An Overview of Other Topics, Conclusions, and Perspectives

Piyush Rai

Machine Learning (CS771A)

Nov 11, 2016

Plan for today..

- Survey of some other topics
  - Sparse Modeling
  - Time-series modeling
  - Reinforcement Learning
  - Multitask/Transfer Learning
  - Active Learning
  - Bayesian Learning
- Conclusion and take-aways..

Sparse Modeling

Sparse Modeling

- Many ML problems can be written in the following general form

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{w} + \boldsymbol{\epsilon}$$

where $\boldsymbol{y}$ is $N \times 1$, $\mathbf{X}$ is $N \times D$, $\boldsymbol{w}$ is $D \times 1$, and $\boldsymbol{\epsilon}$ denotes observation noise

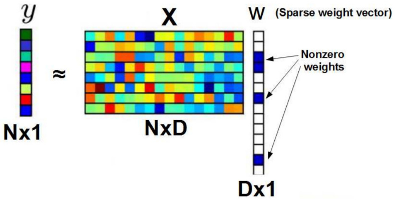
Sparse Modeling

- Many ML problems can be written in the following general form

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{w} + \epsilon$$

where $\boldsymbol{y}$ is $N \times 1$, $\mathbf{X}$ is $N \times D$, $\boldsymbol{w}$ is $D \times 1$, and ϵ denotes observation noise

- Often we expect/want $\boldsymbol{w}$ to be sparse (i.e., at most, say $s \ll D$, nonzeros)



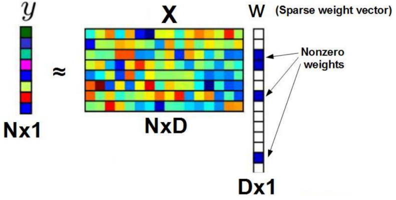
Sparse Modeling

- Many ML problems can be written in the following general form

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{w} + \epsilon$$

where $\boldsymbol{y}$ is $N \times 1$, $\mathbf{X}$ is $N \times D$, $\boldsymbol{w}$ is $D \times 1$, and ϵ denotes observation noise

- Often we expect/want $\boldsymbol{w}$ to be sparse (i.e., at most, say $s \ll D$, nonzeros)



- Becomes especially important when $D >> N$

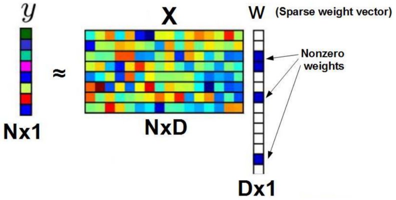
Sparse Modeling

- Many ML problems can be written in the following general form

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{w} + \epsilon$$

where $\boldsymbol{y}$ is $N \times 1$, $\mathbf{X}$ is $N \times D$, $\boldsymbol{w}$ is $D \times 1$, and ϵ denotes observation noise

- Often we expect/want $\boldsymbol{w}$ to be sparse (i.e., at most, say $s \ll D$, nonzeros)



- Becomes especially important when $D >> N$
- Examples: Sparse regression/classification, sparse matrix factorization, compressive sensing, dictionary learning, and many others.

Sparse Modeling

- Ideally, our goal will be to solve the following problem

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 \qquad \text{s.t. } ||\boldsymbol{w}||_0 \leq s$$

- Note: $||\boldsymbol{w}||_0$ is known as the ℓ_0 norm of $\boldsymbol{w}$

Sparse Modeling

- Ideally, our goal will be to solve the following problem

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 \qquad \text{s.t. } ||\boldsymbol{w}||_0 \leq s$$

- Note: $||\boldsymbol{w}||_0$ is known as the ℓ_0 norm of $\boldsymbol{w}$
- In general, an NP-hard problem: Combinatorial optimization problem with $\sum_{i=1}^{s} \binom{D}{i}$ possible locations of nonzeros in $\boldsymbol{w}$

Sparse Modeling

- Ideally, our goal will be to solve the following problem

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 \qquad \text{s.t. } ||\boldsymbol{w}||_0 \leq s$$

- Note: $||\boldsymbol{w}||_0$ is known as the ℓ_0 norm of $\boldsymbol{w}$
- In general, an NP-hard problem: Combinatorial optimization problem with $\sum_{i=1}^{s} \binom{D}{i}$ possible locations of nonzeros in $\boldsymbol{w}$
- Also, the constraint $||\boldsymbol{w}||_0 \leq s$ is non-convex (figure on next slide).

Sparse Modeling

- Ideally, our goal will be to solve the following problem

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 \qquad \text{s.t. } ||\boldsymbol{w}||_0 \leq s$$

- Note: $||\boldsymbol{w}||_0$ is known as the ℓ_0 norm of $\boldsymbol{w}$
- In general, an NP-hard problem: Combinatorial optimization problem with $\sum_{i=1}^{s} \binom{D}{i}$ possible locations of nonzeros in $\boldsymbol{w}$
- Also, the constraint $||\boldsymbol{w}||_0 \leq s$ is non-convex (figure on next slide).
- A huge body of work on solving these problems. Primarily in two categories

Sparse Modeling

- Ideally, our goal will be to solve the following problem

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 \qquad \text{s.t. } ||\boldsymbol{w}||_0 \leq s$$

