Overview of Other Topics, Conclusion and Perspectives

Piyush Rai

Probabilistic Machine Learning (CS772A)

Nov 14, 2017

Announcements

- Project presentations from Nov 20-21
- Duration of each presentation is 12+3 mins
- Will send a Google form to pick your preferred slots

Announcements

- Project presentations from Nov 20-21
- Duration of each presentation is 12+3 mins
- Will send a Google form to pick your preferred slots
- Final report will be due on Nov 26, 11:59pm (guidelines for preparation to be mailed soon)

Announcements

- Project presentations from Nov 20-21
- Duration of each presentation is 12+3 mins
- Will send a Google form to pick your preferred slots
- Final report will be due on Nov 26, 11:59pm (guidelines for preparation to be mailed soon)
- Will hold a final exam review session. Will send a Google form for the preferred date.

Announcements

- Project presentations from Nov 20-21
- Duration of each presentation is 12+3 mins
- Will send a Google form to pick your preferred slots
- Final report will be due on Nov 26, 11:59pm (guidelines for preparation to be mailed soon)
- Will hold a final exam review session. Will send a Google form for the preferred date.
- Important: Final exam venue will be RM-101 (Nov 25, 9am-12pm), not LHC

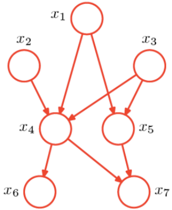
Plan for today

- A quick recap of the previous lecture (graphical models - directed and undirected)
- An overview of other topics
  - Probabilistic models for Active Learning
  - Probabilistic models for Semi-supervised Learning
  - Probabilistic models for Transfer and Multitask Learning
- Conclusion and perspectives

Recap

Directed Graphical Models (DGM)

- Represent the joint distribution of a set of random variables using a directed acyclic graph



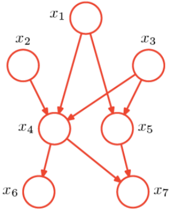
Some nodes may be observed (e.g., data or known hyperparams)

- Joint dist. is a product of the distribution of each node conditioned on its parent nodes

$$p(\boldsymbol{x}) = \prod_{k=1}^{K} p(x_k|\text{pa}_k)$$

Directed Graphical Models (DGM)

- Represent the joint distribution of a set of random variables using a directed acyclic graph



Some nodes may be observed (e.g., data or known hyperparams)

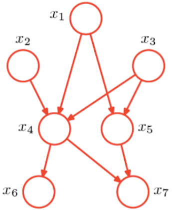
- Joint dist. is a product of the distribution of each node conditioned on its parent nodes

$$p(\boldsymbol{x}) = \prod_{k=1}^{K} p(x_k | \text{pa}_k)$$

- In discrete r.v.'s, each $p(x_k|\text{pa}_k)$ is a table with $|x_k| \prod_{i \in \text{pa}_k} |x_i|$ entries ($|.|$ denotes cardinality)

Directed Graphical Models (DGM)

- Represent the joint distribution of a set of random variables using a directed acyclic graph



Some nodes may be observed (e.g., data or known hyperparams)

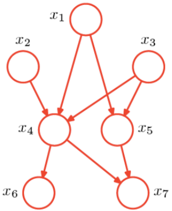
- Joint dist. is a product of the distribution of each node conditioned on its parent nodes

$$p(\boldsymbol{x}) = \prod_{k=1}^{K} p(x_k | \text{pa}_k)$$

- In discrete r.v.'s, each $p(x_k|\text{pa}_k)$ is a table with $|x_k| \prod_{i \in \text{pa}_k} |x_i|$ entries ($|.|$ denotes cardinality)
    - For overall $p(\boldsymbol{x})$, will only need $\sum_{k=1}^{K} |x_k| \prod_{i \in \text{pa}_k} |x_i|$ "numbers" to represent $p(\boldsymbol{x})$

Directed Graphical Models (DGM)

- Represent the joint distribution of a set of random variables using a directed acyclic graph



Some nodes may be observed (e.g., data or known hyperparams)

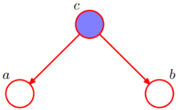
- Joint dist. is a product of the distribution of each node conditioned on its parent nodes

$$p(\boldsymbol{x}) = \prod_{k=1}^{K} p(x_k|\text{pa}_k)$$

- In discrete r.v.'s, each $p(x_k|\text{pa}_k)$ is a table with $|x_k| \prod_{i \in \text{pa}_k} |x_i|$ entries ($|.|$ denotes cardinality)
    - For overall $p(\boldsymbol{x})$, will only need $\sum_{k=1}^{K} |x_k| \prod_{i \in \text{pa}_k} |x_i|$ "numbers" to represent $p(\boldsymbol{x})$
    - On the other hand, a naïve representation of $p(\boldsymbol{x})$ would require $\prod_k |x_k|$ numbers (much larger)

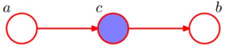
DGM: Tests for Independence

- Some simple tests for conditional independence in a DGM



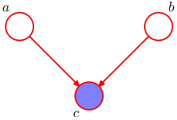
$a \perp\!\!\!\perp b \mid c$

Tail-to-Tail + Observed: Blocks Path



$a \perp\!\!\!\perp b \mid c$

Head-to-Tail + Observed: Blocks Path

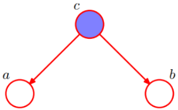


$a \not\perp\!\!\!\perp b \mid c$

Head-to-Head + Observed: Unblocks Path

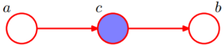
DGM: Tests for Independence

- Some simple tests for conditional independence in a DGM



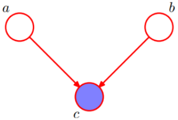
$a \perp\!\!\!\perp b \mid c$

Tail-to-Tail + Observed: Blocks Path



$a \perp\!\!\!\perp b \mid c$

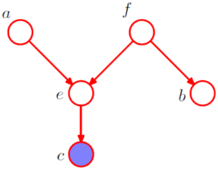
Head-to-Tail + Observed: Blocks Path



$a \not\!\perp\!\!\!\perp b \mid c$

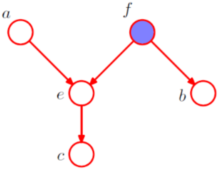
Head-to-Head + Observed: Unblocks Path

- For sophisticated DGM, apply D-separation test (based on a combination of above simple tests)



a to b is unblocked by e

Thus $a \not\!\perp\!\!\!\perp b \mid e$

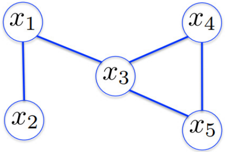


a to b blocked by f

Thus $a \perp\!\!\!\perp b \mid f$

Undirected Graphical Models (UGM) a.k.a. Markov Random Field

- Represent joint distributions as product of non-negative *potentials* $\psi()$ defined over cliques

