Exploiting Temporal Difference for Energy Disaggregation via Discriminative Sparse Coding

Eric Leijonmarck

Inspired by paper by Kotler et al.

Examiner: Timo Koski

Table of Contents

Disaggregation; state of 2015

Pre-Processing Data

Preliminaries

Results and Extensions

Future Work

Table of Contents

Disaggregation; state of 2015

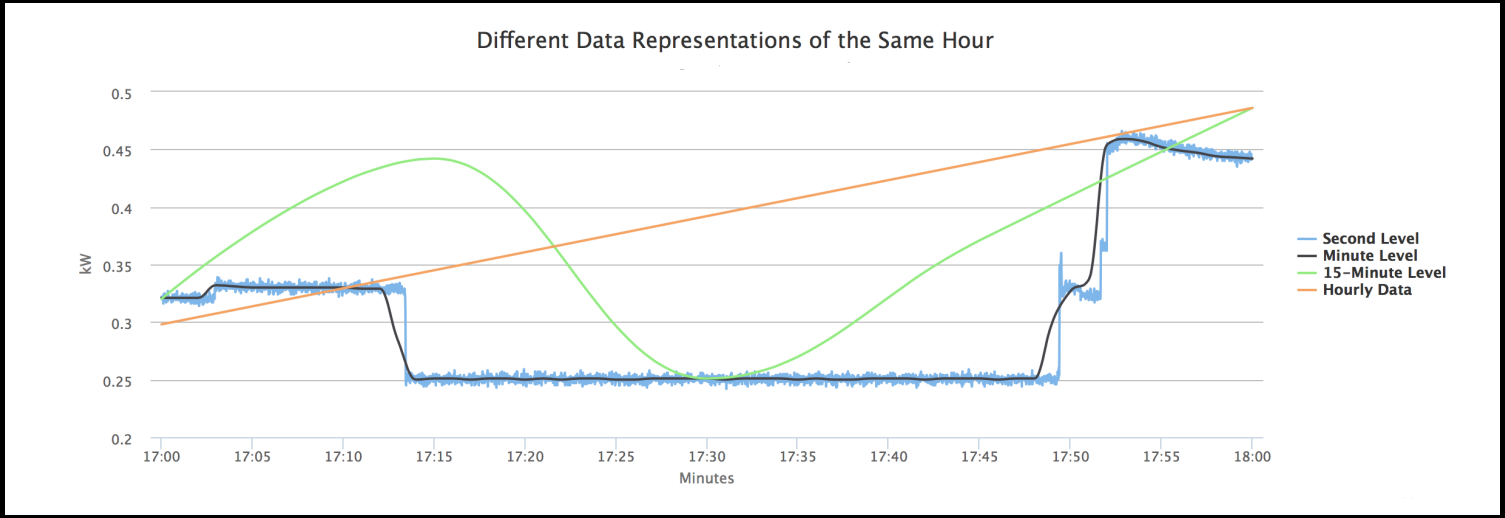
State of 2015

Why

- Rising Energy Consumption
- Energy Problems often Informatics Problem
- Energy Disaggregation
- Receiving Information, less energy

