Reward-modulated inference

Buck Shlegeris   Matthew Alger

COMP3740, 2014

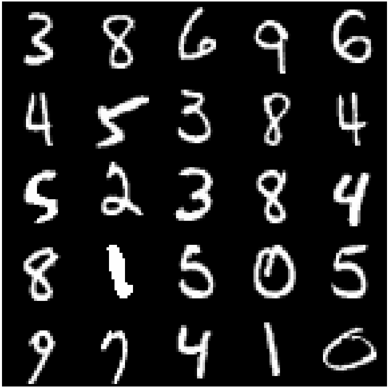
Outline

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

Types of machine learning

- supervised
- unsupervised
- reinforcement

MNIST



Supervised learning

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

- classification
- regression
- prediction

Unsupervised learning

Given data, find patterns.

- Goal is to increase likelihood of observed data.
- Related to compression
- This is useful as a preprocessing step.

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
  - MDP
- Value estimation? (Use supervised prediction?)
- Explore vs exploit?
- Preprocessing somehow?

Neural nets

What's a nice class of functions from $\mathbb{R}^n$ to $\mathbb{R}^m$?

Neural nets

$$f(\vec{\boldsymbol{x}}) = W\vec{\boldsymbol{x}} + \vec{\boldsymbol{b}} \tag{1}$$

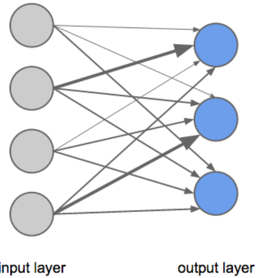
Let's use the name θ for our model, the combination of W and $\vec{\boldsymbol{b}}$.

Logistic regression

$$P(Y = i|\theta, \vec{\boldsymbol{x}}) = s(W\vec{\boldsymbol{x}} + \vec{\boldsymbol{b}})_i \tag{2}$$

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

$$\text{prediction}(\theta, \vec{\boldsymbol{x}}) = \underset{i}{\text{argmax}}(\mathcal{P}(Y = i|\theta, \vec{\boldsymbol{x}})) \tag{3}$$



Logistic regression

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

Overfitting?

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

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

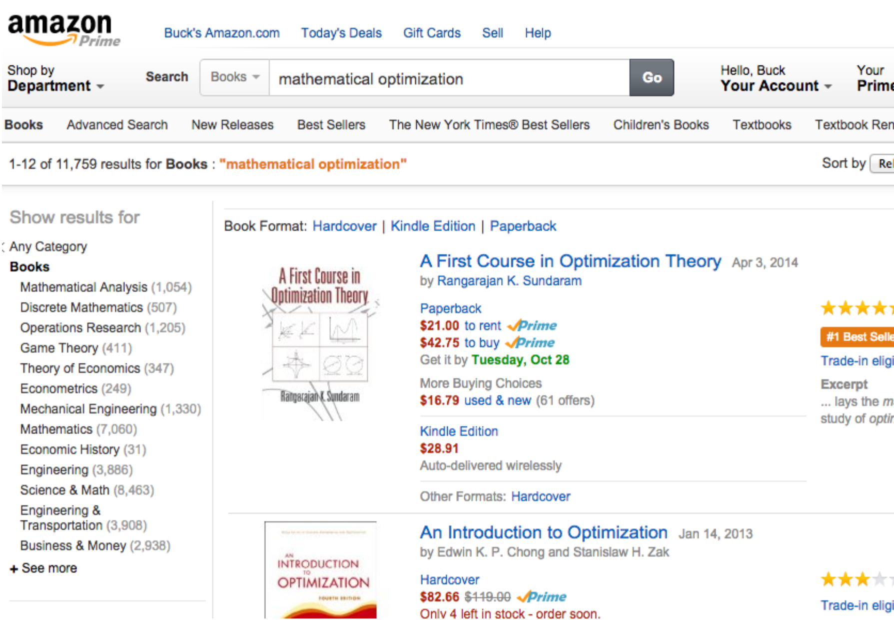
$$\mathcal{L}(\theta, \mathcal{D}, \vec{\boldsymbol{y}}) = \sum_{i \in |D|} \mathcal{L}(\theta, \mathcal{D}_i, \vec{\boldsymbol{y}}_i) - R(\theta) \tag{6}$$