- Note: $||\boldsymbol{w}||_0$ is known as the ℓ_0 norm of $\boldsymbol{w}$
- In general, an NP-hard problem: Combinatorial optimization problem with $\sum_{i=1}^{s}\binom{D}{i}$ possible locations of nonzeros in $\boldsymbol{w}$
- Also, the constraint $||\boldsymbol{w}||_0 \leq s$ is non-convex (figure on next slide).
- A huge body of work on solving these problems. Primarily in two categories
  - Convex-ify the problem (e.g., replace the ℓ_0 norm by the ℓ_1 norm)

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 \quad \text{s.t. } ||\boldsymbol{w}||_1 \leq t \quad \text{OR} \quad \arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 + \lambda||\boldsymbol{w}||_1$$

Sparse Modeling

- Ideally, our goal will be to solve the following problem

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 \qquad \text{s.t. } ||\boldsymbol{w}||_0 \leq s$$

- Note: $||\boldsymbol{w}||_0$ is known as the ℓ_0 norm of $\boldsymbol{w}$
- In general, an NP-hard problem: Combinatorial optimization problem with $\sum_{i=1}^{s} \binom{D}{i}$ possible locations of nonzeros in $\boldsymbol{w}$
- Also, the constraint $||\boldsymbol{w}||_0 \leq s$ is non-convex (figure on next slide).
- A huge body of work on solving these problems. Primarily in two categories
  - Convex-ify the problem (e.g., replace the ℓ_0 norm by the ℓ_1 norm)

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 \quad \text{s.t. } ||\boldsymbol{w}||_1 \leq t \quad \text{OR} \quad \arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 + \lambda||\boldsymbol{w}||_1$$

(note: the ℓ_1 constraint makes the objective non-diff. at 0, but many ways to handle this)

Sparse Modeling

- Ideally, our goal will be to solve the following problem

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 \qquad \text{s.t. } ||\boldsymbol{w}||_0 \leq s$$

- Note: $||\boldsymbol{w}||_0$ is known as the ℓ_0 norm of $\boldsymbol{w}$
- In general, an NP-hard problem: Combinatorial optimization problem with $\sum_{i=1}^{s}\binom{D}{i}$ possible locations of nonzeros in $\boldsymbol{w}$
- Also, the constraint $||\boldsymbol{w}||_0 \leq s$ is non-convex (figure on next slide).
- A huge body of work on solving these problems. Primarily in two categories
  - Convex-ify the problem (e.g., replace the ℓ_0 norm by the ℓ_1 norm)

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 \quad \text{s.t. } ||\boldsymbol{w}||_1 \leq t \quad \text{OR} \quad \arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 + \lambda||\boldsymbol{w}||_1$$

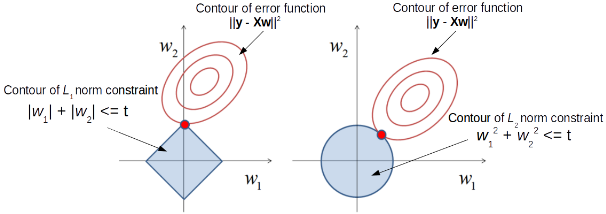
    (note: the ℓ_1 constraint makes the objective non-diff. at 0, but many ways to handle this)
  - Use non-convex optimization methods, e.g., iterative hard threholding (rough idea: use gradient descent to solve for $\boldsymbol{w}$ and set $D - s$ smallest entries to zero in every iteration; basically a projected GD method)

† See "Optimization Methods for ℓ_1 Regularization" by Schmidt *et al* (2009)

Sparsity using ℓ_1 Norm

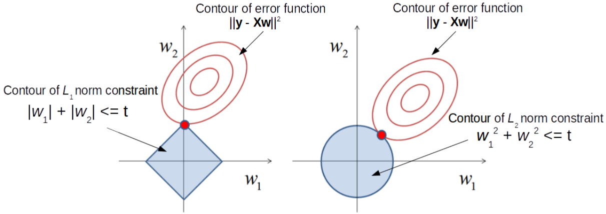
Sparsity using ℓ_1 Norm

- Why ℓ_1 norm gives sparsity? Many explanations. An informal one:

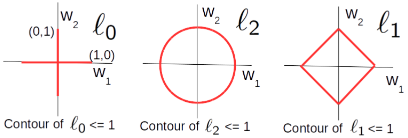


Sparsity using ℓ_1 Norm

- Why ℓ_1 norm gives sparsity? Many explanations. An informal one:



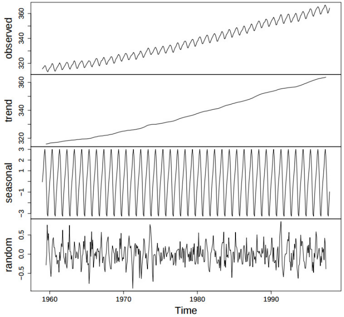
- Chances of the error function contour meeting the constraint contour at the coordinate axes is more likely in case of ℓ_1
- Another explanation: Between ℓ_2 and ℓ_1 norms, ℓ_1 is "closer" to the ℓ_0 norm (in fact, ℓ_1 norm is the closest convex approximation to ℓ_0 norm)



Learning from Time-Series Data

Modeling Time-Series Data

- The input is a sequence of (non-i.i.d.) examples $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T$
- The problem may be supervised or unsupervised, e.g.,
  - Forecasting: Predict $\boldsymbol{y}_{T+1}$, given $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T$
  - Cluster the examples or perform dimensionality reduction
- Evolution of time-series data can be attributed to several factors

