A Tour of Unsupervised Learning – Part II
Clustering. Semi-supervised, active and reinforcement learning

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

MPS2016
Outline

Clustering – finding groups in data

Clustering: Finding groups in data

Basic algorithms
- K-means clustering and the quadratic distortion
- Model based / soft clustering, the EM algorithm and maximizing likelihood

Similarity based / graph clustering and the Spectral clustering algorithm

Further issues and current trends

Reinforcement, semi-supervised, and active learning
Outline

Clustering – finding groups in data

Clustering: Finding groups in data

Basic algorithms
   K-means clustering and the quadratic distortion
   Model based / soft clustering, the EM algorithm and maximizing likelihood

Similarity based / graph clustering and the Spectral clustering algorithm

Further issues and current trends

Reinforcement, semi-supervised, and active learning
What is clustering? Problem and Notation

- **Informal definition** Clustering = Finding groups in data

- **Notation**
  - \( \mathcal{D} = \{ x_1, x_2, \ldots, x_n \} \) a data set
  - \( n \) = number of data points
  - \( K \) = number of clusters (\( K \ll n \))
  - \( \Delta = \{ C_1, C_2, \ldots, C_K \} \) a partition of \( \mathcal{D} \) into disjoint subsets
  - \( k(i) \) = the label of point \( i \)
  - \( \mathcal{L}(\Delta) \) = cost (loss) of \( \Delta \) (to be minimized)

- **Second informal definition** Clustering = given \( n \) data points, separate them into \( K \) clusters

- **Hard vs. soft clusterings**
  - **Hard** clustering \( \Delta \): an item belongs to only 1 cluster
  - **Soft** clustering \( \gamma = \{ \gamma_{ki} \}_{i=1:n}^{k=1:K} \)
    - \( \gamma_{ki} \) = the degree of membership of point \( i \) to cluster \( k \)
    - \( \sum_k \gamma_{ki} = 1 \) for all \( i \)

(usually associated with a probabilistic model)
(from )
Paradigms

Depend on type of data, type of clustering, type of cost (probabilistic or not), and constraints (about $K$, shape of clusters)

- **Data = vectors** $\{x_i\}$ in $\mathbb{R}^d$
  - **Parametric** Cost based [hard]
  - (K known) Model based [soft]

- **Non-parametric**
  - Dirichlet process mixtures [soft]
  - (K determined by algorithm) Information bottleneck [soft]
  - Modes of distribution [hard]
  - Gaussian blurring mean shift [hard]

- **Data = similarities** between pairs of points $[S_{ij}]_{i,j=1:n}$,
  - $S_{ij} = S_{ji} \geq 0$ (called Similarity based clustering)
  - Graph partitioning spectral clustering [hard, K fixed, cost based]
  - typical cuts [hard non-parametric, cost based]
  - Affinity propagation [hard/soft non-parametric]
## Classification vs Clustering

<table>
<thead>
<tr>
<th></th>
<th>Classification</th>
<th>Clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Performance criterion</strong></td>
<td>Expected error</td>
<td>a wide variety</td>
</tr>
<tr>
<td></td>
<td>Supervised</td>
<td>Unsupervised</td>
</tr>
<tr>
<td><strong>Generalization</strong></td>
<td>Performance on new</td>
<td>Performance on current</td>
</tr>
<tr>
<td></td>
<td>data is what matters</td>
<td>data is what matters</td>
</tr>
<tr>
<td><strong>K</strong></td>
<td>Known</td>
<td>Unknown</td>
</tr>
<tr>
<td><strong>“Goal”</strong></td>
<td>Prediction</td>
<td>Exploration, etc</td>
</tr>
<tr>
<td><strong>Stage of field</strong></td>
<td>Mature</td>
<td>Young: new paradigms and theoretical results are emerging</td>
</tr>
</tbody>
</table>
Outline

Clustering – finding groups in data

Clustering: Finding groups in data

Basic algorithms
  K-means clustering and the quadratic distortion
  Model based / soft clustering, the EM algorithm and maximizing likelihood