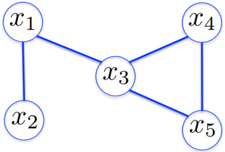


$$p(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$$

where $Z = \sum_{x_1,x_2,x_3,x_4,x_5} \psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$ is a normalizer

Undirected Graphical Models (UGM) a.k.a. Markov Random Field

- Represent joint distributions as product of non-negative *potentials* $\psi()$ defined over cliques



$$p(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$$

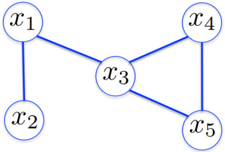
where $Z = \sum_{x_1,x_2,x_3,x_4,x_5} \psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$ is a normalizer

- Can also write each potential using an energy function

$$\psi_c(\boldsymbol{x}_c) = e^{-E(\boldsymbol{x}_c|\theta_c)}$$

Undirected Graphical Models (UGM) a.k.a. Markov Random Field

- Represent joint distributions as product of non-negative *potentials* $\psi()$ defined over cliques



$$p(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$$

where $Z = \sum_{x_1,x_2,x_3,x_4,x_5} \psi(x_1, x_2)\psi(x_1, x_3)\psi(x_3, x_4, x_5)$ is a normalizer

- Can also write each potential using an energy function

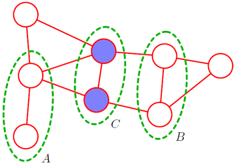
$$\psi_c(\boldsymbol{x}_c) = e^{-E(\boldsymbol{x}_c|\theta_c)}$$

- The joint distribution of a UGM can be then written as

$$p(\boldsymbol{x}) = \frac{1}{Z}\prod_{c\in C}\psi_c(\boldsymbol{x}_c) = \frac{1}{Z}\prod_{c\in C} e^{-E(\boldsymbol{x}_c|\theta_c)} = \frac{1}{Z}e^{-\sum_{c\in C} E(\boldsymbol{x}_c|\theta_c)}$$

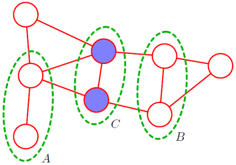
UGM: Tests for Independence

- Usually simpler to check than the DGM case (UGMs have no causal relationships)
- $A \perp\!\!\!\perp B | C$ if all paths from A to B pass through one or more nodes in C



UGM: Tests for Independence

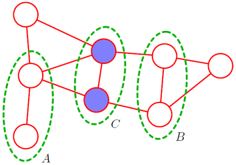
- Usually simpler to check than the DGM case (UGMs have no causal relationships)
- $A \perp\!\!\!\perp B | C$ if all paths from A to B pass through one or more nodes in C



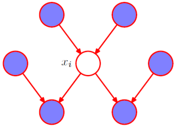
- Another way: See if removing all the nodes (with their edges) in C will "disconnect" A and C

UGM: Tests for Independence

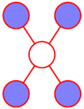
- Usually simpler to check than the DGM case (UGMs have no causal relationships)
- $A \perp\!\!\!\perp B | C$ if all paths from A to B pass through one or more nodes in C



- Another way: See if removing all the nodes (with their edges) in C will "disconnect" A and C
- Also, unlike DGM, Markov blanket of a UGM node only consists only of nodes it is connected to



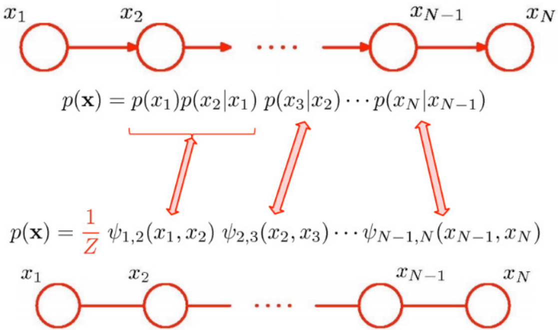
Markov Blanket for a DGM



Markov Blanket for a UGM

DGM and UGM

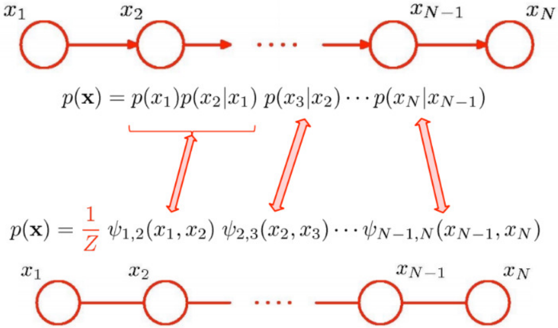
- In general, any probability distribution $p(\boldsymbol{x})$ can be modeled via both a DGM or a UGM
- One form can be transformed into the other. Straightforward for chain-structured DGM



Here Z = 1

DGM and UGM

- In general, any probability distribution $p(\boldsymbol{x})$ can be modeled via both a DGM or a UGM
- One form can be transformed into the other. Straightforward for chain-structured DGM



Here Z = 1

- In general, the conditional distributions of DGM become clique potentials of UGM

DGM and UGM

- In general, any probability distribution $p(\boldsymbol{x})$ can be modeled via both a DGM or a UGM
- One form can be transformed into the other. Straightforward for chain-structured DGM

$$p(\mathbf{x}) = \underbrace{p(x_1)p(x_2|x_1)}\, p(x_3|x_2) \cdots p(x_N|x_{N-1})$$

Here Z = 1

$$p(\mathbf{x}) = \frac{1}{Z}\, \psi_{1,2}(x_1, x_2)\, \psi_{2,3}(x_2, x_3) \cdots \psi_{N-1,N}(x_{N-1}, x_N)$$

- In general, the conditional distributions of DGM become clique potentials of UGM
- Note: For more general DGMs, some more additional care is required in the conversion

Inference via Message-Passing

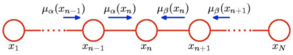
- Some typical DGM/UGM inference tasks: Computing marginals or MAP assignments of nodes
  - Assumption: The structure of the graph is already known

Inference via Message-Passing

- Some typical DGM/UGM inference tasks: Computing marginals or MAP assignments of nodes
  - Assumption: The structure of the graph is already known
- Such tasks can be solved efficiently using message-passing algorithms
  - Basically, an approach to re-order and re-use computations (recursively)

Inference via Message-Passing

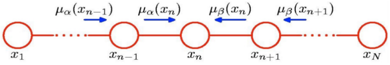
- Some typical DGM/UGM inference tasks: Computing marginals or MAP assignments of nodes
  - Assumption: The structure of the graph is already known
- Such tasks can be solved efficiently using message-passing algorithms
  - Basically, an approach to re-order and re-use computations (recursively)
- E.g., for a chain-structured UGM, computing node marginal $p(x_n) = \sum_{x_1, x_2, x_{n-1}, x_{n+1}, \ldots, x_N} p(\boldsymbol{x})$



$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2)\right]\right] \cdots\right]}_{\mu_\alpha(x_n)}$$

Cost = O(NK^2)

$$\times \underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)\right] \cdots\right]}_{\mu_\beta(x_n)}$$