State of 2015

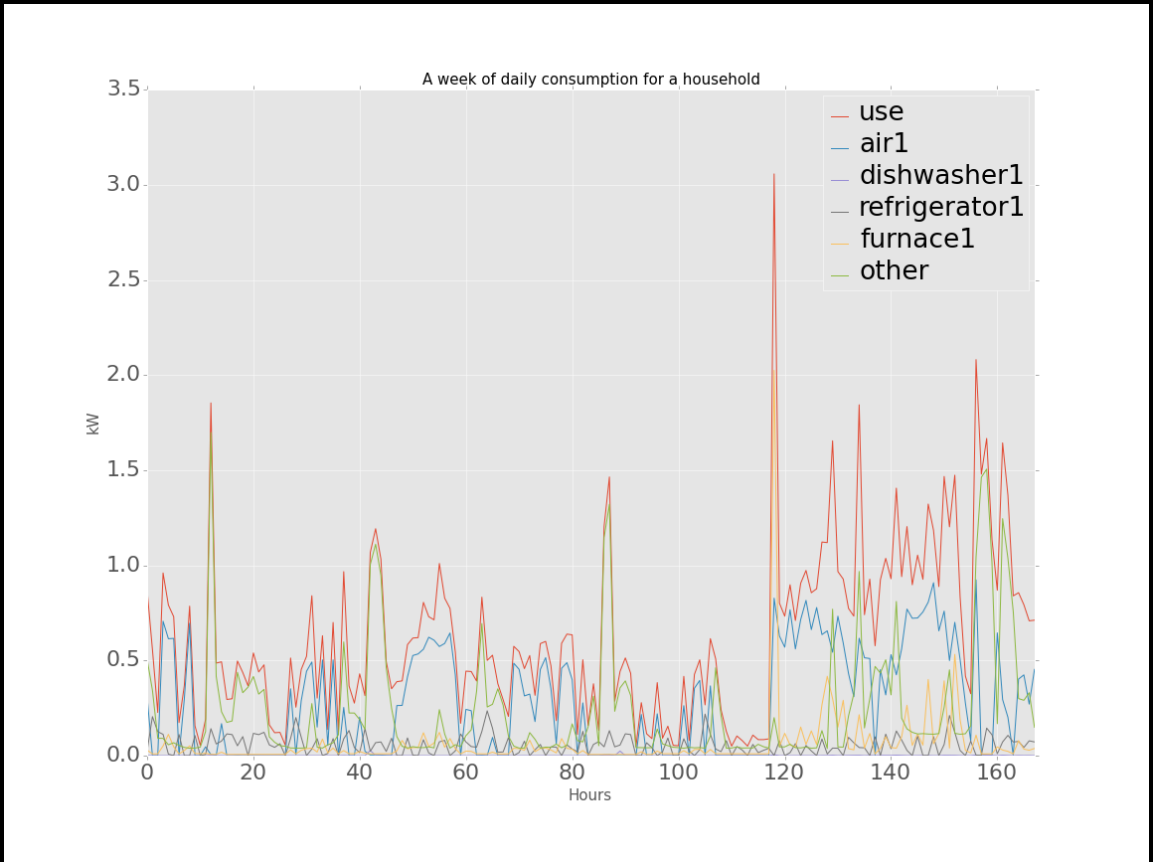
Challenge



State of 2015

Challenge

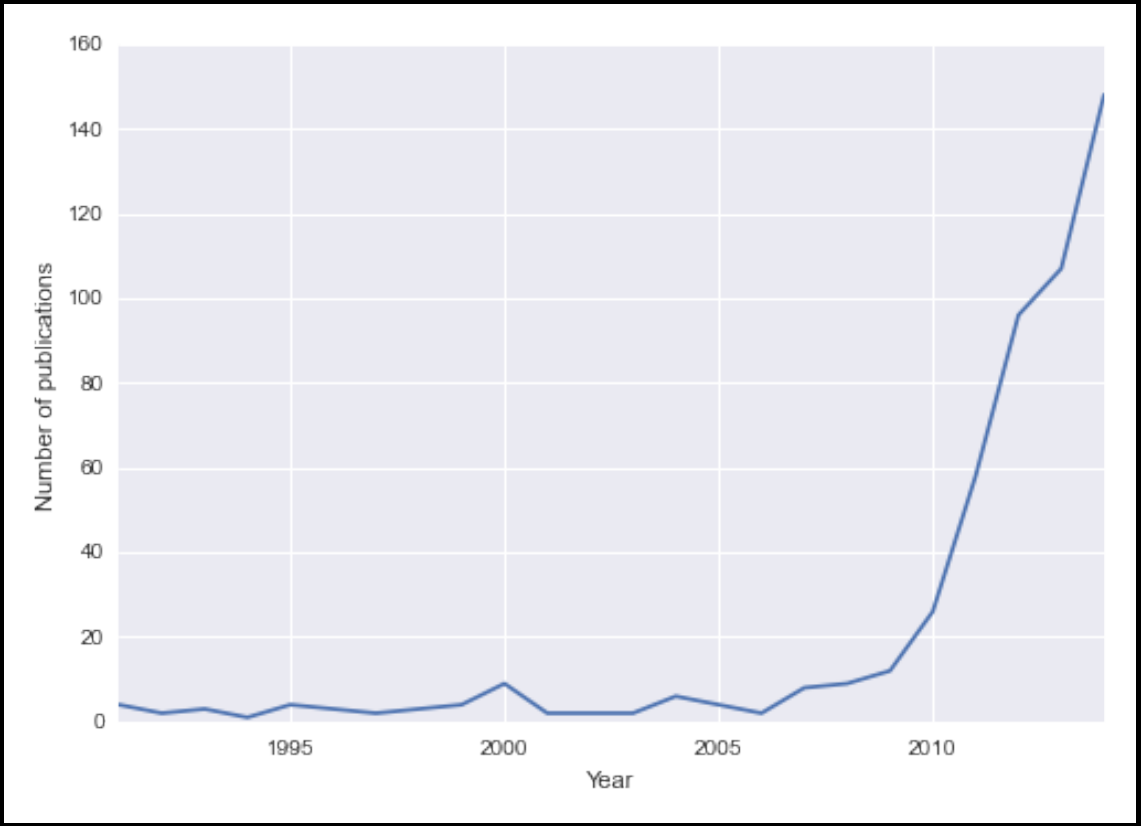
- Challenge



State of 2015

Research

- Research in Energy Disaggregation



Increase at 2010 from 20 - 140 in 4 years

State of 2015

Solution

- Ed Freeman, More Data or Better Algorithms?
  1. Having a good problem to work on
  2. Having a good approach to that problem
  3. Having decent data. Quality is much more important than quantity
  4. Handling the data well – good transforms, good missing value handling, making sure that all approaches make sense for the problem.

> Not everything that can be counted counts, and not everything that counts can be counted.
>
> Albert Einstein

Project Goal

- Implement the Discriminative Disaggregation algorithm using Sparse Coding
- Having it work on the dataset provided by PecanStreet
- Try to exploit temporal differences, since Kotler did not take that into account

Table of Contents

Pre-Processing Data

Input

We have $1 : k$ appliances

- We define one class (e.g. heater) $\mathbf{X}_i \leftarrow 1, \ldots, k$
- Where $\mathbf{X}_i \in \mathbb{R}^{T \times m}$, T hourly week data for m houses
- **One** aggregated household $\bar{\mathbf{X}} \leftarrow \sum_{i:k} \mathbf{X}_i$
- Assuming we have individual energy readings $\mathbf{X}_1, \ldots, \mathbf{X}_k$
- Goal: test with new data $\bar{\mathbf{X}}'$ to components $\mathbf{X}'_1, \ldots, \mathbf{X}'_k$

Input data

We have $1 : k$ appliances

$$\underbrace{\mathbf{X}_i \in \mathbb{R}^{T \times m}}$$

$$\begin{bmatrix} \text{App} & \mathbf{x}_1^{(j)} & \cdots & \mathbf{x}_1^{(m)} \\ 1h & 0.8kWh & \cdot & \cdot \\ 2h & 0.7kWh & \cdot & \cdot \\ \vdots & \vdots & \cdot & \cdot \\ 168h & 0.1kWh & \cdot & \cdot \end{bmatrix}$$

How do we get this format?