Similarity based / graph clustering and the Spectral clustering algorithm

Further issues and current trends

Reinforcement, semi-supervised, and active learning
K-means clustering

**Algorithm K-Means**

**Input** Data $\mathcal{D} = \{x_i\}_{i=1:n}$, number clusters $K$

**Initialize** centers $\mu_1, \mu_2, \ldots \mu_K \in \mathbb{R}^d$ at random

**Iterate** until convergence

1. for $i = 1 : n$ (assign points to clusters $\Rightarrow$ new clustering)
   
   $$k(i) = \arg\min_k ||x_i - \mu_k||$$

2. for $k = 1 : K$ (recalculate centers)

   $$\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i \quad (1)$$

▶ Convergence
▶ if $\Delta$ doesn’t change at iteration $m$ it will never change after that
▶ convergence in finite number of steps to local optimum of cost $L$ (defined next)

▶ therefore, initialization will matter
K-means clustering

## Algorithm K-Means

**Input** Data $\mathcal{D} = \{x_i\}_{i=1:n}$, number clusters $K$

**Initialize** centers $\mu_1, \mu_2, \ldots \mu_K \in \mathbb{R}^d$ at random

**Iterate** until convergence

1. for $i = 1 : n$ (assign points to clusters $\Rightarrow$ new clustering)
   
   $$k(i) = \arg\min_k ||x_i - \mu_k||$$

2. for $k = 1 : K$ (recalculate centers)
   
   $$\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$ \hspace{1cm} (1)

**Convergence**

- if $\Delta$ doesn’t change at iteration $m$ it will never change after that
- convergence in finite number of steps

$L$ (defined next)
K-means clustering

**Algorithm K-Means**

**Input**  Data $\mathcal{D} = \{x_i\}_{i=1:n}$, number clusters $K$

**Initialize** centers $\mu_1, \mu_2, \ldots \mu_K \in \mathbb{R}^d$ at random

**Iterate** until convergence

1. for $i = 1 : n$ (assign points to clusters $\Rightarrow$ new clustering)
   
   $k(i) = \arg\min_k ||x_i - \mu_k||$

2. for $k = 1 : K$ (recalculate centers)
   
   $\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i \quad (1)$

**Convergence**

- if $\Delta$ doesn’t change at iteration $m$ it will never change after that
- convergence in finite number of steps to local optimum of cost $\mathcal{L}$ (defined next)
**K-means clustering**

---

**Algorithm K-Means**

**Input**  
Data $\mathcal{D} = \{x_i\}_{i=1:n}$, number clusters $K$

**Initialize**  
centers $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^d$ at random

**Iterate**  
until convergence

1. for $i = 1 : n$ (assign points to clusters ⇒ new clustering)
   
   $$k(i) = \arg\min_k ||x_i - \mu_k||$$

2. for $k = 1 : K$ (recalculate centers)
   
   $$\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$ (1)

---

**Convergence**

- if $\Delta$ doesn’t change at iteration $m$ it will never change after that
- convergence in finite number of steps to local optimum of cost $\mathcal{L}$ (defined next)
- therefore, initialization will matter
The K-means cost

\[ \mathcal{L}(\Delta) = \sum_{k=1}^{K} \sum_{i \in C_k} ||x_i - \mu_k||^2 \]

- K-means solves a least-squares problem
- the cost \( \mathcal{L} \) is called quadratic distortion

**Proposition** The K-means algorithm decreases \( \mathcal{L}(\Delta) \) at every step.

Sketch of proof
- step 1: reassigning the labels can only decrease \( \mathcal{L} \)
- step 2: reassigning the centers \( \mu_k \) can only decrease \( \mathcal{L} \) because \( \mu_k \) as given by (1) is the solution to

\[ \mu_k = \min_{\mu \in \mathbb{R}^d} \sum_{i \in C_k} ||x_i - \mu||^2 \] (2)
Initialization of the centroids $\mu_1:K$

- Idea 1: start with $K$ points at random

- Idea 2: start with $K$ data points at random

What's wrong with choosing $K$ data points at random?

The probability of hitting all $K$ clusters with $K$ samples approaches 0 when $K > 5$

- Idea 3: start with $K$ data points using Fastest First Traversal (greedy simple approach to spread out centers)

- Idea 4: $k$-means++ (randomized, theoretically backed approach to spread out centers)

- Idea 5: “$K$-log$K$” Initialization (start with enough centers to hit all clusters, then prune down to $K$)

For EM Algorithm, for $K$-means
Initialization of the centroids $\mu_1:K$

- Idea 1: start with $K$ points at random
- Idea 2: start with $K$ data points at random

What's wrong with choosing $K$ data points at random?