Logistic regression

$$\theta^*(\mathcal{D}, \vec{y}) = \underset{\theta}{\operatorname{argmin}} \left(\mathcal{L}(\theta, \mathcal{D}, \vec{y})\right) \tag{7}$$

$\theta \in (\mathbb{R}^{|x| \cdot |y|} \times \mathbb{R}^{|y|})$, so good luck finding that analytically

Logistic regression



Logistic regression

Gradient descent

while last update was bigger than ϵ **do**

$\quad W_{\text{new}} \leftarrow W - \alpha \frac{\partial \mathcal{L}(\theta, \mathcal{D}, \vec{y})}{\partial W}$

end while

Logistic regression

Stochastic gradient descent

for input $\vec{x}$ and label $y \in (\mathcal{D}, \vec{y})$ **do**

$w \leftarrow W - \alpha \frac{\partial \mathcal{L}(\theta, \vec{x}, y)}{\partial W}$

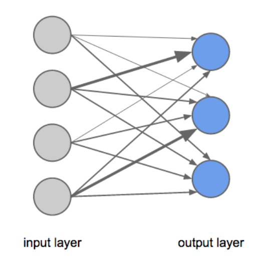
end for

Logistic regression

Only linearly separable things can be separated!

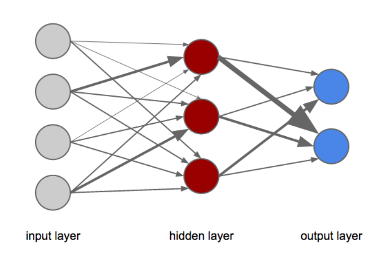
Multilayer perceptron

Logistic regression



$$s(W\vec{x} + \vec{b})$$

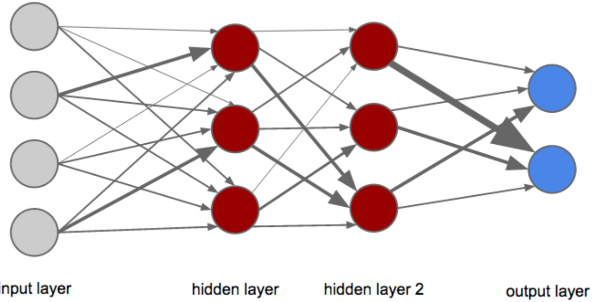
Multilayer perceptron



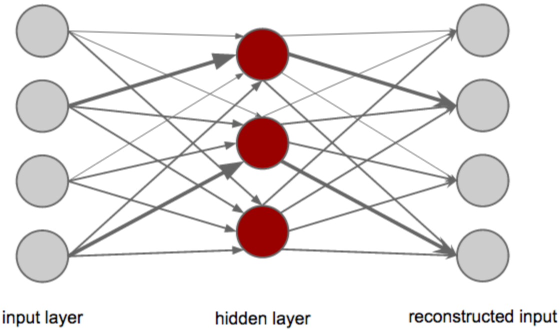
$$s(Ws(W_2\vec{x} + \vec{b}_2) + \vec{b})$$
$$= s(W\vec{h} + \vec{b})$$

Multilayer perceptron

Why stop at 1 layer?