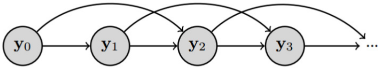


- Teasing apart these factors of variation is also an important problem

Auto-regressive Models

- Auto-regressive (AR): Regress each example on p previous examples

$$\boldsymbol{y}_t = c + \sum_{i=1}^{p} w_i \boldsymbol{y}_{t-i} + \epsilon_t \qquad : \text{ An AR}(p) \text{ model}$$

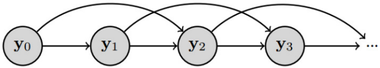


Auto-regressive Model (shown above: 2nd order AR)

Auto-regressive Models

- Auto-regressive (AR): Regress each example on p previous examples

$$\boldsymbol{y}_t = c + \sum_{i=1}^{p} w_i \boldsymbol{y}_{t-i} + \epsilon_t \qquad : \text{ An AR}(p) \text{ model}$$



Auto-regressive Model (shown above: 2nd order AR)

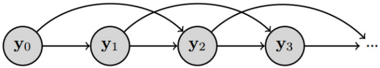
- Moving Average (MA): Regress each example on p previous stochastic errors

$$\boldsymbol{y}_t = c + \epsilon_t + \sum_{i=1}^{p} w_i \epsilon_{t-i} \qquad : \text{ An MA}(p) \text{ model}$$

Auto-regressive Models

- Auto-regressive (AR): Regress each example on p previous examples

$$\boldsymbol{y}_t = c + \sum_{i=1}^{p} w_i \boldsymbol{y}_{t-i} + \epsilon_t \qquad : \text{ An AR}(p) \text{ model}$$



Auto-regressive Model (shown above: 2nd order AR)

- Moving Average (MA): Regress each example on p previous stochastic errors

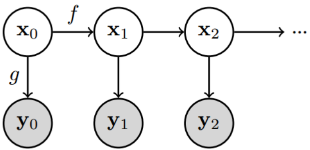
$$\boldsymbol{y}_t = c + \epsilon_t + \sum_{i=1}^{p} w_i \epsilon_{t-i} \qquad : \text{ An MA}(p) \text{ model}$$

- Auto-regressive Moving Average (ARMA): Regress each example of p previous examples and q previous stochastic errors

$$\boldsymbol{y}_t = c + \epsilon_t + \sum_{i=1}^{p} w_i \boldsymbol{y}_{t-i} + \sum_{i=1}^{q} v_i \epsilon_{t-i} \qquad : \text{ An ARMA}(p, q) \text{ model}$$

State-Space Models

- Assume that each observation $\boldsymbol{y}_t$ in the time-series is generated by a low-dimensional latent factor $\boldsymbol{x}_t$ (one-hot or continuous)



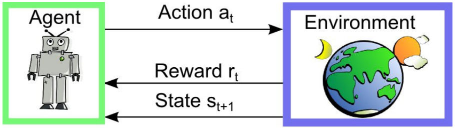
State-Space Model (shown above: 1st order SSM)

- Basically, a generative latent factor model: $\boldsymbol{y}_t = g(\boldsymbol{x}_t)$ and $\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1})$, where g and f are probability distributions
  - Very similar to PPCA/FA, except that latent factor $\boldsymbol{x}_t$ depends on $\boldsymbol{x}_{t-1}$
- Some popular SSMs: Hidden Markov Models (one-hot latent factor $\boldsymbol{x}_t$), Kalman Filters (real-valued latent factor $\boldsymbol{x}_t$)
- Note: Models like RNN/LSTM are also similar, except that these are not generative (but can be made generative)

Reinforcement Learning

Reinforcement Learning

- A paradigm for interactive learning or "learning by doing"

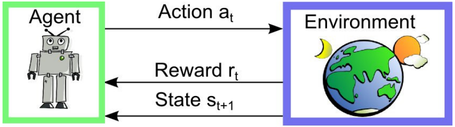


Reinforcement Learning Setup

- Different from supervised learning. Supervision is "implicit"

Reinforcement Learning

- A paradigm for interactive learning or "learning by doing"

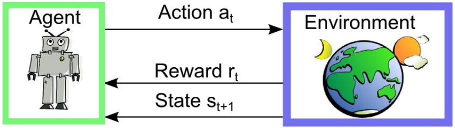


Reinforcement Learning Setup

- Different from supervised learning. Supervision is "implicit"
  - The learner (agent) performs "actions" and gets "rewards"

Reinforcement Learning

- A paradigm for interactive learning or "learning by doing"

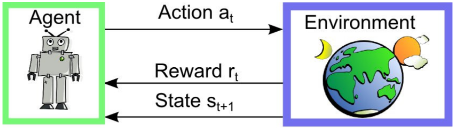


Reinforcement Learning Setup

- Different from supervised learning. Supervision is "implicit"
  - The learner (agent) performs "actions" and gets "rewards"
  - Goal: Learn a policy that maximizes the agent's cumulative expected reward (policy tells what action the agent should take next)

Reinforcement Learning

- A paradigm for interactive learning or "learning by doing"

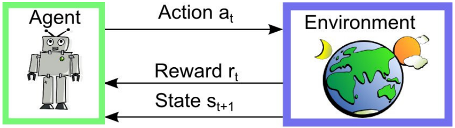


Reinforcement Learning Setup

- Different from supervised learning. Supervision is "implicit"
  - The learner (agent) performs "actions" and gets "rewards"
  - Goal: Learn a policy that maximizes the agent's cumulative expected reward (policy tells what action the agent should take next)
- Order in which data arrives matters (sequential, non i.i.d data)