Pre-processing raw-data

- Number of missing values in data
- When missing - what to do?
- Demonstation of processing

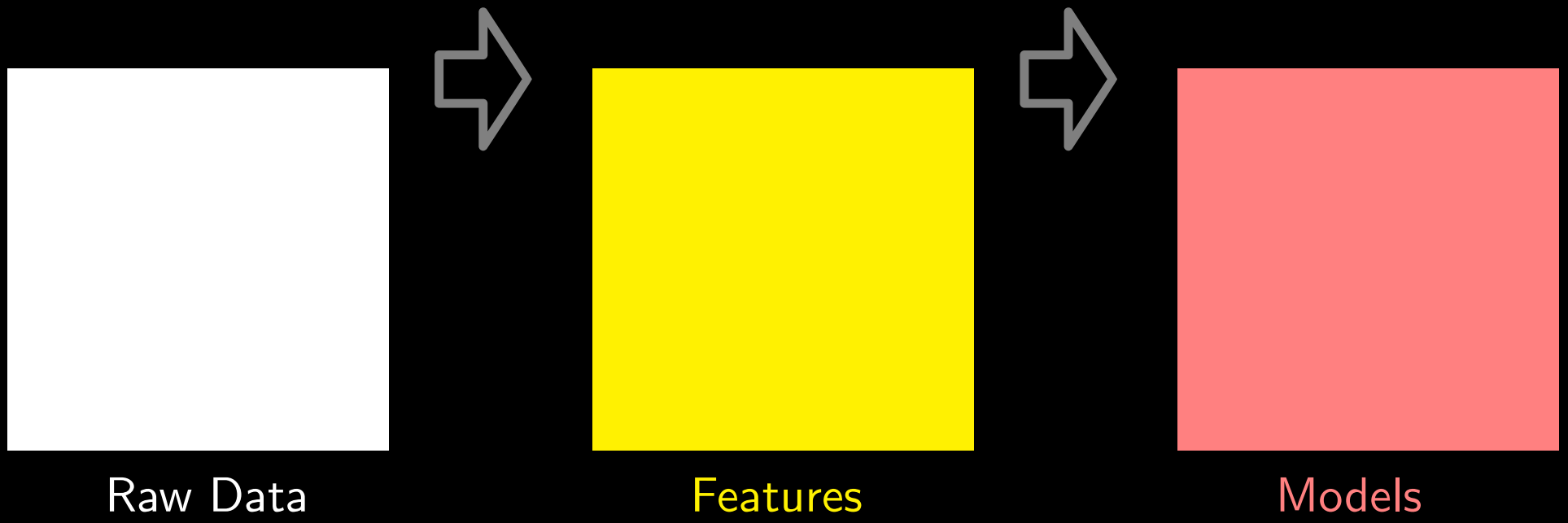
Table of Contents

Preliminaries

Fundamentals

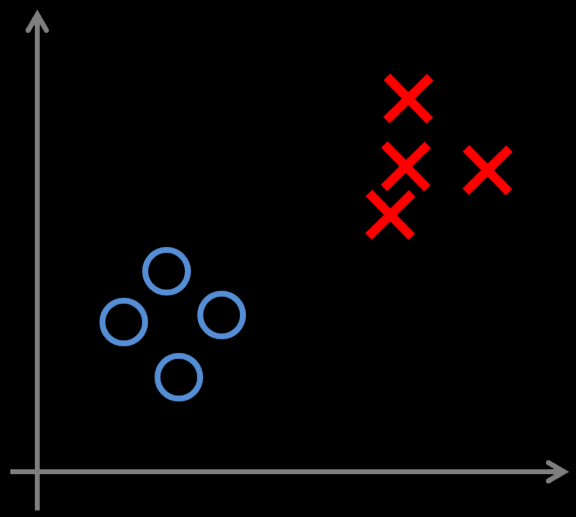
- Optimization, cost/penalty function
- Machine Learning
  - SupervisedLearning & Unsupervised Learning
  - Dimensionality Reduction, PCA & Autoencoders
  - Artificial Neural Networks, Deep Learning, Sparse-Coding