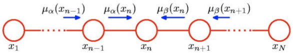


Forward message:
$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots\right] = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \boxed{\mu_\alpha(x_{n-1})}$$

Backward message:
$$\mu_\beta(x_n) = \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \left[\sum_{x_{n+2}} \cdots\right] = \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \boxed{\mu_\beta(x_{n+1})}$$

Inference via Message-Passing

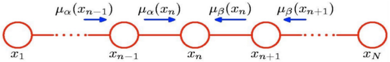
- Some typical DGM/UGM inference tasks: Computing marginals or MAP assignments of nodes
  - Assumption: The structure of the graph is already known
- Such tasks can be solved efficiently using message-passing algorithms
  - Basically, an approach to re-order and re-use computations (recursively)
- E.g., for a chain-structured UGM, computing node marginal $p(x_n) = \sum_{x_1,x_2,x_{n-1},x_{n+1},\ldots,x_N} p(\mathbf{x})$



$$p(x_n) = \frac{1}{Z}\underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1},x_n)\cdots\left[\sum_{x_2}\psi_{2,3}(x_2,x_3)\left[\sum_{x_1}\psi_{1,2}(x_1,x_2)\right]\right]\cdots\right]}_{\mu_\alpha(x_n)}$$

Cost = O(NK²)

$$\times \underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n,x_{n+1})\cdots\left[\sum_{x_N}\psi_{N-1,N}(x_{N-1},x_N)\right]\cdots\right]}_{\mu_\beta(x_n)}$$



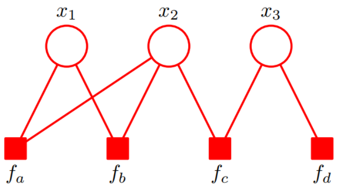
Forward message: $$\mu_\alpha(x_n) = \sum_{x_{n-1}}\psi_{n-1,n}(x_{n-1},x_n)\left[\sum_{x_{n-2}}\cdots\right] = \sum_{x_{n-1}}\psi_{n-1,n}(x_{n-1},x_n)\boxed{\mu_\alpha(x_{n-1})}$$

Backward message: $$\mu_\beta(x_n) = \sum_{x_{n+1}}\psi_{n,n+1}(x_n,x_{n+1})\left[\sum_{x_{n+2}}\cdots\right] = \sum_{x_{n+1}}\psi_{n,n+1}(x_n,x_{n+1})\boxed{\mu_\beta(x_{n+1})}$$

- Significant saving: Assume each node is discrete with K possible values, the cost reduces from $O(K^N)$ to $O(NK^2)$ for computing each node's marginal

Factor Graphs

- A unified representation for general DGM/UGM
- Useful for designing message-passing algos for general DGM/UGM
- A bipartite graph consisting of variable nodes and factor nodes



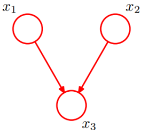
$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s)$$

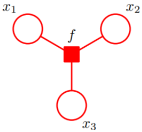
- Basic idea: Original nodes become the variable nodes, conditionals/potentials becomes factor nodes (which represent a computation over the variable nodes connected to the factor node)

Factor Graphs

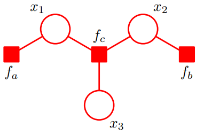
- Both DGM and UGM can be converted into a factor graph representation



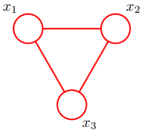
$p(x) = p(x_1)p(x_2)$
$p(x_3|x_1,x_2)$



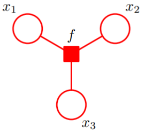
$f(x_1,x_2,x_3) =$
$p(x_1)p(x_2)p(x_3|x_1,x_2)$



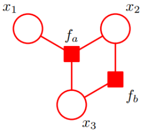
$f_a(x_1) = p(x_1)$
$f_b(x_2) = p(x_2)$
$f_c(x_1,x_2,x_3) = p(x_3|x_2,x_1)$



$\psi(x_1,x_2,x_3)$



$f(x_1,x_2,x_3)$
$= \psi(x_1,x_2,x_3)$



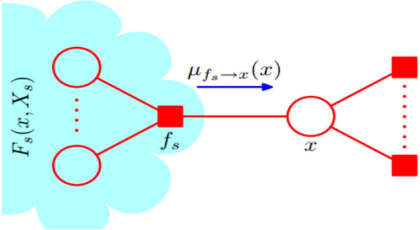
$f_a(x_1,x_2,x_3)f_b(x_2,x_3)$
$= \psi(x_1,x_2,x_3)$

Inference via Message-Passing on Factor Graph: High-Level Idea

- Given the factor graph representation of a distribution $p(\boldsymbol{x})$, can do inference via message passing

Inference via Message-Passing on Factor Graph: High-Level Idea

- Given the factor graph representation of a distribution $p(\boldsymbol{x})$, can do inference via message passing
- Example: Computing marginal $p(x) = \sum_{\boldsymbol{x} \setminus x} p(\boldsymbol{x})$ of some node given the factor graph of $p(\boldsymbol{x})$



$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

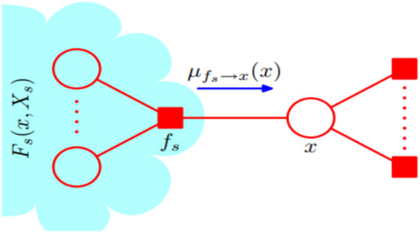
$$p(\mathbf{x}) = \prod_{s \in Ne(x)} F_s(x, X_s)$$

$$p(x) = \prod_{s \in Ne(x)} \left[\sum_{X_s} F_s(x, X_s) \right] = \prod_{s \in Ne(x)} \mu_{f_s \to x}(x)$$

$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s)$$

Inference via Message-Passing on Factor Graph: High-Level Idea

- Given the factor graph representation of a distribution $p(\boldsymbol{x})$, can do inference via message passing
- Example: Computing marginal $p(x) = \sum_{\boldsymbol{x} \setminus x} p(\boldsymbol{x})$ of some node given the factor graph of $p(\boldsymbol{x})$



$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

$$p(\mathbf{x}) = \prod_{s \in Ne(x)} F_s(x, X_s)$$

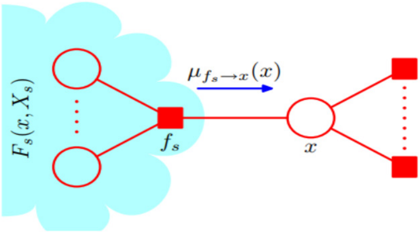
$$p(x) = \prod_{s \in Ne(x)} \left[\sum_{X_s} F_s(x, X_s) \right] = \prod_{s \in Ne(x)} \mu_{f_s \to x}(x)$$

$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s)$$

- This is the sum-product algorithm for marginals with each message $\mu_{f_s \to x}$ recursively defined

