Reward-modulated inference

Buck Shlegeris    Matthew Alger

COMP3740, 2014
Outline

- Supervised, unsupervised, and reinforcement learning
- Neural nets
- RMI
- Results with RMI
Types of machine learning

- supervised
- unsupervised
- reinforcement
MNIST

Buck Shlegeris, Matthew Alger

Reward-modulated inference
Supervised learning

Given some pairs \((x, f(x))\), approximate \(f\).
Given some pairs \((x, f(x))\), approximate \(f\).

- classification
Supervised learning

Given some pairs $(x, f(x))$, approximate $f$.

- classification
- regression
Supervised learning

Given some pairs \((x, f(x))\), approximate \(f\).

- classification
- regression
- prediction
Unsupervised learning

Given data, find patterns.
Unsupervised learning

Given data, find patterns.

- Goal is to increase likelihood of observed data.
Unsupervised learning

Given data, find patterns.

- Goal is to increase likelihood of observed data.
- Related to compression
Unsupervised learning

Given data, find patterns.

- Goal is to increase likelihood of observed data.
- Related to compression
- This is useful as a preprocessing step.
You see some stuff, and get some reward. What do you want to do?
You see some stuff, and get some reward. What do you want to do?

- Way more general $\rightarrow$ much harder $\rightarrow$ make assumptions.
Reinforcement learning

You see some stuff, and get some reward. What do you want to do?

- Way more general $\rightarrow$ much harder $\rightarrow$ make assumptions.
- Stationary

Stationary MDP

Value estimation? (Use supervised prediction?)

Explore vs exploit?

Preprocessing somehow?
You see some stuff, and get some reward. What do you want to do?

- Way more general → much harder → make assumptions.
  - Stationary
  - MDP
Reinforcement learning

You see some stuff, and get some reward. What do you want to do?

- Way more general → much harder → make assumptions.
  - Stationary
  - MDP
- Value estimation? (Use supervised prediction?)
You see some stuff, and get some reward. What do you want to do?

- **Way more general → much harder → make assumptions.**
  - Stationary
  - MDP

- **Value estimation? (Use supervised prediction?)**

- **Explore vs exploit?**
You see some stuff, and get some reward. What do you want to do?

- Way more general → much harder → make assumptions.
  - Stationary
  - MDP
- Value estimation? (Use supervised prediction?)
- Explore vs exploit?
- Preprocessing somehow?
What’s a nice class of functions from $\mathbb{R}^n$ to $\mathbb{R}^m$?
Neural nets

\[ f(\vec{x}) = \vec{Wx} + \vec{b} \] (1)
Neural nets

\[ f(\vec{x}) = W\vec{x} + \vec{b} \quad (1) \]

Let’s use the name \( \theta \) for our model, the combination of \( W \) and \( \vec{b} \).
Logistic regression

\[ P(Y = i | \theta, \mathbf{x}) = s(W\mathbf{x} + \mathbf{b})_i \]  

(2)

(s rescales vectors in \( \mathbb{R}^n \) to have an L1 norm of 1.)
Logistic regression

\[ P(Y = i|\theta, \vec{x}) = s(W\vec{x} + \vec{b})_i \]  \hspace{1cm} (2)

(s rescales vectors in \(\mathbb{R}^n\) to have an L1 norm of 1.)

\[
\text{prediction}(\theta, \vec{x}) = \arg\max_i (P(Y = i|\theta, \vec{x}))
\]  \hspace{1cm} (3)
Logistic regression

\[ \mathcal{L}(\theta, \vec{x}, y) = -\log(s(W\vec{x} + \vec{b})_y) \] (4)
Logistic regression

\[ \mathcal{L}(\theta, \vec{x}, y) = -\log(s(W\vec{x} + \vec{b})_y) \]  

Overfitting?
Logistic regression

\[ \mathcal{L}(\theta, \tilde{x}, y) = -\log(s(W\tilde{x} + \tilde{b})_y) \] (4)

Overfitting?

\[ \mathcal{L}(\theta, \tilde{x}, y) = -\log(s(W\tilde{x} + \tilde{b})_y) - R(\theta) \] (5)
Logistic regression

\[ \mathcal{L}(\theta, \vec{x}, y) = -\log(s(W\vec{x} + \vec{b})_y) \] (4)