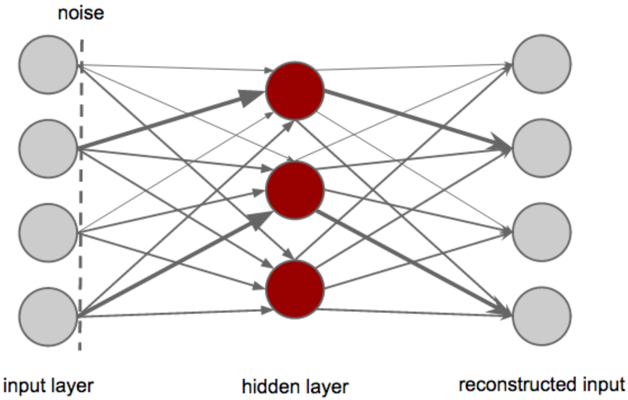
Autoencoders



$$\vec{x}' = s(W \cdot s(W_2\vec{x} + \vec{b}_2) + \vec{b}) \tag{8}$$

$$\mathcal{L}(\theta, \vec{x}) = ||\vec{x} - \vec{x}'|| \tag{9}$$

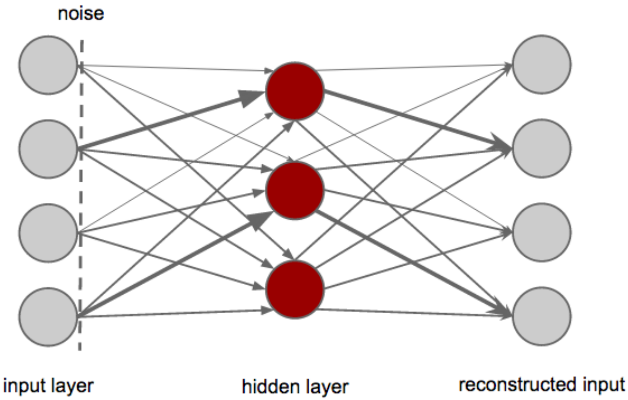
Denoising autoencoders (dAs)



$$s(W \cdot s(W_2 n(\vec{\boldsymbol{x}}) + \vec{\boldsymbol{b}}_2) + \vec{\boldsymbol{b}}) \tag{10}$$

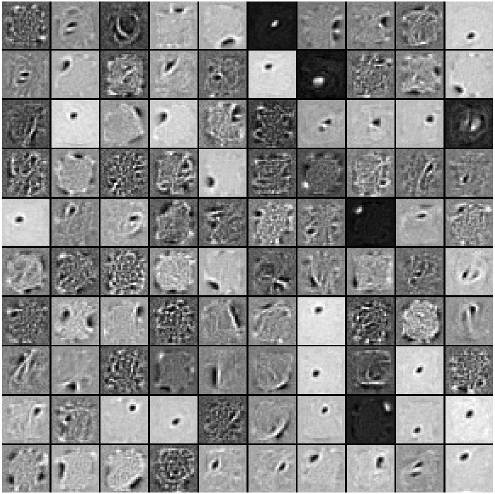
(where n is a stochastic noise function)

Denoising autoencoders (dAs)

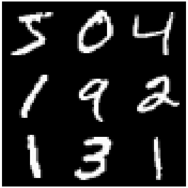


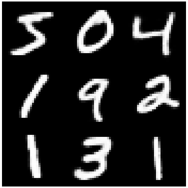
$$s(W \cdot s(W_2\vec{\boldsymbol{x}} + \vec{\boldsymbol{b}}_2) + \vec{\boldsymbol{b}}) \tag{11}$$

Denoising autoencoders (dAs)

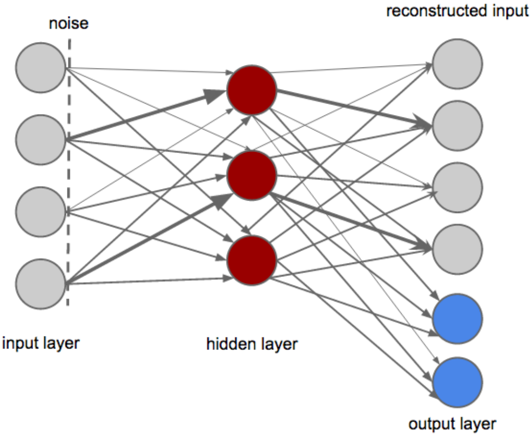


Denoising autoencoders (dAs)