Machine Learning

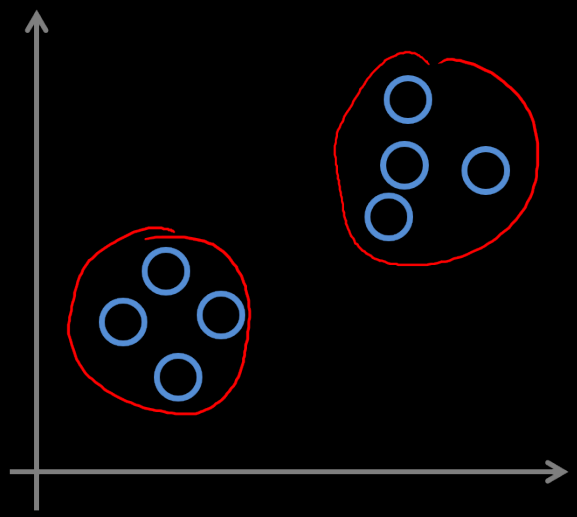


Super/Unsupervised learning

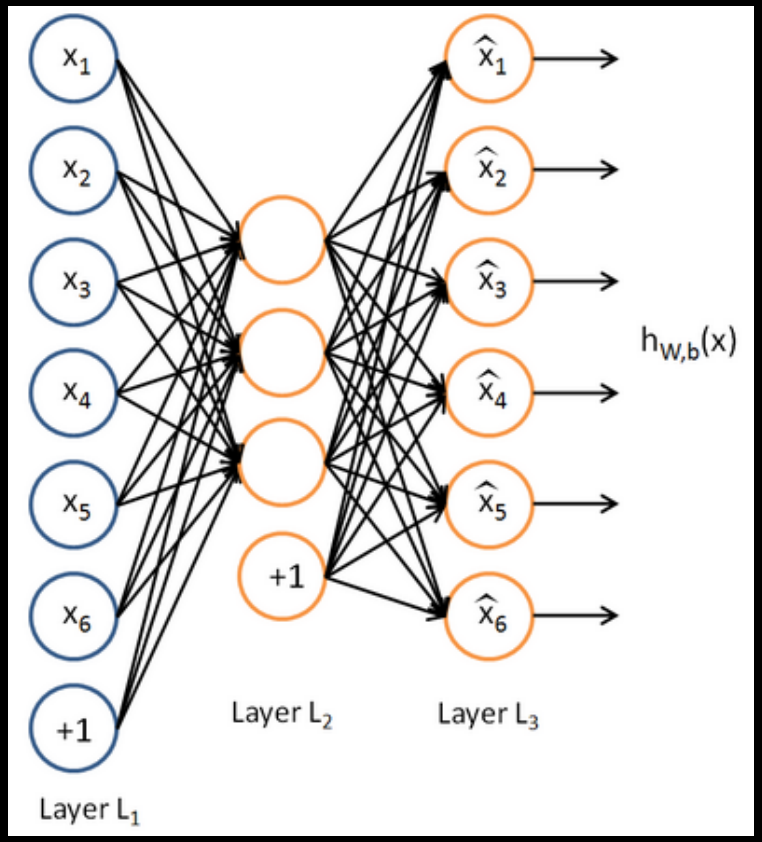
Supervised Learning



Unsupervised Learning



Autoencoders & PCAs



$$\left\| \mathbf{A}\sigma(\mathbf{A}^T\mathbf{x}) - \mathbf{x} \right\|^2$$

Deep Learning & Sparse Coding

Sparse Coding (Olshausen & Field, 1996)

Input:

Data $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}$, (each in $\mathrm{R}^{m \times n}$)

Learn:

Dictionary of bases $\theta_1, \theta_2, \ldots, \theta_k$, each input can be approximately decomposed as:

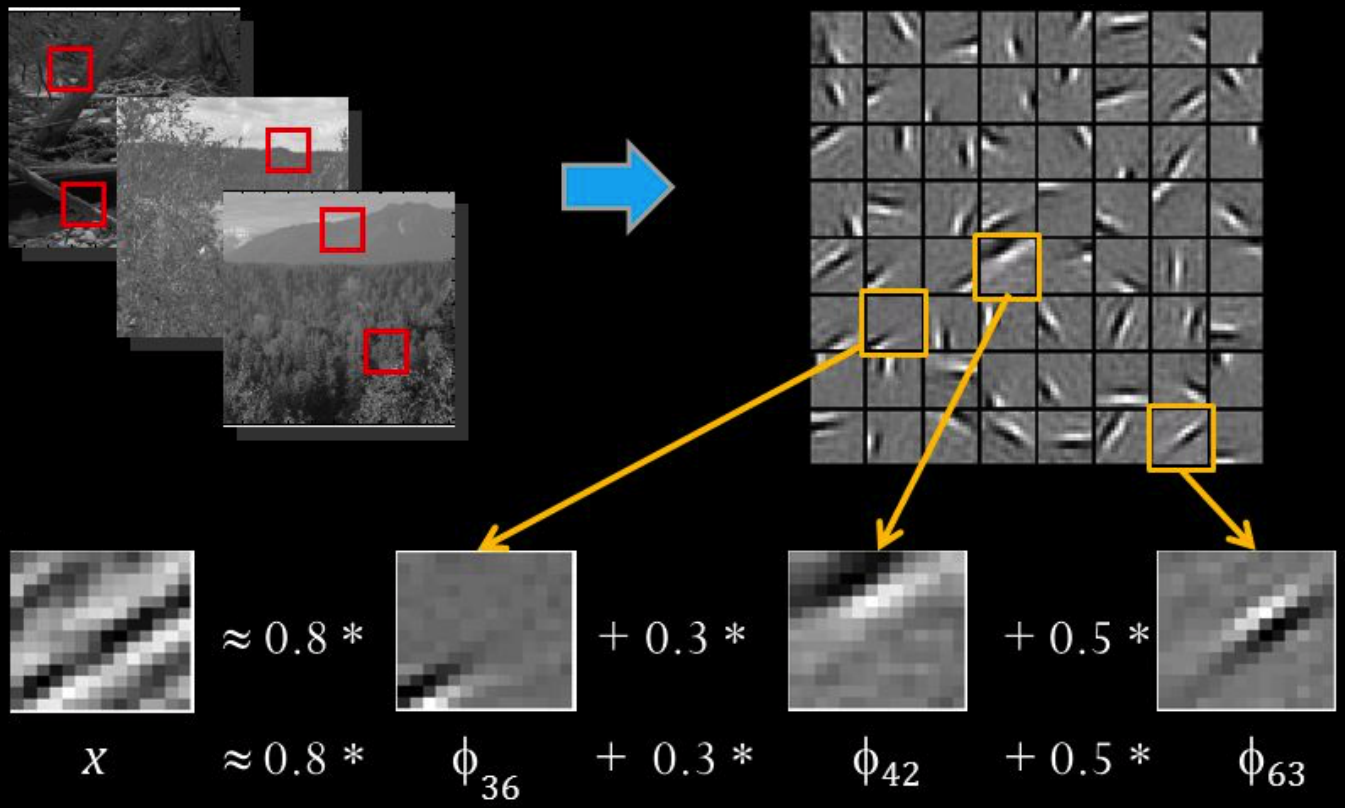
$$\mathbf{x} \approx \sum_{j=1}^{k} a_j \theta_j$$