Reinforcement Learning

- A paradigm for interactive learning or "learning by doing"

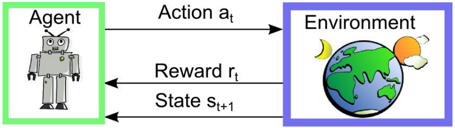


Reinforcement Learning Setup

- Different from supervised learning. Supervision is "implicit"
  - The learner (agent) performs "actions" and gets "rewards"
  - Goal: Learn a policy that maximizes the agent's cumulative expected reward (policy tells what action the agent should take next)
- Order in which data arrives matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives

Reinforcement Learning

- A paradigm for interactive learning or "learning by doing"



Reinforcement Learning Setup

- Different from supervised learning. Supervision is "implicit"
  - The learner (agent) performs "actions" and gets "rewards"
  - Goal: Learn a policy that maximizes the agent's cumulative expected reward (policy tells what action the agent should take next)
- Order in which data arrives matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives
- Many applications: Robotics and control, computer game playing (e.g., Atari, GO), online advertising, financial trading, etc.

Markov Decision Processes

- Markov Decision Process (MDP) gives a way to formulate RL problems

Markov Decision Processes

- Markov Decision Process (MDP) gives a way to formulate RL problems
- MDP formulation assumes the agent knows the its state in the environment

Markov Decision Processes

- Markov Decision Process (MDP) gives a way to formulate RL problems
- MDP formulation assumes the agent knows the its state in the environment
  - Partially Observed MDP (POMDP) can be used when state is unknown

Markov Decision Processes

- Markov Decision Process (MDP) gives a way to formulate RL problems
- MDP formulation assumes the agent knows the its state in the environment
  - Partially Observed MDP (POMDP) can be used when state is unknown
- Assume there are K possible states and a set of actions at any state

Markov Decision Processes

- Markov Decision Process (MDP) gives a way to formulate RL problems
- MDP formulation assumes the agent knows the its state in the environment
  - Partially Observed MDP (POMDP) can be used when state is unknown
- Assume there are K possible states and a set of actions at any state
  - A transition model $p(s_{t+1} = \ell | s_t = k, a_t = a) = P_a(k, \ell)$
    - Each P_a is a $K \times K$ matrix of transition probabilites (one for each action a)

Markov Decision Processes

- Markov Decision Process (MDP) gives a way to formulate RL problems
- MDP formulation assumes the agent knows the its state in the environment
  - Partially Observed MDP (POMDP) can be used when state is unknown
- Assume there are K possible states and a set of actions at any state
  - A transition model $p(s_{t+1} = \ell | s_t = k, a_t = a) = P_a(k, \ell)$
    - Each P_a is a $K \times K$ matrix of transition probabilites (one for each action a)
  - A reward function $R(s_t = k, a_t = a)$

Markov Decision Processes

- Markov Decision Process (MDP) gives a way to formulate RL problems
- MDP formulation assumes the agent knows the its state in the environment
  - Partially Observed MDP (POMDP) can be used when state is unknown
- Assume there are K possible states and a set of actions at any state
  - A transition model $p(s_{t+1} = \ell | s_t = k, a_t = a) = P_a(k, \ell)$
    - Each P_a is a $K \times K$ matrix of transition probabilites (one for each action a)
  - A reward function $R(s_t = k, a_t = a)$
  - Goal: Find a "policy" $\pi(s_t = k)$ which returns the optimal action for $s_t = k$

Markov Decision Processes

- Markov Decision Process (MDP) gives a way to formulate RL problems
- MDP formulation assumes the agent knows the its state in the environment
  - Partially Observed MDP (POMDP) can be used when state is unknown
- Assume there are K possible states and a set of actions at any state
  - A transition model $p(s_{t+1} = \ell | s_t = k, a_t = a) = P_a(k, \ell)$
    - Each P_a is a $K \times K$ matrix of transition probabilites (one for each action a)
  - A reward function $R(s_t = k, a_t = a)$
  - Goal: Find a "policy" $\pi(s_t = k)$ which returns the optimal action for $s_t = k$
  - (P_a, R) and π can be estimated in an alternating fashion

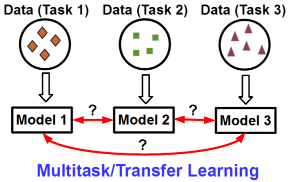
Markov Decision Processes

- Markov Decision Process (MDP) gives a way to formulate RL problems
- MDP formulation assumes the agent knows the its state in the environment
  - Partially Observed MDP (POMDP) can be used when state is unknown
- Assume there are K possible states and a set of actions at any state
  - A transition model $p(s_{t+1} = \ell | s_t = k, a_t = a) = P_a(k, \ell)$
    - Each P_a is a $K \times K$ matrix of transition probabilites (one for each action a)
  - A reward function $R(s_t = k, a_t = a)$
  - Goal: Find a "policy" $\pi(s_t = k)$ which returns the optimal action for $s_t = k$
  - (P_a, R) and π can be estimated in an alternating fashion
  - Estimating P_a and R requires some training data. Can be done even when the state space is continuous (requires solving a function approximation problem)

Multitask/Transfer Learning

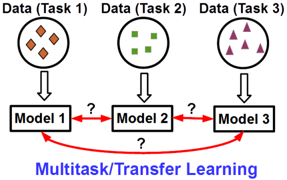
Multitask/Transfer Learning

- An umbrella term to refer to settings when we want to learn multiple models that could potentially benefit from sharing information with each other
- Each learning problem is a "task"