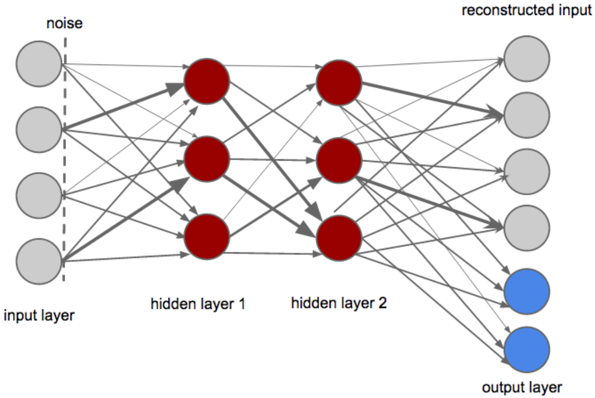




Denoising autoencoders (dAs)



Denoising autoencoders (dAs)



Denoising autoencoders (dAs)

$$\mathcal{L}(\theta, \vec{x}, y) = -\log(s(W\vec{x} + \vec{b})_y) \tag{12}$$

$$\mathcal{L}(\theta, \vec{x}) = ||\vec{x} - \vec{x'}|| \tag{13}$$

How do we mix between these?

Reward modulation

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

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

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

Reward modulation

Motivations:

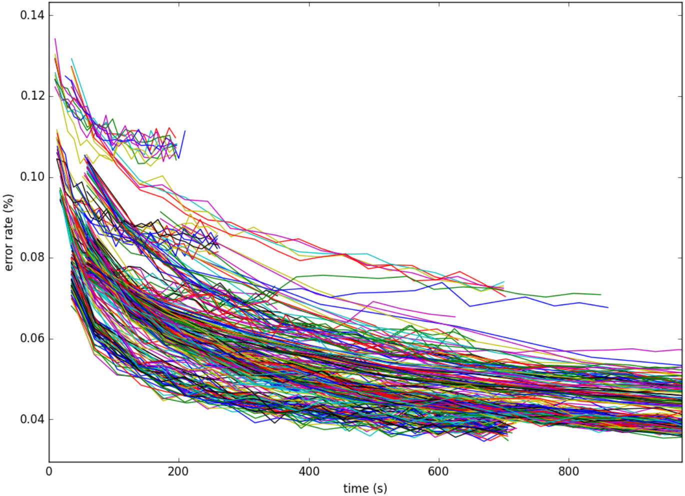
- Information theory
- Fine tuning
- Extreme learning

Reward modulation

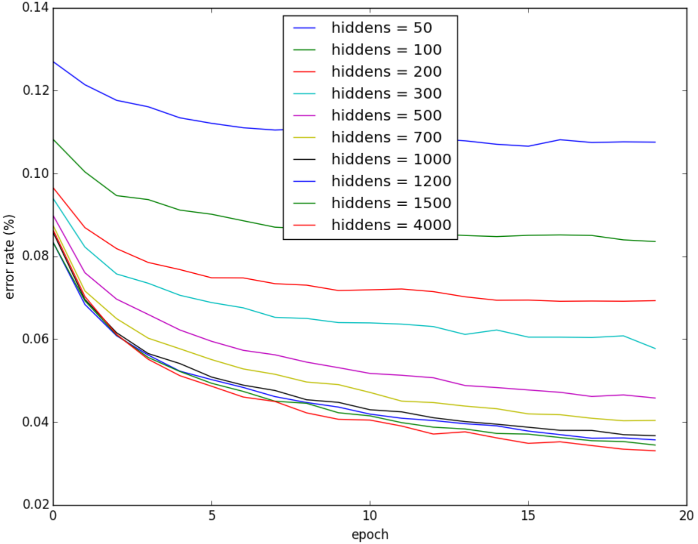
Questions:

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

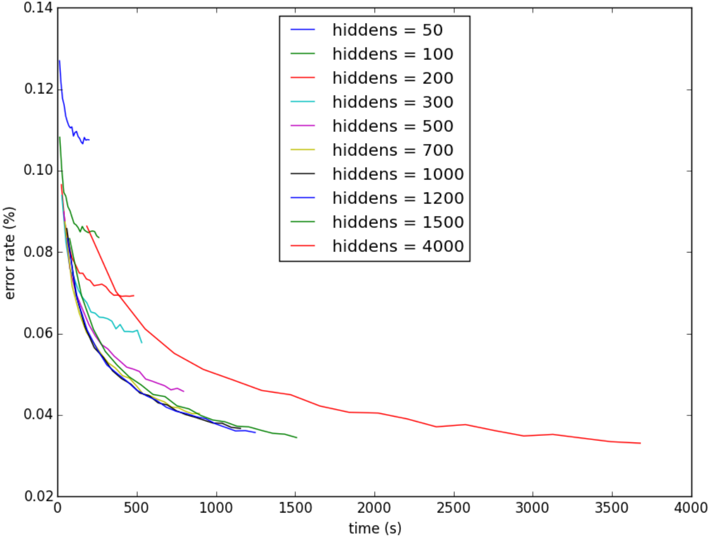
Results



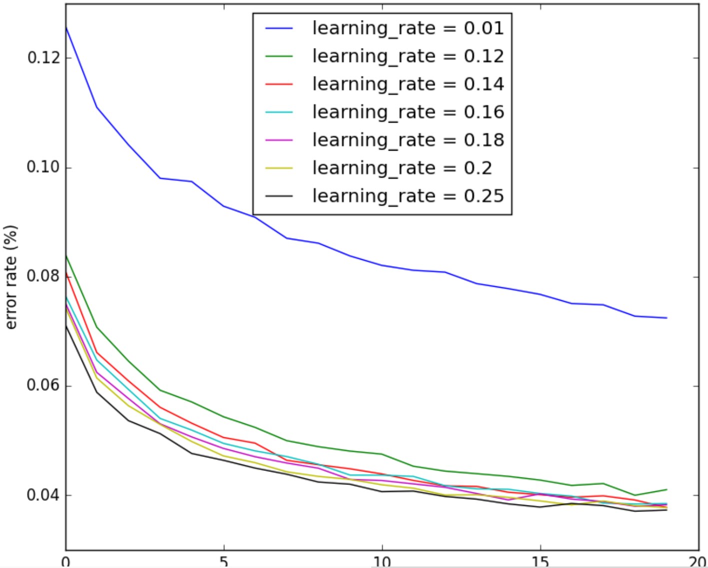
Hyperparameters



Hyperparameters



Hyperparameters



Classification problem

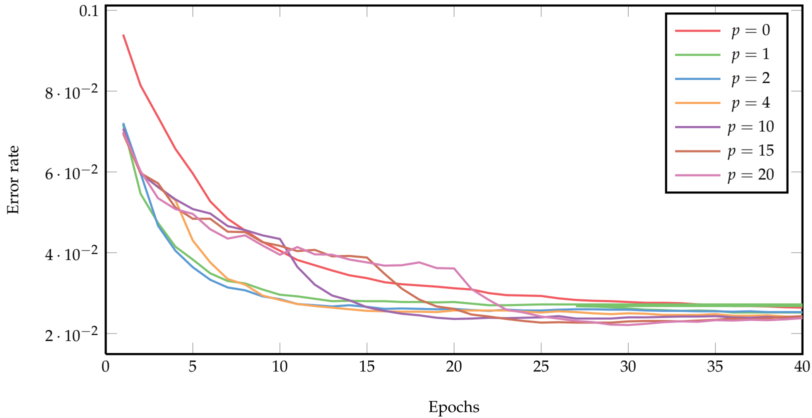


Figure: Step reward modulation with $\lambda = 0$ if $t < p$, and $\lambda = 0$ otherwise.