where a_j are mostly zeros (sparse)

Deep Learning & Sparse Coding

Natural Images

Learned bases $(\theta_1, \ldots, \theta_{64})$:



[0, 0, ..., 0, **0.8**, 0, ..., 0, **0.3**, 0, ..., 0, **0.5**, ...]

= $[a_1, \ldots, a_{64}]$ (feature representation)

Deep Learning & Sparse Coding

NonConvex Formulations

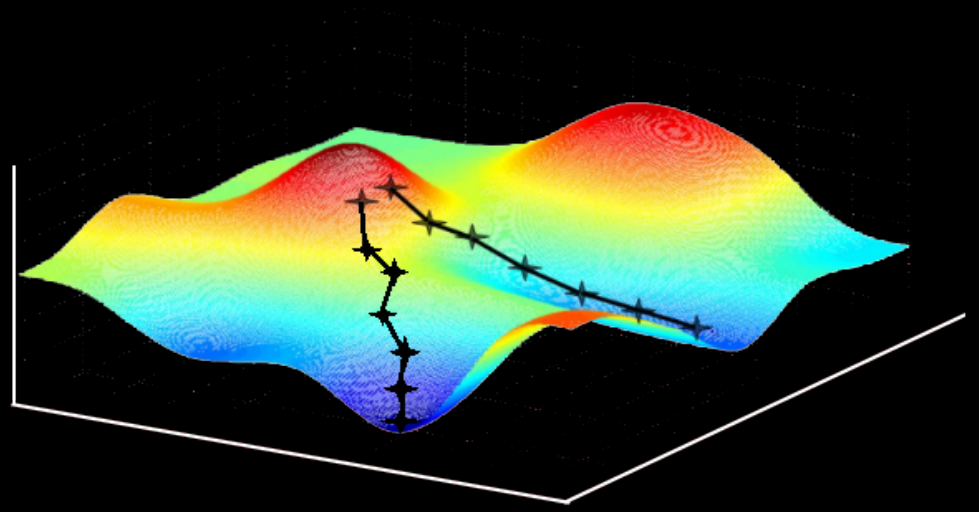
$$\min_{a_i^{(j)},\phi_i} \sum_{j=1}^{m} \underbrace{\left\| \mathbf{x}^{(j)} - \sum_{i=1}^{k} a_i^{(j)} \phi_i \right\|^2}_{\text{reconstruction term}} + \underbrace{\lambda \sum_{i=1}^{k} S(a_i)^{(j)}}_{\text{sparsity penalty}}$$

This optimization technique is **NP-hard**, can have many local optima; but **heuristics** work well empirically.

Deep Learning & Sparse Coding

This network performs a gradient descent by alternating between:

1. $r \quad \blacktriangleleft \quad b - Ax$
2. $x \quad \blacktriangleleft \quad x + \nu(A^T r - \nabla L(x))$



Deep Learning & Sparse Coding

Remember from the Autoencoder slide,

$$\left\| \mathbf{A}\sigma(\mathbf{A}^T\mathbf{x}) - \mathbf{x} \right\|^2, \quad \mathbf{B} = \sigma(\mathbf{A}^T\mathbf{x})$$

Instead we think of it as a neural implementation, the basic idea of the Discriminative Disaggregation Sparse Coding algorithm.

$$\underset{A \geq 0}{\operatorname{argmin}} \|\mathbf{X}_i - \mathbf{B}_i\mathbf{A}\|_F^2 + \underbrace{\lambda \sum_{p,q} \mathbf{A}_{pq}}_{\text{added sparsity}}$$

$\mathbf{A}, \mathbf{B}$ are neural encoders as layers of neuronal synapses. Where the learning comes into $\mathbf{A}$.

Sparse coding pre-training

pre-train activations and basis vectors

Algorithm 1: Dicriminative disaggregation sparse coding

input: data points for each individual source $\mathbf{X}_i \in \mathbb{R}^{T \times m}, i = 1 : k$, regularization $\lambda \in \mathbb{R}_+$, with gradient step size $\alpha \in \mathbb{R}_+$.