Multitask/Transfer Learning

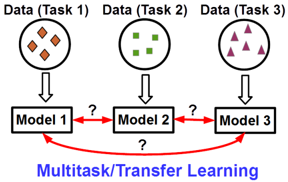
- An umbrella term to refer to settings when we want to learn multiple models that could potentially benefit from sharing information with each other
- Each learning problem is a "task"



- Note: We rarely know how the different tasks are related with each other

Multitask/Transfer Learning

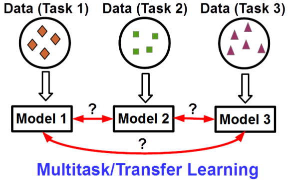
- An umbrella term to refer to settings when we want to learn multiple models that could potentially benefit from sharing information with each other
- Each learning problem is a "task"



- Note: We rarely know how the different tasks are related with each other
  - Have to learn the relatedness structure as well

Multitask/Transfer Learning

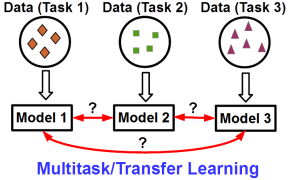
- An umbrella term to refer to settings when we want to learn multiple models that could potentially benefit from sharing information with each other
- Each learning problem is a "task"



- Note: We rarely know how the different tasks are related with each other
  - Have to learn the relatedness structure as well
- For M tasks, we will jointly learn M models, say $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M$

Multitask/Transfer Learning

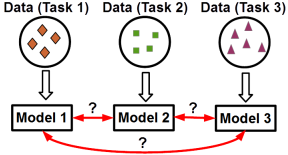
- An umbrella term to refer to settings when we want to learn multiple models that could potentially benefit from sharing information with each other
- Each learning problem is a "task"



- Note: We rarely know how the different tasks are related with each other
  - Have to learn the relatedness structure as well
- For M tasks, we will jointly learn M models, say $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M$
  - Need to jointly regularize the models to make them similar to each other based on their degree/way of relatedness with each other (to be learned)

Multitask/Transfer Learning

- An umbrella term to refer to settings when we want to learn multiple models that could potentially benefit from sharing information with each other
- Each learning problem is a "task"



- Note: We rarely know how the different tasks are related with each other
  - Have to learn the relatedness structure as well
- For M tasks, we will jointly learn M models, say $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M$
  - Need to jointly regularize the models to make them similar to each other based on their degree/way of relatedness with each other (to be learned)
- Some other related problem settings: Domain Adaptation, Covariate Shift

Covariate Shift

- Note: The input or the feature vector $\boldsymbol{x}$ is also known as "covariate"

Covariate Shift

- Note: The input or the feature vector $\boldsymbol{x}$ is also known as "covariate"
- Suppose training inputs come from distribution $p_{tr}(\boldsymbol{x})$

Covariate Shift

- Note: The input or the feature vector $\boldsymbol{x}$ is also known as "covariate"
- Suppose training inputs come from distribution $p_{tr}(\boldsymbol{x})$
- Suppose test inputs come from distribution $p_{tr}(\boldsymbol{x})$ and $p_{tr}(\boldsymbol{x}) \neq p_{te}(\boldsymbol{x})$

Covariate Shift

- Note: The input or the feature vector $\boldsymbol{x}$ is also known as "covariate"
- Suppose training inputs come from distribution $p_{tr}(\boldsymbol{x})$
- Suppose test inputs come from distribution $p_{tr}(\boldsymbol{x})$ and $p_{tr}(\boldsymbol{x}) \neq p_{te}(\boldsymbol{x})$
- The following loss function can be used to handle this issue

$$\sum_{n=1}^{N} \frac{p_{te}(\boldsymbol{x}_n)}{p_{tr}(\boldsymbol{x}_n)} \ell(y_n, f(\boldsymbol{x}_n)) + R(f)$$

Covariate Shift

- Note: The input or the feature vector $\boldsymbol{x}$ is also known as "covariate"
- Suppose training inputs come from distribution $p_{tr}(\boldsymbol{x})$
- Suppose test inputs come from distribution $p_{tr}(\boldsymbol{x})$ and $p_{tr}(\boldsymbol{x}) \neq p_{te}(\boldsymbol{x})$
- The following loss function can be used to handle this issue

$$\sum_{n=1}^{N} \frac{p_{te}(\boldsymbol{x}_n)}{p_{tr}(\boldsymbol{x}_n)} \ell(y_n, f(\boldsymbol{x}_n)) + R(f)$$

- Why will this work? Well, because

$$\mathbb{E}_{(\boldsymbol{x},y)\sim p_{te}}[\ell(y, \boldsymbol{x}, \boldsymbol{w})] = \mathbb{E}_{(\boldsymbol{x},y)\sim p_{tr}}\left[\frac{p_{te}(\boldsymbol{x}, y)}{p_{tr}(\boldsymbol{x}, y)}\ell(y, \boldsymbol{x}, \boldsymbol{w})\right]$$

Covariate Shift

- Note: The input or the feature vector $\boldsymbol{x}$ is also known as "covariate"
- Suppose training inputs come from distribution $p_{tr}(\boldsymbol{x})$
- Suppose test inputs come from distribution $p_{tr}(\boldsymbol{x})$ and $p_{tr}(\boldsymbol{x}) \neq p_{te}(\boldsymbol{x})$
- The following loss function can be used to handle this issue

$$\sum_{n=1}^{N} \frac{p_{te}(\boldsymbol{x}_n)}{p_{tr}(\boldsymbol{x}_n)} \ell(y_n, f(\boldsymbol{x}_n)) + R(f)$$