Overfitting?

\[ \mathcal{L}(\theta, \vec{x}, y) = -\log(s(W\vec{x} + \vec{b})_y) - R(\theta) \] (5)

What if instead of a single input vector \( \vec{x} \) and single label \( y \), we had a whole list of inputs \( \mathcal{D} \) and a vector of labels \( \vec{y} \)?
Logistic regression

\[ \mathcal{L}(\theta, \vec{x}, y) = -\log(s(W\vec{x} + \vec{b})_y) \]  \hspace{1cm} (4)

Overfitting?

\[ \mathcal{L}(\theta, \vec{x}, y) = -\log(s(W\vec{x} + \vec{b})_y) - R(\theta) \]  \hspace{1cm} (5)

What if instead of a single input vector \( \vec{x} \) and single label \( y \), we had a whole list of inputs \( \mathcal{D} \) and a vector of labels \( \vec{y} \)?

\[ \mathcal{L}(\theta, \mathcal{D}, \vec{y}) = \sum_{i \in |\mathcal{D}|} \mathcal{L}(\theta, \mathcal{D}_i, \vec{y}_i) - R(\theta) \]  \hspace{1cm} (6)
$\theta^* (D, \bar{y}) =$
Logistic regression

\[ \theta^*(\mathcal{D}, \vec{y}) = \arg\min_{\theta} (\mathcal{L}(\theta, \mathcal{D}, \vec{y})) \quad (7) \]
\[
\theta^*(D, \vec{y}) = \arg\min_{\theta} (L(\theta, D, \vec{y}))
\]  

(Brock Shlegeris, Matthew Alger)

So good luck finding that analytically...
Logistic regression

Buck Shlegeris, Matthew Alger

Reward-modulated inference
Logistic regression

while last update was bigger than $\epsilon$ do
  $W_{\text{new}} \leftarrow W - \alpha \frac{\partial L(\theta, D, \vec{y})}{\partial W}$
end while

Gradient descent

Buck Shlegeris, Matthew Alger

Reward-modulated inference
Logistic regression

for input $\vec{x}$ and label $y \in (\mathcal{D}, \vec{y})$ do
\[ w \leftarrow W - \alpha \frac{\partial L(\theta, \vec{x}, y)}{\partial W} \]
end for

Stochastic gradient descent

Buck Shlegeris, Matthew Alger
Reward-modulated inference
Logistic regression

Only linearly separable things can be separated!
Multilayer perceptron

Logistic regression

\[ s(W\tilde{x} + \tilde{b}) \]

Multilayer perceptron

\[ s(Ws(W_2\tilde{x} + \tilde{b}_2) + \tilde{b}) = s(W\tilde{h} + \tilde{b}) \]
Why stop at 1 layer?
Why stop at 1 layer?

Figure: Multilayer perceptron with an input layer, hidden layer, hidden layer 2, and output layer.
Autoencoders

\[ \vec{x}' = s(W \cdot s(W^2 \vec{x} + \vec{b}^2) + \vec{b}) \] (8)

\[ L(\theta, \vec{x}) = ||\vec{x} - \vec{x}'|| \] (9)
\[ \bar{x}' = s(W \cdot s(W_2 \bar{x} + \bar{b}_2) + \bar{b}) \]
Autoencoders

\[ \tilde{x}' = s(W \cdot s(W_2 \tilde{x} + \tilde{b}_2) + \tilde{b}) \]  \hspace{1cm} (8)

\[ \mathcal{L}(\theta, \tilde{x}) = ||\tilde{x} - \tilde{x}'|| \]  \hspace{1cm} (9)
Denoising autoencoders (dAs)

\[
\text{input layer} \quad \text{hidden layer} \quad \text{reconstructed input}
\]

\[
\text{noise}
\]

\[
(W \cdot s(W_2x) + \vec{b}_2) + \vec{b}_1
\]

(where \( n \) is a stochastic noise function)