The probability of hitting all $K$ clusters with $K$ samples approaches 0 when $K > 5$.

- Idea 3: start with $K$ data points using Fastest First Traversal (greedy simple approach to spread out centers)
- Idea 4: k-means++ (randomized, theoretically backed approach to spread out centers)
- Idea 5: "K-logK" Initialization (start with enough centers to hit all clusters, then prune down to $K$)
Initialization of the centroids $\mu_1:K$

- Idea 1: start with $K$ points at random
- Idea 2: start with $K$ data points at random
  What’s wrong with choosing $K$ data points at random?

The probability of hitting all $K$ clusters with $K$ samples approaches 0 when $K > 5$
Initialization of the centroids $\mu_{1:K}$

- Idea 1: start with $K$ points at random
- Idea 2: start with $K$ data points at random

What’s wrong with choosing $K$ data points at random?

The probability of hitting all $K$ clusters with $K$ samples approaches 0 when $K > 5$

- Idea 3: start with $K$ data points using Fastest First Traversal (greedy simple approach to spread out centers)
Initialization of the centroids $\mu_{1:K}$

- **Idea 1**: start with $K$ points at random
- **Idea 2**: start with $K$ data points at random
  What’s wrong with choosing $K$ data points at random?

The probability of hitting all $K$ clusters with $K$ samples approaches 0 when $K > 5$

- **Idea 3**: start with $K$ data points using Fastest First Traversal (greedy simple approach to spread out centers)
- **Idea 4**: k-means++ (randomized, theoretically backed approach to spread out centers)
Initialization of the centroids $\mu_1:K$

- Idea 1: start with $K$ points at random
- Idea 2: start with $K$ data points at random

What’s wrong with choosing $K$ data points at random?

The probability of hitting all $K$ clusters with $K$ samples approaches 0 when $K > 5$

- Idea 3: start with $K$ data points using Fastest First Traversal (greedy simple approach to spread out centers)
- Idea 4: k-means++ (randomized, theoretically backed approach to spread out centers)
- Idea 5: “K-logK” Initialization (start with enough centers to hit all clusters, then prune down to $K$)

For EM Algorithm , for K-means
The “K-logK” initialization

The K-logK Initialization (see also )
1. pick $\mu_{1:K'}^0$ at random from data set, where $K' = O(K \log K)$
   (this assures that each cluster has at least 1 center w.h.p)
2. run 1 step of K-means
3. remove all centers $\mu_k^0$ that have few points, e.g $|C_k| < \frac{n}{eK'}$
4. from the remaining centers select $K$ centers by Fastest First Traversal
   4.1 pick $\mu_1$ at random from the remaining $\{\mu_{1:K'}^0\}$
   4.2 for $k = 2 : K$, $\mu_k \leftarrow \text{argmax}_{\mu_{k'}^0} \min_{j=1:k-1} ||\mu_{k'}^0 - \mu_j||$, i.e next $\mu_k$ is furthest away
       from the already chosen centers
5. continue with the standard K-means algorithm
K-means clustering with K-logK Initialization

Example using a mixture of 7 Normal distributions with 100 outliers sampled uniformly

K-\text{LOG}K \ K = 7, \ T = 100, \ n = 1100, \ c = 1

\text{Naive} \ K = 7 \ T = 100, \ n = 1100
Model based clustering: Mixture models

- The mixture density

\[ f(x) = \sum_{k=1}^{K} \pi_k f_k(x) \]

- \( f_k(x) \) = the components of the mixture
  - each is a density
  - \( f \) called mixture of Gaussians if \( f_k = \text{Normal}_{\mu_k, \Sigma_k} \)

- \( \pi_k \) = the mixing proportions,
  \[ \sum_{k=1}^{K} \pi_k = 1, \quad \pi_k \geq 0. \]

- model parameters \( \theta = (\pi_1:K, \mu_1:K, \Sigma_1:K) \)
Model based clustering: Mixture models

- The mixture density

\[ f(x) = \sum_{k=1}^{K} \pi_k f_k(x) \]

- \( f_k(x) \) = the components of the mixture
  - each is a density
  - \( f \) called mixture of Gaussians if \( f_k = \text{Normal}_{\mu_k, \Sigma_k} \)

- \( \pi_k \) = the mixing proportions,
  \( \sum_k = 1^K \pi_k = 1, \pi_k \geq 0 \).