- Why will this work? Well, because

$$\mathbb{E}_{(\boldsymbol{x},y)\sim p_{te}}[\ell(y, \boldsymbol{x}, \boldsymbol{w})] = \mathbb{E}_{(\boldsymbol{x},y)\sim p_{tr}}\left[\frac{p_{te}(\boldsymbol{x}, y)}{p_{tr}(\boldsymbol{x}, y)} \ell(y, \boldsymbol{x}, \boldsymbol{w})\right]$$

- If $p(y|\boldsymbol{x})$ doesn't change and only $p(\boldsymbol{x})$ changes, then $\frac{p_{te}(\boldsymbol{x},y)}{p_{tr}(\boldsymbol{x},y)} = \frac{p_{te}(\boldsymbol{x})}{p_{tr}(\boldsymbol{x})}$

Covariate Shift

- Note: The input or the feature vector $\boldsymbol{x}$ is also known as "covariate"
- Suppose training inputs come from distribution $p_{tr}(\boldsymbol{x})$
- Suppose test inputs come from distribution $p_{tr}(\boldsymbol{x})$ and $p_{tr}(\boldsymbol{x}) \neq p_{te}(\boldsymbol{x})$
- The following loss function can be used to handle this issue

$$\sum_{n=1}^{N} \frac{p_{te}(\boldsymbol{x}_n)}{p_{tr}(\boldsymbol{x}_n)} \ell(y_n, f(\boldsymbol{x}_n)) + R(f)$$

- Why will this work? Well, because

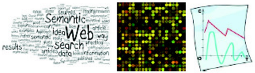
$$\mathbb{E}_{(\boldsymbol{x},y)\sim p_{te}}[\ell(y, \boldsymbol{x}, \boldsymbol{w})] = \mathbb{E}_{(\boldsymbol{x},y)\sim p_{tr}}\left[\frac{p_{te}(\boldsymbol{x}, y)}{p_{tr}(\boldsymbol{x}, y)} \ell(y, \boldsymbol{x}, \boldsymbol{w})\right]$$

- If $p(y|\boldsymbol{x})$ doesn't change and only $p(\boldsymbol{x})$ changes, then $\frac{p_{te}(\boldsymbol{x},y)}{p_{tr}(\boldsymbol{x},y)} = \frac{p_{te}(\boldsymbol{x})}{p_{tr}(\boldsymbol{x})}$
- Can actually estimate the ratio without estimating the densities (a huge body of work on this problem)

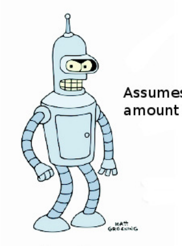
Active Learning

Active Learning

Allow the learner to ask for the most informative training examples



raw unlabeled data
$x_1, x_2, x_3, \ldots$



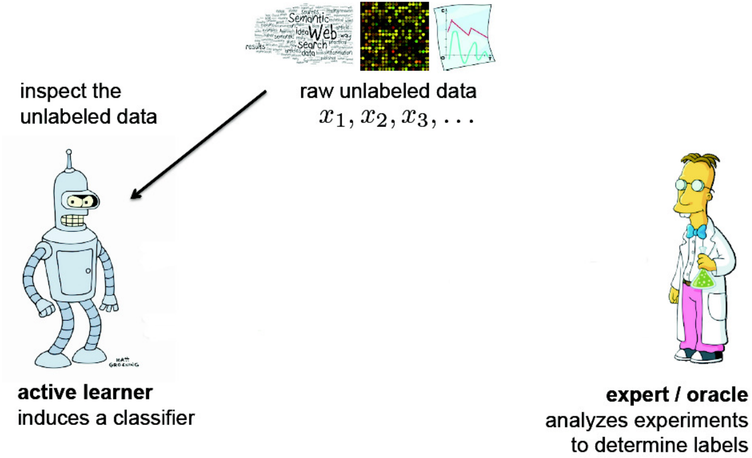
Assumes some small
amount of initial labeled training data