Sparse coding pre-training:

1. Initalize $\mathbf{B}_i$, $\mathbf{A}_i; \geq 0$, scale columns $\mathbf{B}_i$ s.t. $\left\| \mathbf{b}_i^{(j)} \right\|_2 = 1$
2. For each $i = 1, \ldots, k$, iterate until convergence:
   $\mathbf{A_i} \leftarrow \text{argmin}_{A \geq 0} \|\mathbf{X}_i - \mathbf{B}_i \mathbf{A}\|_F^2 + \lambda \sum_{p,q} \mathbf{A}_{pq}$
   $\mathbf{B}_i \leftarrow \text{argmin}_{B \geq 0, \left\| \mathbf{b}^{(j)} \right\|_2 \leq 1} \|\mathbf{X}_i - \mathbf{B}\mathbf{A}_i\|_F^2$

Discriminative disaggregation training

perceptron algorithm

Algorithm 2: Dicriminative disaggregation sparse coding

Discriminative disaggregation training:

3. Set $\mathbf{A}^*_{1:k} \leftarrow \mathbf{A}_{1:k}, \hat{\mathbf{B}}_{1:k} \leftarrow \mathbf{B}_{1:k}$.
4. Iterate until convergence:

$$\hat{\mathbf{A}_{1:k}} \leftarrow \text{argmin}_{A_{1:k} \geq 0} F\left(\bar{\mathbf{X}}, \tilde{\mathbf{B}}_{1:k}, \mathbf{A}_{1:k}\right)$$

$$\tilde{\mathbf{B}} \leftarrow \left[\tilde{\mathbf{B}} - \alpha\left((\bar{\mathbf{X}} - \tilde{\mathbf{B}}\hat{\mathbf{A}})\hat{\mathbf{A}}^T - (\bar{\mathbf{X}} - \tilde{\mathbf{B}}\mathbf{A}^*)(\mathbf{A}^*)^T\right)\right]_+$$

$$\forall \quad i, j, \mathbf{b}_i^{(j)} \leftarrow \mathbf{b}_i^{(j)} / \left\|\mathbf{b}_i^{(j)}\right\|_2$$

Given aggregated test examples $\bar{\mathbf{X}}'$

5. $\hat{\mathbf{A}}'_{1:k} \leftarrow \text{argmin}_{\mathbf{A}_{1:k} \geq 0} F(\bar{\mathbf{X}}', \tilde{\mathbf{B}}_{1:k}, \mathbf{A}_{1:k})$
6. Predict $\hat{\mathbf{X}}'_i = \mathbf{B}_i \hat{\mathbf{A}}'_i$

Table of Contents

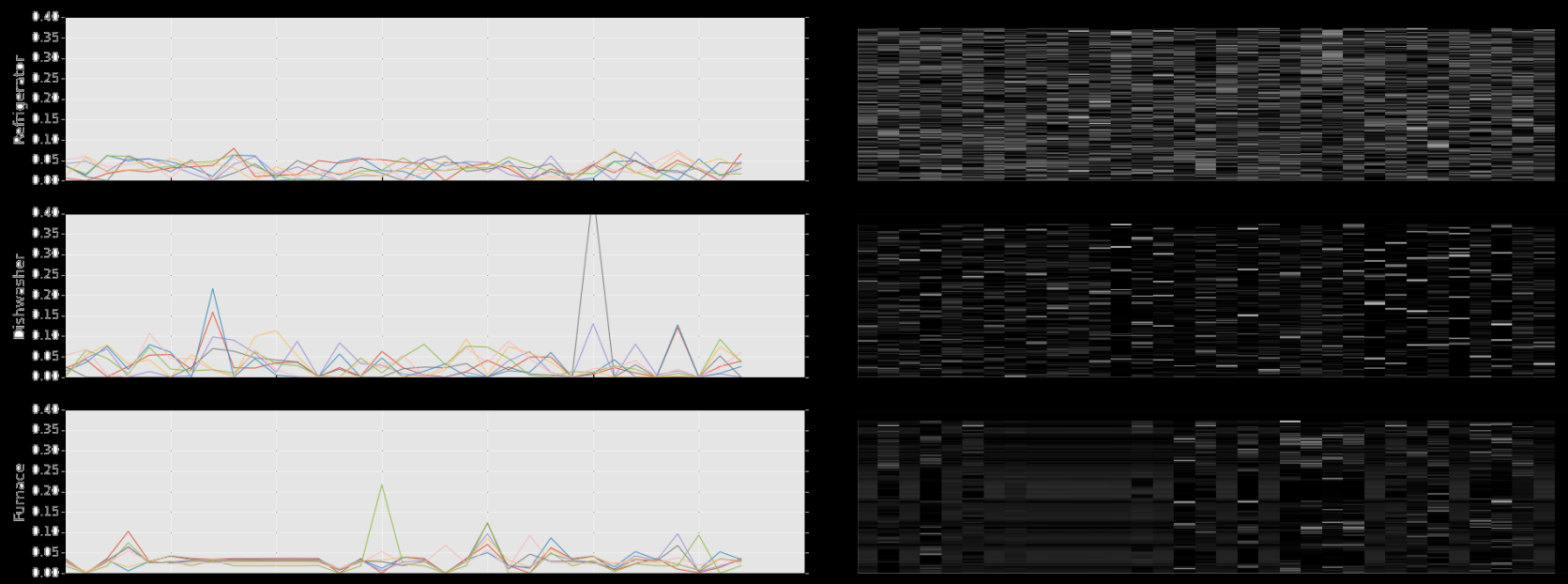
Results and Extensions

Facts

- Kotler Dataset 590 homes, two years. 52 device types.
- **Raw** Data from 2014 and 2015, 8 billion readings, from 21 appliances, 689 houses.
- **Cleaned** Data 213 houses
- Each run (9 runs total excluding grid search) ranged from 4 hours to 24 hours on a 46 GB Ram, quadro Core CPU (Core i7-4790-processor)