Classification problem

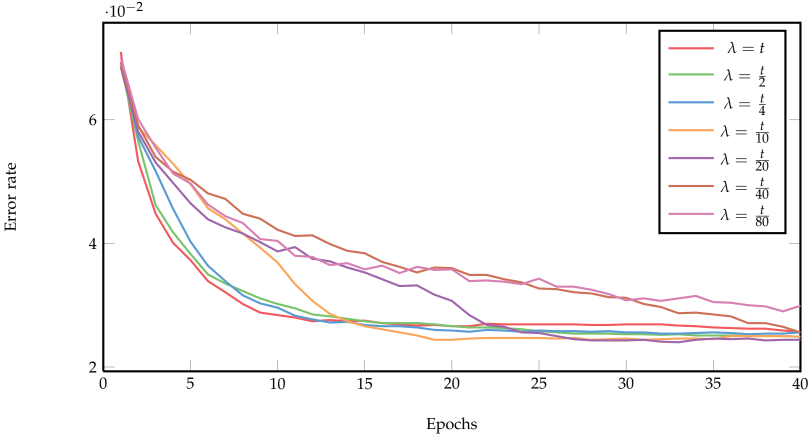


Figure: Linear reward modulation with different changes in λ.

Classification problem

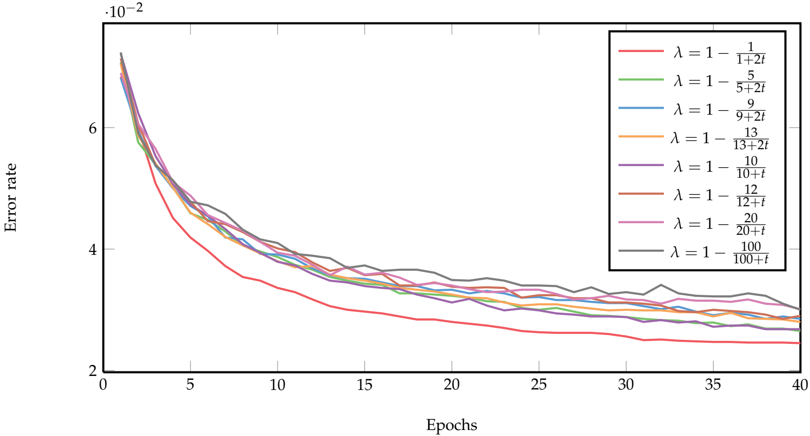
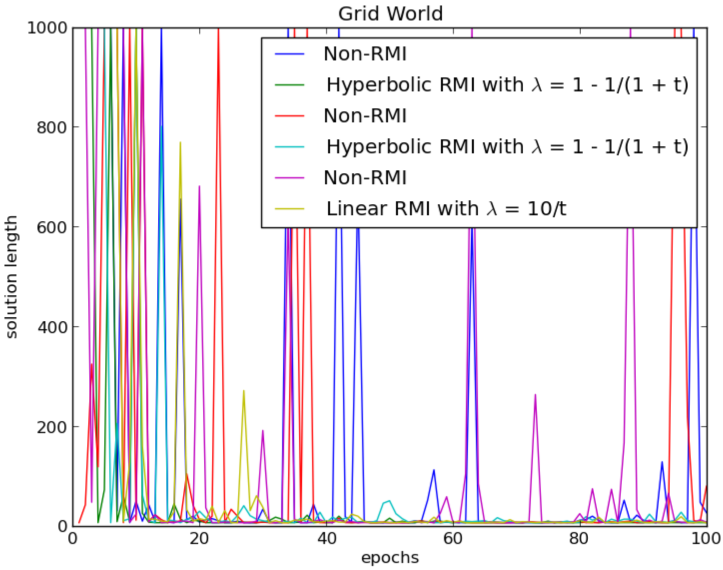


Figure: Hyperbolic reward modulation with different changes in λ.

Contextual bandit

no results yet.

MDP



Grid World
1000
800
600
400
200
0
0
20
40
60
80
100
solution length
epochs
Non-RMI
Hyperbolic RMI with λ = 1 - 1/(1 + t)
Non-RMI
Hyperbolic RMI with λ = 1 - 1/(1 + t)
Non-RMI
Linear RMI with λ = 10/t

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.