active learner
induces a classifier

expert / oracle
analyzes experiments
to determine labels

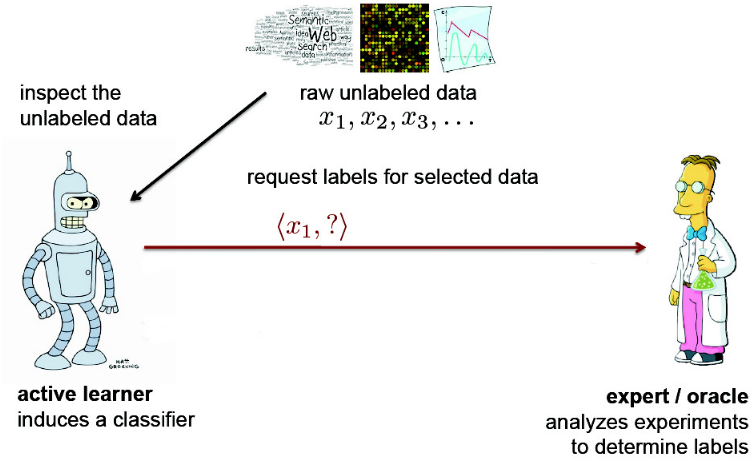
Active Learning

Allow the learner to ask for the most informative training examples



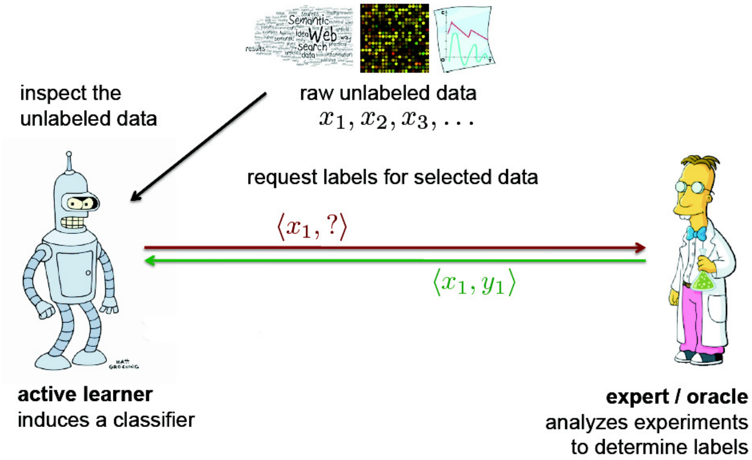
Active Learning

Allow the learner to ask for the most informative training examples



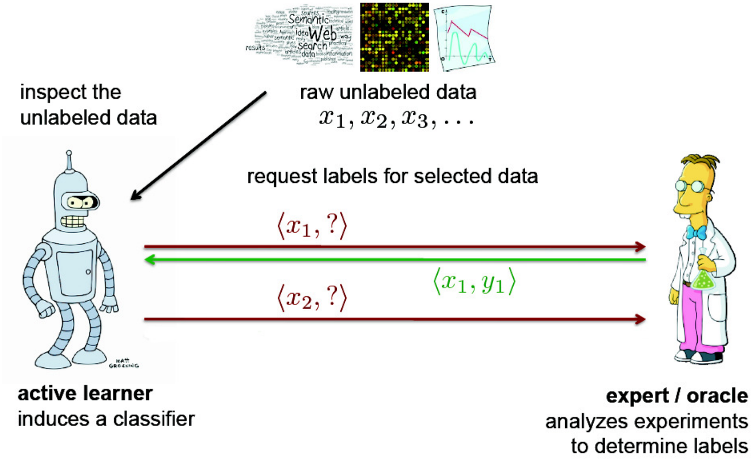
Active Learning

Allow the learner to ask for the most informative training examples



Active Learning

Allow the learner to ask for the most informative training examples



Bayesian Learning

Optimization vs Inference

Learning as Optimization

Learning as (Bayesian) Inference

Optimization vs Inference

Learning as Optimization

- Parameter θ is a fixed unknown

Learning as (Bayesian) Inference

Optimization vs Inference

Learning as Optimization

- Parameter θ is a fixed unknown
- Minimize a "loss" and find a point estimate (best answer) for θ, given data $\mathbf{X}$

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \text{Loss}(\mathbf{X}; \theta)$$

Learning as (Bayesian) Inference

Optimization vs Inference

Learning as Optimization

- Parameter θ is a fixed unknown
- Minimize a "loss" and find a point estimate (best answer) for θ, given data $\mathbf{X}$

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \text{Loss}(\mathbf{X}; \theta) \quad \underline{\text{or}} \quad \underbrace{\arg\max_{\theta} \log p(\mathbf{X}|\theta)}_{\text{Maximum Likelihood}} \quad \underline{\text{or}} \quad \underbrace{\arg\max_{\theta} \log p(\mathbf{X}|\theta)p(\theta)}_{\text{Maximum-a-Posteriori Estimation}}$$

Learning as (Bayesian) Inference

Optimization vs Inference

Learning as Optimization

- Parameter θ is a fixed unknown
- Minimize a "loss" and find a point estimate (best answer) for θ, given data $\mathbf{X}$

$$\hat{\theta} = \arg\min_{\theta\in\Theta} \text{Loss}(\mathbf{X};\theta) \quad \underline{\text{or}} \quad \underbrace{\arg\max_{\theta} \log p(\mathbf{X}|\theta)}_{\text{Maximum Likelihood}} \quad \underline{\text{or}} \quad \underbrace{\arg\max_{\theta} \log p(\mathbf{X}|\theta)p(\theta)}_{\text{Maximum-a-Posteriori Estimation}}$$

Learning as (Bayesian) Inference

- Treat the parameter θ as a random variable with a prior distribution $p(\theta)$

Optimization vs Inference

Learning as Optimization

- Parameter θ is a fixed unknown
- Minimize a "loss" and find a point estimate (best answer) for θ, given data $\mathbf{X}$

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \text{Loss}(\mathbf{X}; \theta) \quad \underline{\text{or}} \quad \underbrace{\arg\max_{\theta} \log p(\mathbf{X}|\theta)}_{\text{Maximum Likelihood}} \quad \underline{\text{or}} \quad \underbrace{\arg\max_{\theta} \log p(\mathbf{X}|\theta)p(\theta)}_{\text{Maximum-a-Posteriori Estimation}}$$

Learning as (Bayesian) Inference

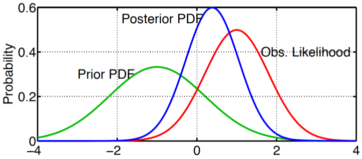
- Treat the parameter θ as a random variable with a prior distribution $p(\theta)$
- Infer a posterior distribution over the parameters using **Bayes rule**

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})} \propto \text{Likelihood} \times \text{Prior}$$

- Posterior becomes the new prior for next batch of observed data
- No "fitting", so no overfitting!

