha}) + b^k_{\alpha}(y^k_{\alpha}) \right) - \log \sum_{y_{\alpha}} \exp \left( f_{\alpha}(x^k, y_{\alpha}) + b^k_{\alpha}(y_{\alpha}) \right) \right]$$

- Form updated parameters

$$\theta^k(y_{\alpha}) \leftarrow \varepsilon f_{\alpha}(x^k, y_{\alpha}) + \Delta(y^k_{\alpha}, y_{\alpha}).$$

- Perform a fixed number of message-passing iterations to update $\lambda^k$ using $\theta^k$. 
Outline

1. Overview
2. Loss Functions
3. Joint Learning/Inference
4. Logistic Regression
5. Experiments
   - Binary Denoising
   - Horses
6. Conclusions
Function Sets

Five sets of functions $\mathcal{F}_\alpha$:

- Zero
- Constant (ignores $x$)
- Linear
- Single Hidden-Layer MLP
- Boosted Decision Trees
Outline

1. Overview
2. Loss Functions
3. Joint Learning/Inference
4. Logistic Regression
5. Experiments
   - Binary Denoising
   - Horses
6. Conclusions
Binary Denoising

\[ f_{\{i\}} \] - univariate features \( \phi_i \)
\[ f_{\{i,j\}} \] - pairwise features \( \phi_{ij} \)

\[ \begin{array}{c}
\text{x} \\
\end{array} \quad \begin{array}{c}
\text{y} \\
\end{array} \]

\( y \) - convolve gaussian noise with a gaussian and threshold.

- For \( \{\alpha\} = \{i\} \), if \( y_i = 0 \), \( \phi_i \in [0, .9] \), else \( \phi_i \in [.1, 1] \).
- For \( \{\alpha\} = \{i, j\} \), if \( y_i = y_j \), \( \phi_{ij} \in [0, .8] \), else \( \phi_{ij} \in [.2, 1] \).
<table>
<thead>
<tr>
<th>Zero</th>
<th>Const</th>
<th>Linear</th>
<th>Boosting</th>
<th>MLP</th>
<th>$\mathcal{F}_{ij}$</th>
<th>$\mathcal{F}_{i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Zero" /></td>
<td><img src="image" alt="Const" /></td>
<td><img src="image" alt="Linear" /></td>
<td><img src="image" alt="Boosting" /></td>
<td><img src="image" alt="MLP" /></td>
<td><img src="image" alt="Zero" /></td>
<td><img src="image" alt="Const" /></td>
</tr>
</tbody>
</table>

---

$x$

---

$y$
Zero Const Linear Boosting MLP $\mathcal{F}_{ij}$ $\setminus$ $\mathcal{F}_i$
Zero Const Linear Boosting MLP $\mathcal{F}_{ij} \setminus \mathcal{F}_i$
Learned $f_\alpha$

Linear

Boosting

MLP

Univariate

Pairs
### Denoising - Train

<table>
<thead>
<tr>
<th>$F_i \setminus F_{ij}$</th>
<th>Zero</th>
<th>Const.</th>
<th>Linear</th>
<th>Boost.</th>
<th>MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>.490</td>
<td>.490</td>
<td>.490</td>
<td>.465</td>
<td>.490</td>
</tr>
<tr>
<td>Const.</td>
<td>.490</td>
<td>.490</td>
<td>.490</td>
<td>.465</td>
<td>.490</td>
</tr>
<tr>
<td>Linear</td>
<td>.443</td>
<td>.077</td>
<td>.059</td>
<td>.056</td>
<td>.033</td>
</tr>
<tr>
<td>Boost.</td>
<td>.429</td>
<td>.032</td>
<td>.014</td>
<td>.012</td>
<td>.008</td>
</tr>
<tr>
<td>MLP</td>
<td>.435</td>
<td>.031</td>
<td>.014</td>
<td>.011</td>
<td>.008</td>
</tr>
</tbody>
</table>

### Denoising - Test

<table>
<thead>
<tr>
<th>$F_i \setminus F_{ij}$</th>
<th>Zero</th>
<th>Const.</th>
<th>Linear</th>
<th>Boost.</th>
<th>MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Const.</td>
<td>.502</td>
<td>.502</td>
<td>.502</td>
<td>.504</td>
<td>.502</td>
</tr>
<tr>
<td>Linear</td>
<td>.444</td>
<td>.077</td>
<td>.059</td>
<td>.057</td>
<td>.034</td>
</tr>
<tr>
<td>Boost.</td>
<td>.445</td>
<td>.033</td>
<td>.015</td>
<td>.013</td>
<td>.008</td>
</tr>
<tr>
<td>MLP</td>
<td>.445</td>
<td>.032</td>
<td>.015</td>
<td>.013</td>
<td>.008</td>
</tr>
</tbody>
</table>
Alternate Comparison

Take $f_i$ - MLP, $f_{ij}$ - Linear. Compare:

- Joint Training
- Fit $f_i$ with $\lambda = 0$, fit pairs with $f_i$ fixed.
- “Piecewise” - Fix $\lambda = 0$, train $f_i$ and $f_{ij}$ separately
## Alternate Comparison

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>Joint Error = .015</th>
<th>Univariate Fixed Error = .095</th>
<th>“Piecewise” Error = .438</th>
</tr>
</thead>
</table>
Alternate Comparison

Joint

Univariate Fixed

“Piecewise”

Univariate (MLP)

Pairs (Linear)
Outline

1. Overview
2. Loss Functions
3. Joint Learning/Inference
4. Logistic Regression
5. Experiments  
   Binary Denoising  
   Horses
6. Conclusions
Horses

\[ f_{\{i\}} - \text{univariate features } \phi_i \]
\[ f_{\{i,j\}} - \text{pairwise features } \phi_{ij} \]

\[ x \quad y \]

\[ y - \text{horse or not?} \]

- For \( \{\alpha\} = \{i\}, \) (1) RBG values (2), vertical and horizontal position (3) histogram of gradients.
- For \( \{\alpha\} = \{i, j\}, \) (1) \( l_2 \) distance of RGB for pixels \( i \) and \( j \) (2) Sobel edge filter
<table>
<thead>
<tr>
<th>Zero</th>
<th>Const</th>
<th>Linear</th>
<th>Boosting</th>
<th>MLP $\mathcal{F}_{ij} \setminus \mathcal{F}_i$</th>
</tr>
</thead>
</table>

$x$ and $y$ are inputs to the Zero Const Linear Boosting MLP model.
Zero Const Linear Boosting MLP \( F_{ij} \setminus F_i \)

\( x \)

\( y \)
### Horses - Train

<table>
<thead>
<tr>
<th>$\mathcal{F}<em>i \setminus \mathcal{F}</em>{ij}$</th>
<th>Zero</th>
<th>Const.</th>
<th>Linear</th>
<th>Boost.</th>
<th>MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>.211</td>
<td>.211</td>
<td>.212</td>
<td>.211</td>
<td>.210</td>
</tr>
<tr>
<td>Const.</td>
<td>.211</td>
<td>.211</td>
<td>.212</td>
<td>.211</td>
<td>.210</td>
</tr>
<tr>
<td>Linear</td>
<td>.141</td>
<td>.139</td>
<td>.126</td>
<td>.111</td>
<td>.113</td>
</tr>
<tr>
<td>Boost.</td>
<td>.087</td>
<td>.079</td>
<td>.074</td>
<td>.069</td>
<td>.068</td>
</tr>
<tr>
<td>MLP</td>
<td>.054</td>
<td>.051</td>
<td>.046</td>
<td>.043</td>
<td>.041</td>
</tr>
</tbody>
</table>

### Horses - Test

<table>
<thead>
<tr>
<th>$\mathcal{F}<em>i \setminus \mathcal{F}</em>{ij}$</th>
<th>Zero</th>
<th>Const.</th>
<th>Linear</th>
<th>Boost.</th>
<th>MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>.246</td>
<td>.246</td>
<td>.247</td>
<td>.246</td>
<td>.245</td>
</tr>
<tr>
<td>Const.</td>
<td>.246</td>
<td>.246</td>
<td>.247</td>
<td>.246</td>
<td>.245</td>
</tr>
<tr>
<td>Linear</td>
<td>.185</td>
<td>.185</td>
<td>.168</td>
<td>.158</td>
<td>.156</td>
</tr>
<tr>
<td>Boost.</td>
<td>.115</td>
<td>.107</td>
<td>.100</td>
<td>.095</td>
<td>.094</td>
</tr>
<tr>
<td>MLP</td>
<td>.096</td>
<td>.094</td>
<td>.087</td>
<td>.085</td>
<td>.081</td>
</tr>
</tbody>
</table>
Outline

1. Overview
2. Loss Functions
3. Joint Learning/Inference
4. Logistic Regression
5. Experiments
   - Binary Denoising
   - Horses
6. Conclusions
Conclusions

- Take home message: Can reduce structured training to a sequence of logistic regression problems.
Conclusions

• Take home message: Can reduce structured training to a sequence of logistic regression problems.

• Advantages:
  • Can use your favorite logistic regressor.
  • Modularity is good software engineering
Conclusions

• Take home message: Can reduce structured training to a sequence of logistic regression problems.

• Advantages:
  • Can use your favorite logistic regressor.
  • Modularity is good software engineering

• Disadvantage:
  • Stochastic training is (presumably) faster for linear energies and large datasets.
Conclusions

• Take home message: Can reduce structured training to a sequence of logistic regression problems.

• Advantages:
  • Can use your favorite logistic regressor.
  • Modularity is good software engineering

• Disadvantage:
  • Stochastic training is (presumably) faster for linear energies and large datasets.

• Mitigating factor:
  • Can use stochastic solvers for logistic regression problems.
Thank you