- model parameters \( \theta = (\pi_1:K, \mu_1:K, \Sigma_1:K) \)

- The degree of membership of point \( i \) to cluster \( k \)

\[ \gamma_{ki} \overset{\text{def}}{=} P[x_i \in C_k] = \frac{\pi_k f_k(x)}{f(x)} \quad \text{for} \quad i = 1 : n, \quad k = 1 : K \]

- depends on \( x_i \) and on the model parameters
The Expectation-Maximization (EM) Algorithm

**Algorithm Expectation-Maximization (EM)**

**Input** Data $\mathcal{D} = \{x_i\}_{i=1:n}$, number clusters $K$

**Initialize** parameters $\pi_{1:K} \in \mathbb{R}$, $\mu_{1:K} \in \mathbb{R}^d$, $\Sigma_{1:K} \in \mathbb{R}^{d \times d}$

**Iterate** until convergence

**E step** (Optimize clustering) for $i = 1 : n$, $k = 1 : K$

$$\text{compute } \gamma_{ki} = \frac{\pi_k f_k(x)}{f(x)}$$

**M step** (Optimize parameters)

- Compute “number of points” in cluster $k$

$$\Gamma_k = \sum_{i=1}^{m} \gamma_{ki}, \ k = 1 : K \quad \text{(note: } \sum_k \Gamma_k = n) \quad (6)$$

- Estimate parameters

$$\pi_k = \frac{\Gamma_k}{n}, \ k = 1 : K$$

$$\mu_k = \frac{\sum_{i=1}^{n} \gamma_{ki} x_i}{\Gamma_k}$$

$$\Sigma_k = \frac{\sum_{i=1}^{n} \gamma_{ki} (x_i - \mu_k)(x_i - \mu_k)^T}{\Gamma_k}$$
EM versus K-means

- Alternates between cluster assignments and parameter estimation
- Cluster assignments $\gamma_{ki}$ are probabilistic
- Cluster parametrization more flexible

- Converges to local optimum of log-likelihood
  Initialization recommended by K-logK method

- Modern algorithms with guarantees (for e.g. mixtures of Gaussians)
  - Random projections
  - Projection on principal subspace
  - Two step EM (K-logK initialization + one more EM iteration)
Outline

Clustering – finding groups in data

Clustering: Finding groups in data

Basic algorithms
  - K-means clustering and the quadratic distortion
  - Model based / soft clustering, the EM algorithm and maximizing likelihood

Similarity based / graph clustering and the Spectral clustering algorithm

Further issues and current trends

Reinforcement, semi-supervised, and active learning
Similarity based clustering

- **Paradigm:** the features we observe are measures of similarity/dissimilarity between pairs of data points, e.g.

<table>
<thead>
<tr>
<th>points</th>
<th>features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image segmentation</td>
<td>pixels</td>
</tr>
<tr>
<td></td>
<td>distance in color space, location, separated by a contour, belong to same texture</td>
</tr>
<tr>
<td>Social network</td>
<td>people</td>
</tr>
<tr>
<td>Text analysis</td>
<td>words</td>
</tr>
<tr>
<td></td>
<td>appear in same context</td>
</tr>
</tbody>
</table>

- The features are summarized by a single similarity measure $S_{ij}$
  - e.g. $S_{ij} = e^{\sum_k \alpha_k \text{feature}_k(i,j)}$ for all points $i,j$
  - symmetric $S_{ij} = S_{ji}$
  - non-negative $S_{ij} \geq 0$
Similarity based clustering

**Paradigm:** the features we observe are measures of *similarity/dissimilarity* between pairs of data points, e.g

<table>
<thead>
<tr>
<th>points</th>
<th>features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image segmentation</td>
<td>pixels distance in color space, location, separated by a contour, belong to same texture</td>
</tr>
<tr>
<td>Social network</td>
<td>people friendship, coauthorship, hyperlinks, email</td>
</tr>
<tr>
<td>Text analysis</td>
<td>words appear in same context</td>
</tr>
</tbody>
</table>