Buck Shlegeris, Matthew Alger

Reward-modulated inference
Denoising autoencoders (dAs)

\[ s(W \cdot s(W_2 n(\tilde{x}) + \tilde{b}_2) + \tilde{b}) \]  

(10)

(\text{where } n \text{ is a stochastic noise function})
Denoising autoencoders (dAs)

\[ W \cdot s( W x + b) + b \] (11)

Buck Shlegeris, Matthew Alger

Reward-modulated inference
Denoising autoencoders (dAs)

\[ s(W \cdot s(W_2 \bar{x} + \bar{b}_2) + \bar{b}) \]  

(11)
Denoising autoencoders (dAs)
Denoising autoencoders (dAs)

Buck Shlegeris, Matthew Alger

Reward-modulated inference
Denoising autoencoders (dAs)

Reward-modulated inference
Denoising autoencoders (dAs)

Reward-modulated inference
Denoising autoencoders (dAs)

\[ \mathcal{L}(\theta, \tilde{x}, y) = - \log(s(W\tilde{x} + \tilde{b})_y) \] (12)

How do we mix between these?
Denoising autoencoders (dAs)

\[ \mathcal{L}(\theta, \bar{x}, y) = - \log(s(W\bar{x} + \vec{b})_y) \]  

(12)

\[ \mathcal{L}(\theta, \bar{x}) = ||\bar{x} - \bar{x}'|| \]  

(13)
Denoising autoencoders (dAs)

\[ \mathcal{L}(\theta, \tilde{x}, y) = -\log(s(W\tilde{x} + \tilde{b})_y) \]  \hspace{1cm} (12)

\[ \mathcal{L}(\theta, \tilde{x}) = ||\tilde{x} - \tilde{x}'|| \]  \hspace{1cm} (13)

How do we mix between these?
Add a time-varying modulation function $\lambda(t)$:
Add a time-varying modulation function $\lambda(t)$:

$$
\mathcal{L}(\theta, \vec{x}, y) = \lambda(t) \cdot \text{supervised cost} + (1 - \lambda(t)) \cdot \text{unsupervised cost}
$$

(14)
Add a time-varying modulation function $\lambda(t)$:

$$\mathcal{L}(\theta, \vec{x}, y) = \lambda(t) \cdot \text{supervised cost} + (1 - \lambda(t)) \cdot \text{unsupervised cost}$$  \hspace{1cm} (14)$$

$$\mathcal{L}(\theta, \vec{x}, y) = \lambda(t) \left( -\log(s(W\vec{x} + \vec{b})_y) \right) + (1 - \lambda(t)) \left( ||\vec{x} - \vec{x}'|| \right)$$  \hspace{1cm} (15)$$
Reward modulation

Motivations:

- Information theory
Reward modulation

Motivations:
- Information theory
- Fine tuning
Motivations:

- Information theory
- Fine tuning
- Extreme learning
Questions:
Questions:

- Does it improve performance on problems?
Reward modulation

Questions:

- Does it improve performance on problems?
- How should we vary reward modulation?
Questions:

- Does it improve performance on problems?
- How should we vary reward modulation?
Results

Buck Shlegeris, Matthew Alger

Reward-modulated inference
Hyperparameters

Reward-modulated inference
Hyperparameters

Buck Shlegeris, Matthew Alger

Reward-modulated inference
Figure: Step reward modulation with $\lambda = 0$ if $t < p$, and $\lambda = 0$ otherwise.
Figure: Linear reward modulation with different changes in $\lambda$. 
Figure: Hyperbolic reward modulation with different changes in $\lambda$. 
no results yet.
Conclusions

RMI works on classification (maybe because it's like fine-tuning)

Haven't got contextual bandit results yet.

Works nicely on grid world for some reason, more MDP data to come.
Conclusions

- RMI works on classification (maybe because it’s like fine-tuning)

Buck Shlegeris, Matthew Alger  
Reward-modulated inference
Conclusions

- RMI works on classification (maybe because it’s like fine-tuning)
- Haven’t got contextual bandit results yet.
RMI works on classification (maybe because it’s like fine-tuning)
Haven’t got contextual bandit results yet.
Works nicely on grid world for some reason, more MDP data to come.