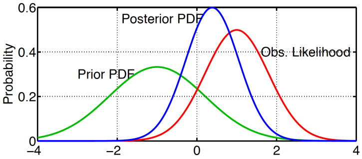
Why be Bayesian?

- Can capture/quantify the uncertainty (or "variance") in θ via the posterior



Why be Bayesian?

- Can capture/quantify the uncertainty (or "variance") in θ via the posterior

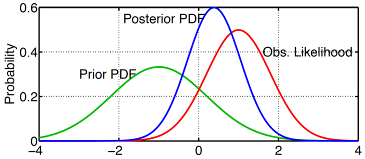


- Can make predictions by averaging over the posterior

$$\underbrace{p(y|x, X, Y)}_{\text{predictive posterior}} = \int p(y|x, \theta) p(\theta|X, Y) d\theta$$

Why be Bayesian?

- Can capture/quantify the uncertainty (or "variance") in θ via the posterior



- Can make predictions by averaging over the posterior

$$\underbrace{p(y|x, X, Y)}_{\text{predictive posterior}} = \int p(y|x, \theta) p(\theta|X, Y) d\theta$$

- Many other benefits (wait for next semester :))

Conclusion and Take-aways

Conclusion and Take-aways

- Most learning problems can be cast as optimizing a regularized loss function

Conclusion and Take-aways

- Most learning problems can be cast as optimizing a regularized loss function
- Probabilistic viewpoint: Most learning problems can be cast as doing MLE/MAP on a probabilistic model of the data
    - Negative log-likelihood (NLL) = loss function, log-prior = regularizer

Conclusion and Take-aways

- Most learning problems can be cast as optimizing a regularized loss function
- Probabilistic viewpoint: Most learning problems can be cast as doing MLE/MAP on a probabilistic model of the data
  - Negative log-likelihood (NLL) = loss function, log-prior = regularizer
- More sophisticated models can be constructed with this basic understanding: Just think of the appropriate loss function/probability model for the data, and the appropriate regularizer/prior

Conclusion and Take-aways

- Most learning problems can be cast as optimizing a regularized loss function
- Probabilistic viewpoint: Most learning problems can be cast as doing MLE/MAP on a probabilistic model of the data
  - Negative log-likelihood (NLL) = loss function, log-prior = regularizer
- More sophisticated models can be constructed with this basic understanding: Just think of the appropriate loss function/probability model for the data, and the appropriate regularizer/prior
- Always start with simple models. Linear models can be really powerful given a good feature representation.

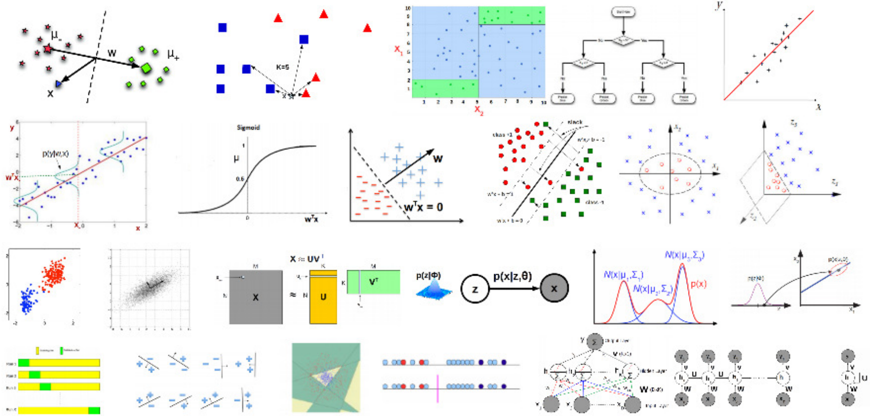
Conclusion and Take-aways

- Most learning problems can be cast as optimizing a regularized loss function
- Probabilistic viewpoint: Most learning problems can be cast as doing MLE/MAP on a probabilistic model of the data
  - Negative log-likelihood (NLL) = loss function, log-prior = regularizer
- More sophisticated models can be constructed with this basic understanding: Just think of the appropriate loss function/probability model for the data, and the appropriate regularizer/prior
- Always start with simple models. Linear models can be really powerful given a good feature representation.
- Learn to first diagnose a learning algorithm rather than trying new ones

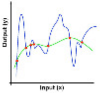
Conclusion and Take-aways

- Most learning problems can be cast as optimizing a regularized loss function
- Probabilistic viewpoint: Most learning problems can be cast as doing MLE/MAP on a probabilistic model of the data
  - Negative log-likelihood (NLL) = loss function, log-prior = regularizer
- More sophisticated models can be constructed with this basic understanding: Just think of the appropriate loss function/probability model for the data, and the appropriate regularizer/prior
- Always start with simple models. Linear models can be really powerful given a good feature representation.
- Learn to first diagnose a learning algorithm rather than trying new ones
- No free lunch. No learning algorithm is "universally" good.

Thank You!

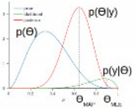


$$\hat{f} = \arg\min_f \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n)) + \lambda R(f)$$



$$p(\theta|\mathbf{y}) = \frac{p(\mathbf{y}|\theta)p(\theta)}{p(\mathbf{y})}$$

$$\hat{\theta}_{MAP} = \arg\max_\theta \sum_{n=1}^{N} \log p(y_n|\theta) + \log p(\theta)$$



And **No Free Lunch**!