Inference via Message-Passing on Factor Graph: High-Level Idea

- Given the factor graph representation of a distribution $p(\boldsymbol{x})$, can do inference via message passing
- Example: Computing marginal $p(x) = \sum_{\boldsymbol{x} \setminus x} p(\boldsymbol{x})$ of some node given the factor graph of $p(\boldsymbol{x})$



$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

$$p(\mathbf{x}) = \prod_{s \in Ne(x)} F_s(x, X_s)$$

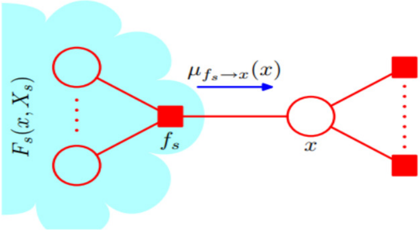
$$p(x) = \prod_{s \in Ne(x)} \left[\sum_{X_s} F_s(x, X_s) \right] = \prod_{s \in Ne(x)} \mu_{f_s \to x}(x)$$

$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s)$$

- This is the sum-product algorithm for marginals with each message $\mu_{f_s \to x}$ recursively defined
- For tree-structured graphs, converges in a finite many steps and gives the exact marginals

Inference via Message-Passing on Factor Graph: High-Level Idea

- Given the factor graph representation of a distribution $p(\boldsymbol{x})$, can do inference via message passing
- Example: Computing marginal $p(x) = \sum_{\boldsymbol{x} \setminus x} p(\boldsymbol{x})$ of some node given the factor graph of $p(\boldsymbol{x})$



$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

$$p(\mathbf{x}) = \prod_{s \in Ne(x)} F_s(x, X_s)$$

$$p(x) = \prod_{s \in Ne(x)} \left[\sum_{X_s} F_s(x, X_s) \right] = \prod_{s \in Ne(x)} \mu_{f_s \to x}(x)$$

$$\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s)$$

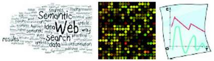
- This is the sum-product algorithm for marginals with each message $\mu_{f_s \to x}$ recursively defined
- For tree-structured graphs, converges in a finite many steps and gives the exact marginals
- PRML Chapter 8 contains more details of such algorithms

An Overview of Some Other Topics

Active Learning

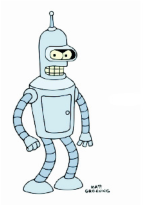
Passive vs Active Learning

Standard supervised learning assumes a "passive" learner



raw unlabeled data

$x_1, x_2, x_3, \ldots$



supervised learner

induces a classifier

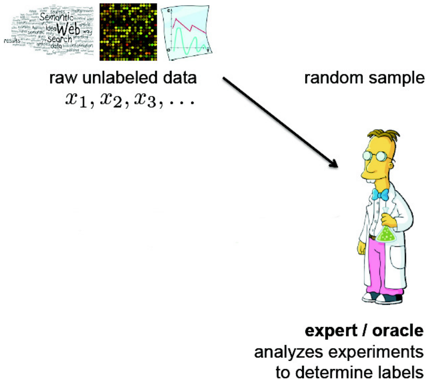
expert / oracle

analyzes experiments

to determine labels

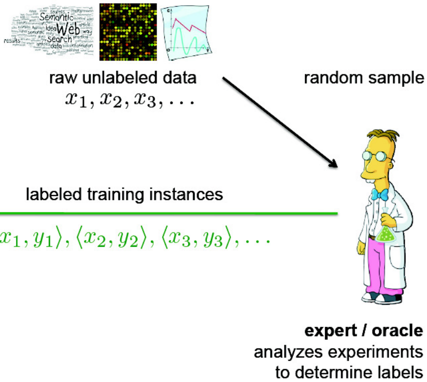
Passive vs Active Learning

Standard supervised learning assumes a "passive" learner



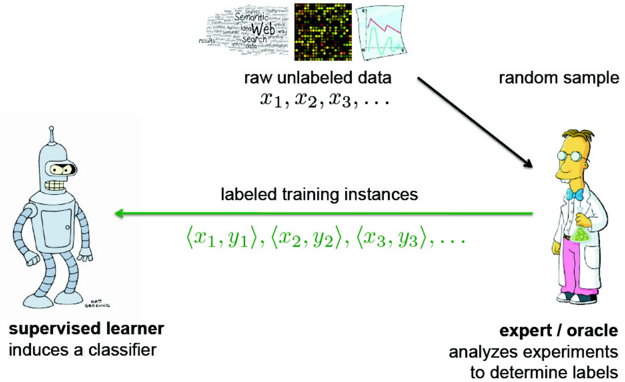
Passive vs Active Learning

Standard supervised learning assumes a "passive" learner



Passive vs Active Learning

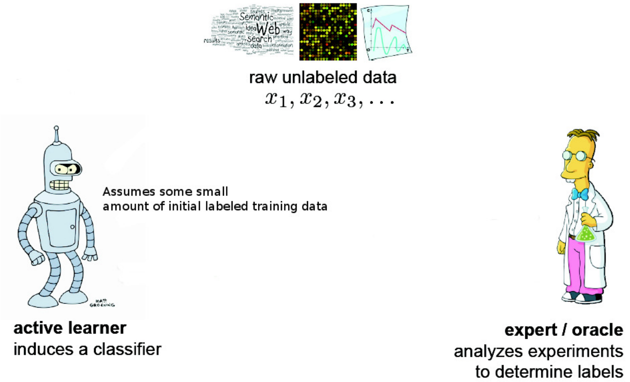
Standard supervised learning assumes a "passive" learner



Called "passive" because the learner doesn't have any control over the labeled data it gets to learn

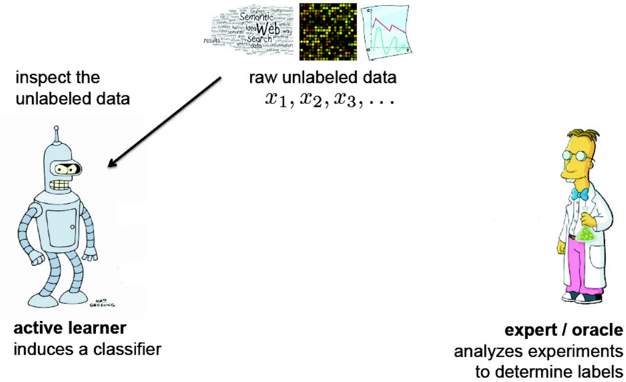
Active Learning

An "active" learner can specifically request labels of "hard" examples that are most useful for learning



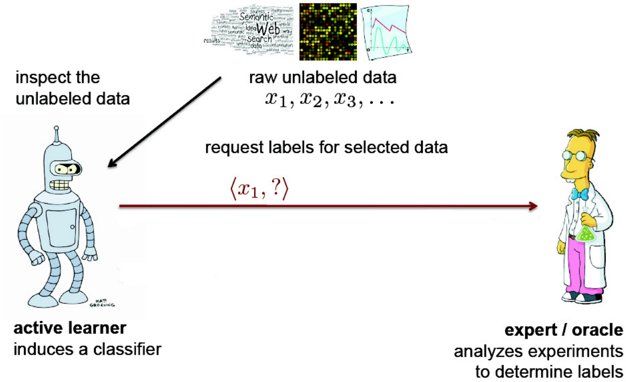
Active Learning

An "active" learner can specifically request labels of "hard" examples that are most useful for learning



