Graphical Models and Kernel Methods

Jerry Zhu

Department of Computer Sciences
University of Wisconsin–Madison, USA

MLSS
June 17, 2014

Outline

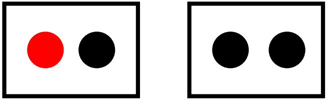
Outline

Graphical Models
- Probabilistic Inference
- Directed vs. Undirected Graphical Models
- Inference
- Parameter Estimation

Kernel Methods
- Support Vector Machines
- Kernel PCA
- Reproducing Kernel Hilbert Spaces

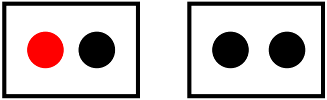
Outline

The envelope quiz



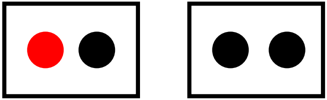
- red ball $=$ \$\$\$

The envelope quiz



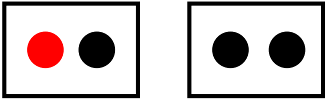
- red ball = \$\$\$
- You randomly picked an envelope, randomly took out a ball – and it was black

The envelope quiz



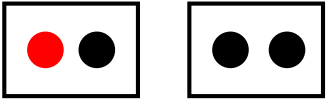
- red ball = \$\$\$
- You randomly picked an envelope, randomly took out a ball – and it was black
- Should you choose this envelope or the other envelope?

The envelope quiz



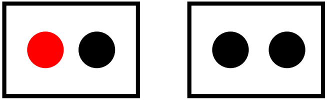
- Probabilistic inference

The envelope quiz



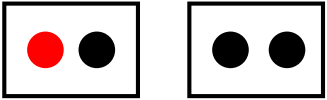
- Probabilistic inference
  - Joint distribution on $E \in \{1, 0\}, B \in \{r, b\}$:
    $P(E, B) = P(E)P(B \mid E)$

The envelope quiz



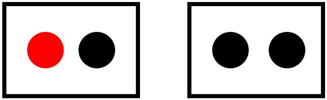
- Probabilistic inference
  - Joint distribution on $E \in \{1, 0\}, B \in \{r, b\}$:
    $P(E, B) = P(E)P(B \mid E)$
  - $P(E = 1) = P(E = 0) = 1/2$

The envelope quiz

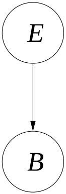


- Probabilistic inference
  - Joint distribution on $E \in \{1, 0\}, B \in \{r, b\}$:
    $P(E, B) = P(E)P(B \mid E)$
  - $P(E = 1) = P(E = 0) = 1/2$
  - $P(B = r \mid E = 1) = 1/2, P(B = r \mid E = 0) = 0$

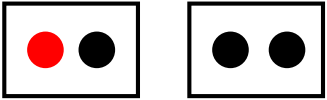
The envelope quiz



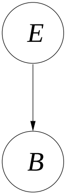
- Probabilistic inference
  - Joint distribution on $E \in \{1, 0\}, B \in \{r, b\}$:
    $P(E, B) = P(E)P(B \mid E)$
  - $P(E = 1) = P(E = 0) = 1/2$
  - $P(B = r \mid E = 1) = 1/2, P(B = r \mid E = 0) = 0$
  - The graphical model:



The envelope quiz

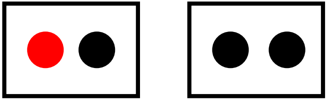


- Probabilistic inference
  - Joint distribution on $E \in \{1, 0\}, B \in \{r, b\}$:
    $P(E, B) = P(E)P(B \mid E)$
  - $P(E = 1) = P(E = 0) = 1/2$
  - $P(B = r \mid E = 1) = 1/2, P(B = r \mid E = 0) = 0$
  - The graphical model:

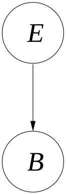


- Statistical decision theory: switch if $P(E = 1 \mid B = b) < 1/2$

The envelope quiz



- Probabilistic inference
  - Joint distribution on $E \in \{1, 0\}, B \in \{r, b\}$: $P(E, B) = P(E)P(B \mid E)$
  - $P(E = 1) = P(E = 0) = 1/2$
  - $P(B = r \mid E = 1) = 1/2, P(B = r \mid E = 0) = 0$
  - The graphical model:



- Statistical decision theory: switch if $P(E = 1 \mid B = b) < 1/2$
- $P(E = 1 \mid B = b) = \frac{P(B=b|E=1)P(E=1)}{P(B=b)} = \frac{1/2 \times 1/2}{3/4} = 1/3$. Switch.