The features are summarized by a single similarity measure $S_{ij}$

- e.g $S_{ij} = e^{\sum_k \alpha_k \text{feature}_k(i,j)}$ for all points $i, j$
- symmetric $S_{ij} = S_{ji}$
- non-negative $S_{ij} \geq 0$

Mathematically, we can see the data as

- a $n \times n$ matrix $S = [S_{ij}]$
- a (weighted) graph
  - points = graph nodes, similarity $S_{ij} =$ weight of edge $ij$
  - meaningful because very few similarities are large
  Then, clustering is *cutting* the graph
Artificial matrix  UW Statistics co-authorship data similarity based useful for $\mathbb{R}^d$ data as well

$S_{ij} = \# \text{tech-reports co-authored}$

$S_{ij} = e^{-\frac{||x_i - x_j||^2}{\sigma^2}}$
Criteria for clustering

- **Graph cuts**
  - remove some edges $\implies$ disconnected graph
  - the groups are the connected components

- **By similar behavior**
  - nodes $i, j$ in the same group iff $i, j$ have the same pattern of transitions at group level

- **By Embedding**
  - map graph nodes $\{1, 2, \ldots, n\}$ to $\mathbb{R}^K$ then use “standard” clustering methods (e.g. K-means)

- **By diffusion distance**

- All are equivalent (approximately) when the data is clusterable
Spectral clustering in a nutshell

weighted graph $\rightarrow$ similarity matrix $S$ $\rightarrow$ transition matrix $P$ $\rightarrow$ first $K$ eigenvectors of $P$ $\rightarrow$ $K$ clusters

$n$ vertices to cluster; observations are pairwise similarities

$n \times n$, symmetric $S_{ij} \geq 0$

normalize rows $\rightarrow$ spectral mapping $\rightarrow$ clustering in $\mathbb{R}^K$
Spectral clustering in a nutshell

Similarity $S$

Preprocess

Top K e-vectors $v^{1:K}$

Data embedded by $v$

Cluster with K-means

$$P = D^{-1}S$$

point $i$
Spectral clustering in a nutshell

Similarity $S$

Preprocess

Top $K$ e-vectors $\mathbf{v}^{1:K}$

Data embedded by $\mathbf{v}$

Cluster with K-means

$P = D^{-1}S$
Properties of spectral clustering

- Arbitrary cluster shapes (main advantage)
- Elegant mathematically
- Practical up to medium sized problems
  - Running time (by Lanczos algorithm) $\mathcal{O}(nk)/\text{iteration}$.
- Works well when $K$ known, not too large estimating $K$
- Depend heavily on the similarity function (main problem) learning the similarities ,,,
- Outliers become separate clusters (user must adjust $K$ accordingly!)
- Very popular, many variants which aim to improve on the above
  Diffusion maps: normalize the eigenvectors $\lambda_k v_k$
- Practical fix, when $K$ large: only compute a fixed number of eigenvectors $d < K$. This avoids the effects of noise in lower ranked eigenvectors
Understanding spectral clustering I

- **Graph cuts** Spectral clustering minimizes $\text{MNCut}(\Delta) = \sum_{k=1}^{K} \sum_{k' \neq k} \frac{\text{Cut}(C_k, C_{k'})}{D_{C_k}}$

  (not the smallest $K$-way cut!)

- By similar behavior in the random walk on the graph

  $P_{\text{red}} = P_{i \rightarrow \text{red} | i} = \sum_{j \in \text{red}} P_{ij}$

  $P_{\text{yellow}} = \frac{1}{5} 4/5$

  $\text{red}$

  $\text{yellow}$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$

  $1/5 4/5$
item All are equivalent (approximately) when the data is clusterable. Clusterability characterized by
Outline

Clustering – finding groups in data

Clustering: Finding groups in data

Basic algorithms
   K-means clustering and the quadratic distortion
   Model based / soft clustering, the EM algorithm and maximizing likelihood

Similarity based / graph clustering and the Spectral clustering algorithm

Further issues and current trends

Reinforcement, semi-supervised, and active learning
Further issues and current trends (an incomplete list) I