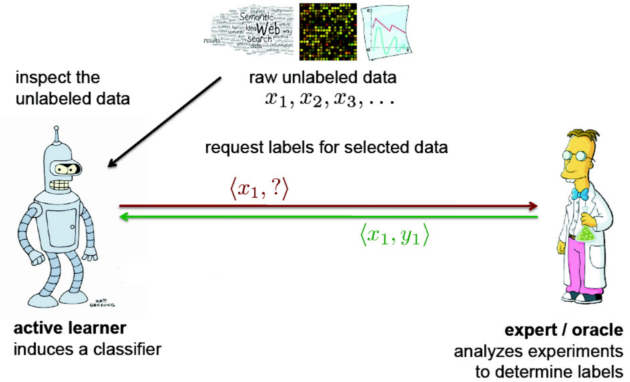
Active Learning

An "active" learner can specifically request labels of "hard" examples that are most useful for learning



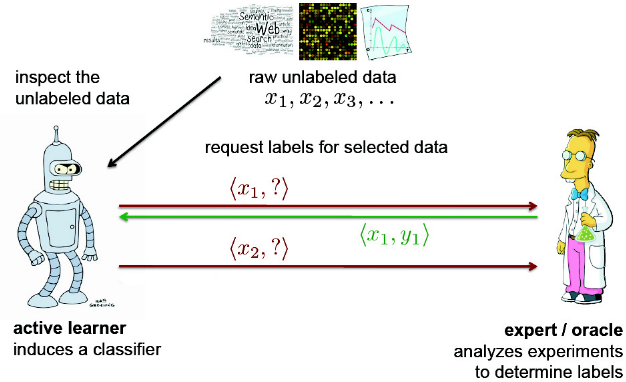
Active Learning

An "active" learner can specifically request labels of "hard" examples that are most useful for learning



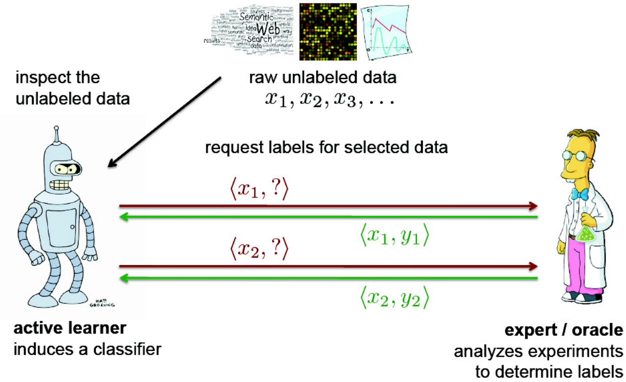
Active Learning

An "active" learner can specifically request labels of "hard" examples that are most useful for learning



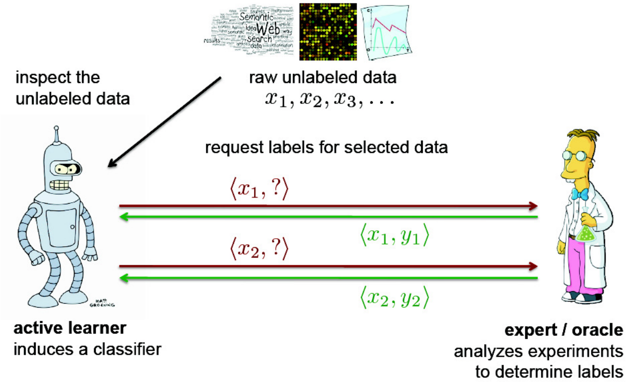
Active Learning

An "active" learner can specifically request labels of "hard" examples that are most useful for learning



Active Learning

An "active" learner can specifically request labels of "hard" examples that are most useful for learning



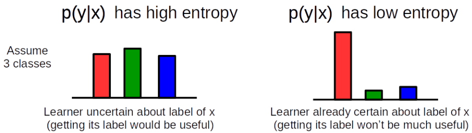
How to quantify the "hardness" or "usefulness" of an unlabeled example?

Active Learning

- Various ways to define the usefulness/hardness of an unlabeled example $\boldsymbol{x}$
- A probabilistic approach can use distribution of y given $\boldsymbol{x}$ to quantify usefulness of $\boldsymbol{x}$

Active Learning

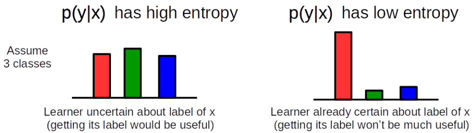
- Various ways to define the usefulness/hardness of an unlabeled example $\boldsymbol{x}$
- A probabilistic approach can use distribution of y given $\boldsymbol{x}$ to quantify usefulness of $\boldsymbol{x}$, e.g.,
  - In classification, can look at the entropy of the distribution $p(y|\boldsymbol{x}, \boldsymbol{w})$ or posterior predictive $p(y|\boldsymbol{x})$



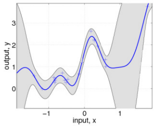
.. and choose example(s) for which the distribution of y given $\boldsymbol{x}$ has the largest entropy

Active Learning

- Various ways to define the usefulness/hardness of an unlabeled example $\boldsymbol{x}$
- A probabilistic approach can use distribution of y given $\boldsymbol{x}$ to quantify usefulness of $\boldsymbol{x}$, e.g.,
  - In classification, can look at the entropy of the distribution $p(y|\boldsymbol{x}, \boldsymbol{w})$ or posterior predictive $p(y|\boldsymbol{x})$



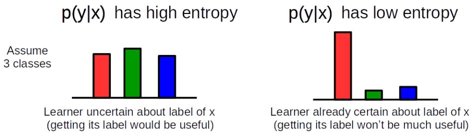
  .. and choose example(s) for which the distribution of y given $\boldsymbol{x}$ has the largest entropy
  - Look at the variance of the posterior predictive $p(y|\boldsymbol{x})$. E.g. for regression,



  .. and choose choose example(s) for which variance of posterior predictive is the largest

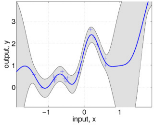
Active Learning

- Various ways to define the usefulness/hardness of an unlabeled example $\boldsymbol{x}$
- A probabilistic approach can use distribution of y given $\boldsymbol{x}$ to quantify usefulness of $\boldsymbol{x}$, e.g.,
  - In classification, can look at the entropy of the distribution $p(y|\boldsymbol{x}, \boldsymbol{w})$ or posterior predictive $p(y|\boldsymbol{x})$



  .. and choose example(s) for which the distribution of y given $\boldsymbol{x}$ has the largest entropy

  - Look at the variance of the posterior predictive $p(y|\boldsymbol{x})$. E.g. for regression,



  .. and choose choose example(s) for which variance of posterior predictive is the largest

  - .. many other ways to utilize the information in $p(y|\boldsymbol{x})$

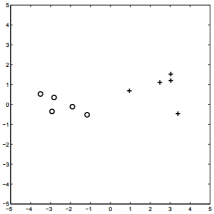
Semi-supervised Learning

Semi-supervised Learning (SSL)

- SSL tries to leverage the structure in unlabeled data to help a supervised learner
- But how might structure of unlabeled data help in supervised learning?