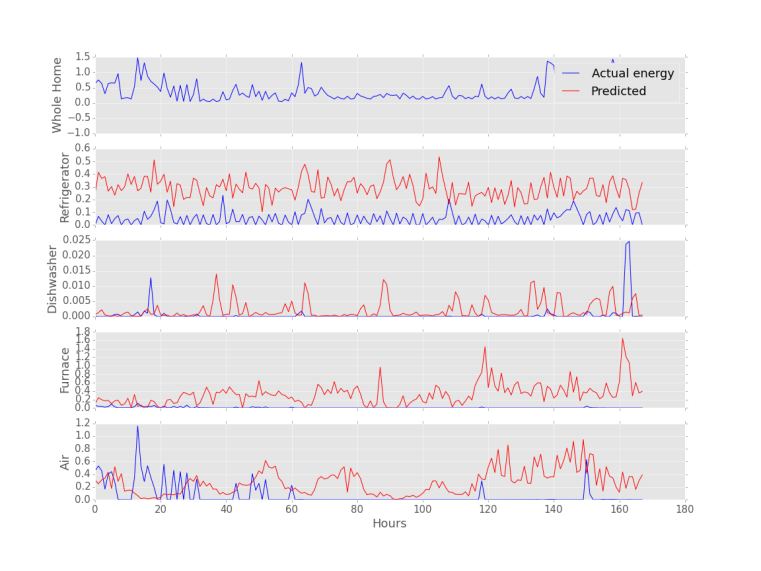
Trained models

Basis Functions



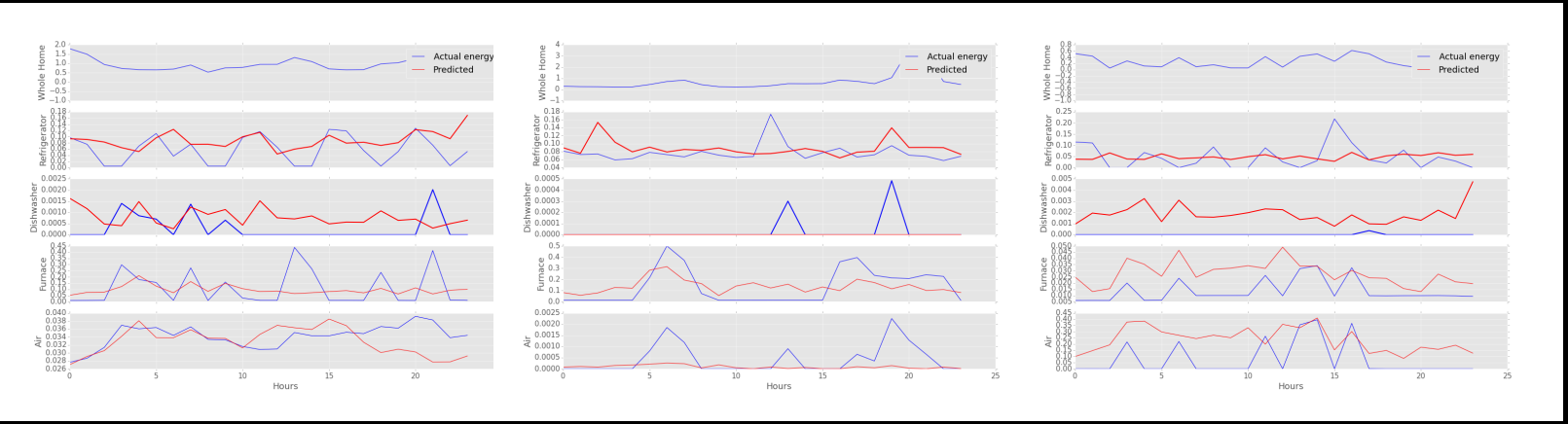
Trained models

DDSC



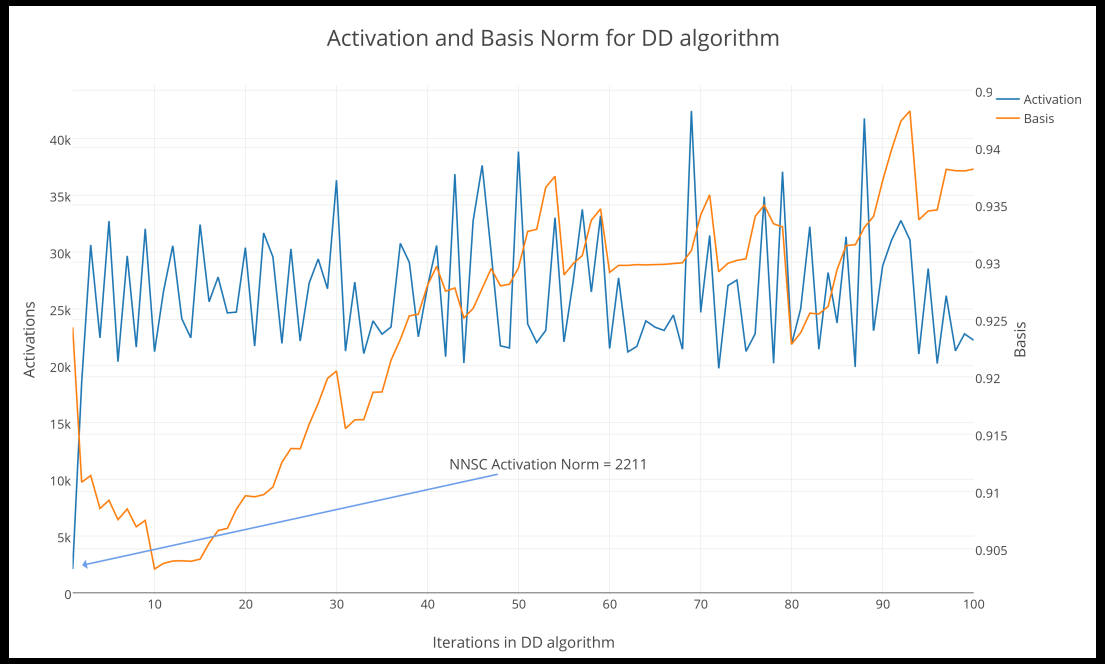
Exploiting temporal difference

Results



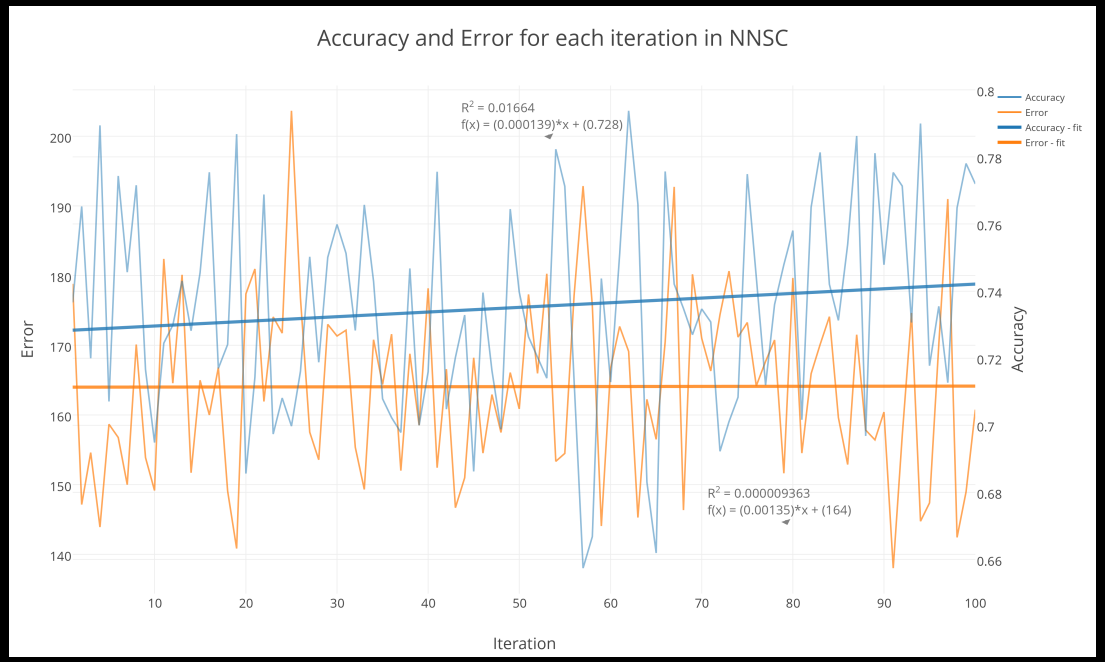
Activations and Basis Norm of DDSC algorithm

Results



Accuracy and Error of the DDSC algorithm

Result



$$\text{Accuracy of Week} \equiv \frac{\sum_{i,q} \min\left\{\sum_p (\mathbf{X}_i)_{pq}, \sum_p (\hat{\mathbf{B}}_i, \mathbf{A}_i)_{pq}\right\}}{\sum_{p,q} \bar{\mathbf{X}}_{p,q}}$$

$$\text{Error}\left(\mathbf{X}_{1:k}, \mathbf{B}_{1:k}\right) := \sum_{i=1}^{k} \frac{1}{2} \left\| \mathbf{X}_i - \mathbf{B}_i \hat{\mathbf{A}}_i \right\|_F^2,$$

Table of Contents

Future Work

Extensions

DDSC

- Gridsearch across parameters; λ, α
- Clustering - label data
- Adding extensions to the model, through Andrew

Total energy priors:

$$F_{TEP}(\bar{\mathbf{X}}, \mathbf{B}_{1:k}, \mathbf{A}_{1:k}) = F(\bar{\mathbf{X}}, \mathbf{B}_{1:k}\mathbf{A}_{1:k}) + \lambda_{TEP} \sum_{i=1}^{k} \left\| \mu_i \mathbf{1}^T - \mathbf{1}^T \mathbf{B}_i \mathbf{A}_i \right\|_2^2$$

Group Lasso:

$$F_{GL}(\bar{\mathbf{X}}, \mathbf{B}_{1:k}, \mathbf{A}_{1:k}) = F(\bar{\mathbf{X}}, \mathbf{B}_{1:k}\mathbf{A}_{1:k}) + \lambda_{GL} \sum_{i=1}^{k} \sum_{j=1}^{m} \left\| \mathbf{a}_i^{(j)} \right\|_2$$

Recent work

- Dropouts, by Hinton et al. '14
- Hyperparameter Optimization
- Block Coordinate Update, '13
- Autoencoders, Google Cat '12, Deep Dream '14
- Simple, Efficient, and Neural Algorithms for Sparse Coding '15
- Final word

Thank you!

Questions?