- Selecting $K$
  - many ad-hoc methods
  - BIC (statistical model selection method) for mixture models theoretically unsupported in clustering
  - stability-based selection (a bootstrap like method)
    - Idea: if a clustering $\Delta$ is supported by the data then it is stable to perturbations
    - theoretical results only preliminary, but empirical evidence promising

- Non-parametric clustering
  - mixture models with unbounded $K$ (known as Dirichlet Process Mixtures, or Bayesian Nonparametric clustering)
  - methods based on a kernel density estimator
    - find the peaks of the density Mean-Shift, Gaussian Blurring Mean-Shift
    - level-set methods (find the high density regions)
  - for similarity data Affinity Propagation

- Clusterability
  - Algorithms with guarantees for clusterable data
  - Scalable algorithms
Clustering – finding groups in data

Clustering: Finding groups in data

Basic algorithms
- K-means clustering and the quadratic distortion
- Model based / soft clustering, the EM algorithm and maximizing likelihood

Similarity based / graph clustering and the Spectral clustering algorithm

Further issues and current trends

Reinforcement, semi-supervised, and active learning
Statistical decisions

Or learning to act in uncertainty (closer to supervised, since objective known)

- Influence diagrams
  - extend graphical models with decision nodes

- Active learning
  - Variant of supervised learning
  - Learner can choose next samples $X_i$ to query, oracle returns the $Y_i$ values

- Semi-supervised learning
  - Variant of supervised learning
  - In addition to labeled pairs $\mathcal{D}_L = \{(X_i, Y_i), i = 1 : m\}$ there are also unlabeled data $\mathcal{D}_U = \{X_j, j = 1 : m\}$.
  - transductive learning: infer the labels for $\mathcal{D}_U$
  - OR, use $\mathcal{D}_U \cup \mathcal{D}_L$ to learn a predictor for $Y$

- Reinforcement learning
  - Sequential (Markov) decisions in uncertainty (aka Markov Decision Process (MDP))
Reinforcement learning

The problem an agent learns how to act in an unknown environment

- Examples: learning to balance a beam, to drive a car, to play backgammon, navigate a maze

Time \( t = 1, 2, 3, \ldots \)
State \( x_t \in \Omega \)
Action \( a_t \in A \) a set of available actions (may depend on \( x \))
Transition probability \( Pr[x_{t+1}|x_t, a_t] \) (Markov transitions with \( T^a \) transition matrix, \( a \in A \))
Reward \( r_t \) stochastic, depends on \( a_t, x_t, x_{t+1} \)

- Goal of agent: maximize \( \mathbb{E} \left[ \sum_{t=1}^{\infty} \gamma^t r_t \right] \) where \( \gamma \in (0, 1) \) is a discount factor

- Basic concepts
  - policy \( \pi : \Omega \rightarrow A \) prescribes an action for every state
  - value function of policy \( \pi \) \( V^\pi : \Omega \rightarrow \mathbb{R} \)

\[
V^\pi(x) = R(x, \text{follow } \pi)
\]

- Note that for each \( \pi \), \( V_\pi \) is a linear function of expected \( r(x, a) \) \( r^\pi + \gamma T^\pi V^\pi = V^\pi \)

- Classic result Bellman equation

the optimal policy \( \pi^* \) satisfies

\[
V^*(x), \pi^*(x) = \max, \argmax_{a \in A} E_{T^a} \left[ r(a, x, x') + \gamma V^*(x') \right]
\]
Reinforcement learning

Approaches to learning

▶ Finite time horizon, small $\Omega$: dynamic programming
▶ $\Omega$ tractable: stochastic optimization for $V^*$, e.g Q-learning
▶ Large $\Omega$, possibly continuous: functional approximations of $V^*$ (may not converge!), e.g neuro-dynamic programming, deep Q-learning
▶ Better yet, functional approximation of the policy; i.e. let $\pi(x) = f(x, \theta)$ and optimize $\theta$ by gradient descent

A harder problem Partially observed MDP (POMDP)

▶ $x_t$ not observed directly
▶ instead, partial information $y_t$
▶ problem becomes non-Markov, sufficient statistics are all $a_t, y_t$ history

Connection with on-line learning, game theory, ...
What I wish I could have included

- Sparse estimation (born @UCLA)
- Model selection
- Other non-parametric models (e.g. shape constrained estimation)
Thank you