Semi-supervised Learning (SSL)

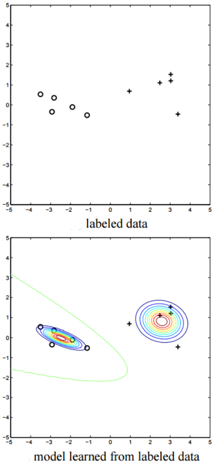
- SSL tries to leverage the structure in unlabeled data to help a supervised learner
- But how might structure of unlabeled data help in supervised learning?



labeled data

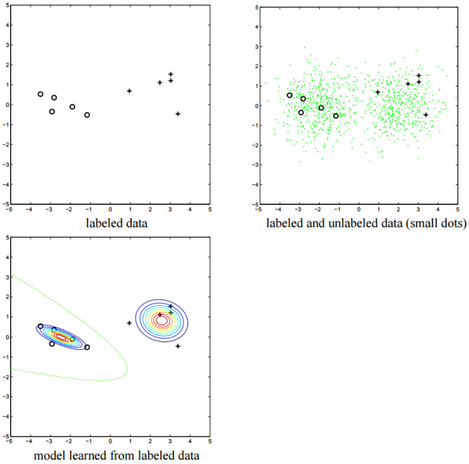
Semi-supervised Learning (SSL)

- SSL tries to leverage the structure in unlabeled data to help a supervised learner
- But how might structure of unlabeled data help in supervised learning?



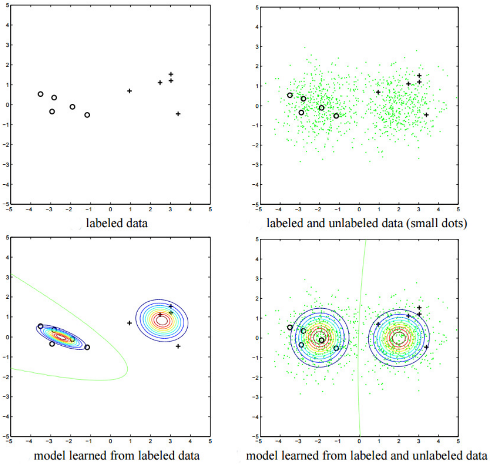
Semi-supervised Learning (SSL)

- SSL tries to leverage the structure in unlabeled data to help a supervised learner
- But how might structure of unlabeled data help in supervised learning?



Semi-supervised Learning (SSL)

- SSL tries to leverage the structure in unlabeled data to help a supervised learner
- But how might structure of unlabeled data help in supervised learning?



A Simple SSL Algo: EM based Generative Classification

- Suppose data $\mathcal{D}$ is labeled $\mathcal{L} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{L}$ and unlabeled $\mathcal{U} = \{\boldsymbol{x}_j\}_{j=L+1}^{L+U}$

A Simple SSL Algo: EM based Generative Classification

- Suppose data $\mathcal{D}$ is labeled $\mathcal{L} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{L}$ and unlabeled $\mathcal{U} = \{\boldsymbol{x}_j\}_{j=L+1}^{L+U}$
- Assume the following model

$$p(\mathcal{D}|\theta) = \underbrace{\prod_{i=1}^{L} p(\boldsymbol{x}_i, y_i|\theta)}_{\text{labeled}} \underbrace{\prod_{j=L+1}^{L+U} p(\boldsymbol{x}_j|\theta)}_{\text{unlabeled}} = \prod_{i=1}^{L} p(\boldsymbol{x}_i, y_i|\theta) \prod_{j=L+1}^{L+U} \sum_{y_j} p(\boldsymbol{x}_j, y_j|\theta)$$

- The unknowns are $\{y_j\}_{j=L+1}^{L+U}$ (latent variables) and θ (parameters)

A Simple SSL Algo: EM based Generative Classification

- Suppose data $\mathcal{D}$ is labeled $\mathcal{L} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{L}$ and unlabeled $\mathcal{U} = \{\boldsymbol{x}_j\}_{j=L+1}^{L+U}$
- Assume the following model

$$p(\mathcal{D}|\theta) = \underbrace{\prod_{i=1}^{L} p(\boldsymbol{x}_i, y_i|\theta)}_{\text{labeled}} \underbrace{\prod_{j=L+1}^{L+U} p(\boldsymbol{x}_j|\theta)}_{\text{unlabeled}} = \prod_{i=1}^{L} p(\boldsymbol{x}_i, y_i|\theta) \prod_{j=L+1}^{L+U} \sum_{y_j} p(\boldsymbol{x}_j, y_j|\theta)$$

- The unknowns are $\{y_j\}_{j=L+1}^{L+U}$ (latent variables) and θ (parameters)
- We can use EM to estimate the unknowns

A Simple SSL Algo: EM based Generative Classification

- Suppose data $\mathcal{D}$ is labeled $\mathcal{L} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{L}$ and unlabeled $\mathcal{U} = \{\boldsymbol{x}_j\}_{j=L+1}^{L+U}$
- Assume the following model

$$p(\mathcal{D}|\theta) = \underbrace{\prod_{i=1}^{L} p(\boldsymbol{x}_i, y_i|\theta)}_{\text{labeled}} \underbrace{\prod_{j=L+1}^{L+U} p(\boldsymbol{x}_j|\theta)}_{\text{unlabeled}} = \prod_{i=1}^{L} p(\boldsymbol{x}_i, y_i|\theta) \prod_{j=L+1}^{L+U} \sum_{y_j} p(\boldsymbol{x}_j, y_j|\theta)$$

- The unknowns are $\{y_j\}_{j=L+1}^{L+U}$ (latent variables) and θ (parameters)
- We can use EM to estimate the unknowns
  - Given $\theta = \hat{\theta}$, E step will compute expected labels for unlabeled examples

$$\mathbb{E}[y_j] = +1 \times P(y_j = +1|\hat{\theta}, \boldsymbol{x}_j) + (-1) \times P(y_j = -1|\hat{\theta}, \boldsymbol{x}_j)$$

A Simple SSL Algo: EM based Generative Classification

- Suppose data $\mathcal{D}$ is labeled $\mathcal{L} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{L}$ and unlabeled $\mathcal{U} = \{\boldsymbol{x}_j\}_{j=L+1}^{L+U}$
- Assume the following model

$$p(\mathcal{D}|\theta) = \underbrace{\prod_{i=1}^{L} p(\boldsymbol{x}_i, y_i|\theta)}_{\text{labeled}} \underbrace{\prod_{j=L+1}^{L+U} p(\boldsymbol{x}_j|\theta)}_{\text{unlabeled}} = \prod_{i=1}^{L} p(\boldsymbol{x}_i, y_i|\theta) \prod_{j=L+1}^{L+U} \sum_{y_j} p(\boldsymbol{x}_j, y_j|\theta)$$

- The unknowns are $\{y_j\}_{j=L+1}^{L+U}$ (latent variables) and θ (parameters)
- We can use EM to estimate the unknowns
    - Given $\theta = \hat{\theta}$, E step will compute expected labels for unlabeled examples

$$\mathbb{E}[y_j] = +1 \times P(y_j = +1|\hat{\theta}, \boldsymbol{x}_j) + (-1) \times P(y_j = -1|\hat{\theta}, \boldsymbol{x}_j)$$

    - M step can then perform standard MLE for re-estimating the parameters θ

A Simple SSL Algo: EM based Generative Classification

- Suppose data $\mathcal{D}$ is labeled $\mathcal{L} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{L}$ and unlabeled $\mathcal{U} = \{\boldsymbol{x}_j\}_{j=L+1}^{L+U}$
- Assume the following model

$$p(\mathcal{D}|\theta) = \underbrace{\prod_{i=1}^{L} p(\boldsymbol{x}_i, y_i|\theta)}_{\text{labeled}} \underbrace{\prod_{j=L+1}^{L+U} p(\boldsymbol{x}_j|\theta)}_{\text{unlabeled}} = \prod_{i=1}^{L} p(\boldsymbol{x}_i, y_i|\theta) \prod_{j=L+1}^{L+U} \sum_{y_j} p(\boldsymbol{x}_j, y_j|\theta)$$

- The unknowns are $\{y_j\}_{j=L+1}^{L+U}$ (latent variables) and θ (parameters)
- We can use EM to estimate the unknowns
  - Given $\theta = \hat{\theta}$, E step will compute expected labels for unlabeled examples

$$\mathbb{E}[y_j] = +1 \times P(y_j = +1|\hat{\theta}, \boldsymbol{x}_j) + (-1) \times P(y_j = -1|\hat{\theta}, \boldsymbol{x}_j)$$

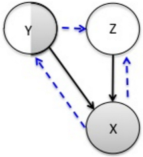
  - M step can then perform standard MLE for re-estimating the parameters θ
- A fairly general framework for semi-supervised learning. Can be used for different types of data (by choosing the appropriate $p(\boldsymbol{x}|y)$ distribution)

SSL: Some Recent Advances

- Some of the recent state-of-the-art classification models are semi-supervised

SSL: Some Recent Advances

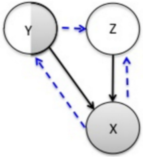
- Some of the recent state-of-the-art classification models are semi-supervised
- Many of these are based on unsupervised deep generative models by adding labels to the model
  - Kingma et al (2014) present a very simple VAE based model for the same



Basically a VAE model with labels of some examples

SSL: Some Recent Advances

- Some of the recent state-of-the-art classification models are semi-supervised
- Many of these are based on unsupervised deep generative models by adding labels to the model
  - Kingma et al (2014) present a very simple VAE based model for the same

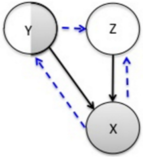


Basically a VAE model with labels of some examples

  - The deep generative model part, by modeling $p(\boldsymbol{x})$, helps learn the structure in unlabeled data

SSL: Some Recent Advances

- Some of the recent state-of-the-art classification models are semi-supervised
- Many of these are based on unsupervised deep generative models by adding labels to the model
    - Kingma et al (2014) present a very simple VAE based model for the same

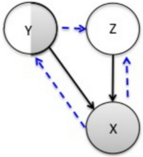


Basically a VAE model with labels of some examples

    - The deep generative model part, by modeling $p(\boldsymbol{x})$, helps learn the structure in unlabeled data
    - A few labels that are available help tune the model for the supervised task at hand

SSL: Some Recent Advances

- Some of the recent state-of-the-art classification models are semi-supervised
- Many of these are based on unsupervised deep generative models by adding labels to the model
  - Kingma et al (2014) present a very simple VAE based model for the same



Basically a VAE model with labels of some examples

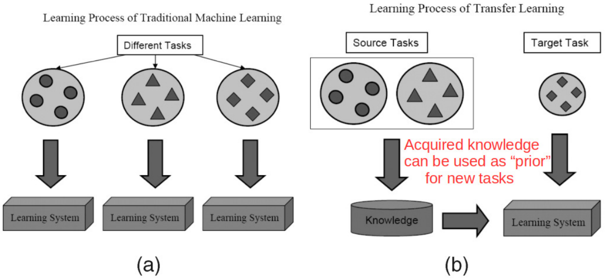
  - The deep generative model part, by modeling $p(\boldsymbol{x})$, helps learn the structure in unlabeled data
  - A few labels that are available help tune the model for the supervised task at hand
- Many other recent advances in SSL using models such as GAN and hybrids of VAE and GAN (too rapid progress to cover in a course like this; look at follow-up work of Kingman et al (2014))

Semi-Supervised Learning with Deep Generative Models (Kingma et al, 2014)

Transfer and Multi-task Learning

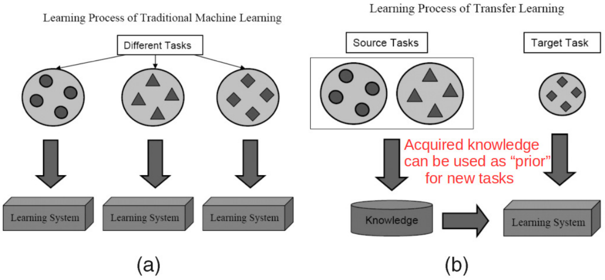
Transfer Learning

- How do we transfer knowledge of what we already know about previous tasks to a new task?



Transfer Learning

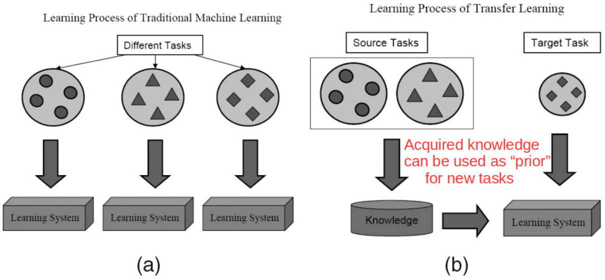
- How do we transfer knowledge of what we already know about previous tasks to a new task?



- A very common and popular practice in deep learning (pre-trained networks)

Transfer Learning

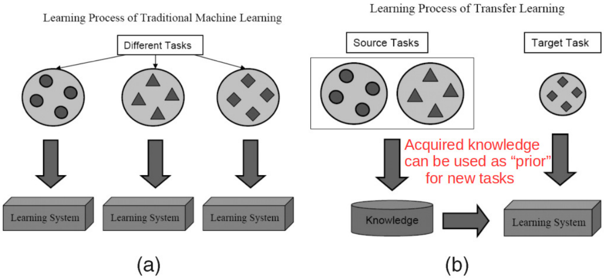
- How do we transfer knowledge of what we already know about previous tasks to a new task?



- A very common and popular practice in deep learning (pre-trained networks)
  - However transfer learning (Caruana, 1997) is more general, not limited to pre-training in deep learning

Transfer Learning

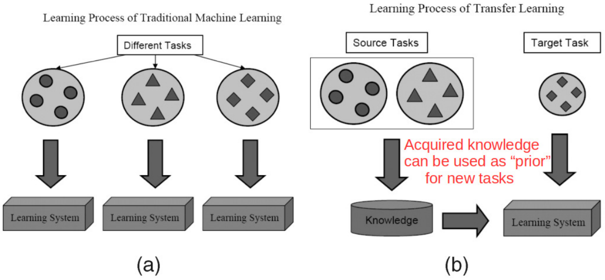
- How do we transfer knowledge of what we already know about previous tasks to a new task?



- A very common and popular practice in deep learning (pre-trained networks)
  - However transfer learning (Caruana, 1997) is more general, not limited to pre-training in deep learning
- Probabilistic modeling gives a natural way to transfer knowledge by using past knowledge as prior

Transfer Learning

- How do we transfer knowledge of what we already know about previous tasks to a new task?

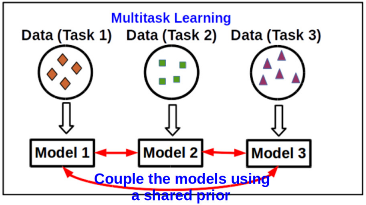


- A very common and popular practice in deep learning (pre-trained networks)
  - However transfer learning (Caruana, 1997) is more general, not limited to pre-training in deep learning
- Probabilistic modeling gives a natural way to transfer knowledge by using past knowledge as prior
  - Train a model for some task, use the learned parameters as hyperparams of prior of the new task

[Multitask Learning (Caruana, 1997)]

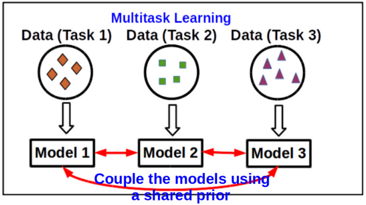
Multitask Learning

- Often we may wish to not learn sequentially (learn one task, then the other) but everything jointly

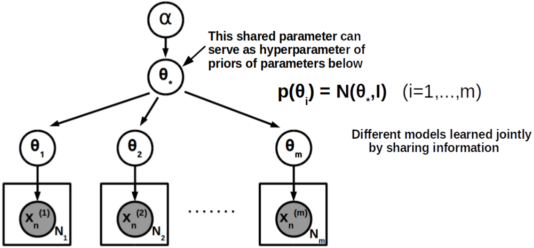


Multitask Learning

- Often we may wish to not learn sequentially (learn one task, then the other) but everything jointly



- A probabilistic model is natural to use in such problems



Conclusion

- Probabilistic modeling provides a natural way to think about models of data

Conclusion

- Probabilistic modeling provides a natural way to think about models of data
- Many benefits as compared to non-probabilistic approaches

Conclusion

- Probabilistic modeling provides a natural way to think about models of data
- Many benefits as compared to non-probabilistic approaches
  - Easier to model and leverage uncertainty in data/parameters

Conclusion

- Probabilistic modeling provides a natural way to think about models of data
- Many benefits as compared to non-probabilistic approaches
  - Easier to model and leverage uncertainty in data/parameters
  - Easier to encode prior knowledge about the problem (via prior/likelihood distributions)

Conclusion

- Probabilistic modeling provides a natural way to think about models of data
- Many benefits as compared to non-probabilistic approaches
  - Easier to model and leverage uncertainty in data/parameters
  - Easier to encode prior knowledge about the problem (via prior/likelihood distributions)
  - Easier to handle missing data (by marginalizing it out if possible, or by treating as latent variable)

Conclusion

- Probabilistic modeling provides a natural way to think about models of data
- Many benefits as compared to non-probabilistic approaches
  - Easier to model and leverage uncertainty in data/parameters
  - Easier to encode prior knowledge about the problem (via prior/likelihood distributions)
  - Easier to handle missing data (by marginalizing it out if possible, or by treating as latent variable)
  - Easier to build complex models can be neatly combining/extending simpler probabilistic models

Conclusion

- Probabilistic modeling provides a natural way to think about models of data
- Many benefits as compared to non-probabilistic approaches
  - Easier to model and leverage uncertainty in data/parameters
  - Easier to encode prior knowledge about the problem (via prior/likelihood distributions)
  - Easier to handle missing data (by marginalizing it out if possible, or by treating as latent variable)
  - Easier to build complex models can be neatly combining/extending simpler probabilistic models
  - Easier to learn the “right model” (hyperparameter estimation, nonparametric Bayesian models)

Conclusion

- Probabilistic modeling provides a natural way to think about models of data
- Many benefits as compared to non-probabilistic approaches
  - Easier to model and leverage uncertainty in data/parameters
  - Easier to encode prior knowledge about the problem (via prior/likelihood distributions)
  - Easier to handle missing data (by marginalizing it out if possible, or by treating as latent variable)
  - Easier to build complex models can be neatly combining/extending simpler probabilistic models
  - Easier to learn the “right model” (hyperparameter estimation, nonparametric Bayesian models)
  - .. and various other benefits as we saw during this course

Conclusion

- Probabilistic modeling provides a natural way to think about models of data
- Many benefits as compared to non-probabilistic approaches
  - Easier to model and leverage uncertainty in data/parameters
  - Easier to encode prior knowledge about the problem (via prior/likelihood distributions)
  - Easier to handle missing data (by marginalizing it out if possible, or by treating as latent variable)
  - Easier to build complex models can be neatly combining/extending simpler probabilistic models
  - Easier to learn the “right model” (hyperparameter estimation, nonparametric Bayesian models)
  - .. and various other benefits as we saw during this course

Conclusion

- Probabilistic modeling provides a natural way to think about models of data
- Many benefits as compared to non-probabilistic approaches
  - Easier to model and leverage uncertainty in data/parameters
  - Easier to encode prior knowledge about the problem (via prior/likelihood distributions)
  - Easier to handle missing data (by marginalizing it out if possible, or by treating as latent variable)
  - Easier to build complex models can be neatly combining/extending simpler probabilistic models
  - Easier to learn the "right model" (hyperparameter estimation, nonparametric Bayesian models)
  - .. and various other benefits as we saw during this course
- Fast-moving field, lots of recent advances on new models and inference methods
  - The course is an attempt to guide you into exploring the area further

We won't be here without..

Thomas Bayes

Pierre-Simon Laplace

Thank You!