Reasoning with uncertainty

- The world is reduced to a set of random variables $x_1, \ldots, x_d$

Reasoning with uncertainty

- The world is reduced to a set of random variables $x_1, \ldots, x_d$
  - e.g. $(x_1, \ldots, x_{d-1})$ a feature vector, $x_d \equiv y$ the class label

Reasoning with uncertainty

- The world is reduced to a set of random variables $x_1, \ldots, x_d$
  - e.g. $(x_1, \ldots, x_{d-1})$ a feature vector, $x_d \equiv y$ the class label
- Inference: given joint distribution $p(x_1, \ldots, x_d)$, compute $p(X_Q \mid X_E)$ where $X_Q \cup X_E \subseteq \{x_1 \ldots x_d\}$

Reasoning with uncertainty

- The world is reduced to a set of random variables $x_1, \ldots, x_d$
  - e.g. $(x_1, \ldots, x_{d-1})$ a feature vector, $x_d \equiv y$ the class label
- Inference: given joint distribution $p(x_1, \ldots, x_d)$, compute $p(X_Q \mid X_E)$ where $X_Q \cup X_E \subseteq \{x_1 \ldots x_d\}$
  - e.g. $Q = \{d\}$, $E = \{1 \ldots d-1\}$, by the definition of conditional

$$p(x_d \mid x_1, \ldots, x_{d-1}) = \frac{p(x_1, \ldots, x_{d-1}, x_d)}{\sum_v p(x_1, \ldots, x_{d-1}, x_d = v)}$$

Reasoning with uncertainty

- The world is reduced to a set of random variables $x_1, \ldots, x_d$
  - e.g. $(x_1, \ldots, x_{d-1})$ a feature vector, $x_d \equiv y$ the class label
- Inference: given joint distribution $p(x_1, \ldots, x_d)$, compute $p(X_Q \mid X_E)$ where $X_Q \cup X_E \subseteq \{x_1 \ldots x_d\}$
  - e.g. $Q = \{d\}$, $E = \{1 \ldots d-1\}$, by the definition of conditional

$$p(x_d \mid x_1, \ldots, x_{d-1}) = \frac{p(x_1, \ldots, x_{d-1}, x_d)}{\sum_v p(x_1, \ldots, x_{d-1}, x_d = v)}$$

- Learning: estimate $p(x_1, \ldots, x_d)$ from training data $X^{(1)}, \ldots, X^{(N)}$, where $X^{(i)} = (x_1^{(i)}, \ldots, x_d^{(i)})$

It is difficult to reason with uncertainty

- joint distribution $p(x_1, \ldots, x_d)$

It is difficult to reason with uncertainty

- joint distribution $p(x_1, \ldots, x_d)$
  - exponential naïve storage (2^d for binary r.v.)

It is difficult to reason with uncertainty

- joint distribution $p(x_1, \dots, x_d)$
  - exponential naïve storage (2^d for binary r.v.)
  - hard to interpret (conditional independence)

It is difficult to reason with uncertainty

- joint distribution $p(x_1, \ldots, x_d)$
  - exponential naïve storage (2^d for binary r.v.)
  - hard to interpret (conditional independence)
- inference $p(X_Q \mid X_E)$

It is difficult to reason with uncertainty

- joint distribution $p(x_1, \ldots, x_d)$
  - exponential naïve storage (2^d for binary r.v.)
  - hard to interpret (conditional independence)
- inference $p(X_Q \mid X_E)$
  - Often can't afford to do it by brute force

It is difficult to reason with uncertainty

- joint distribution $p(x_1, \ldots, x_d)$
  - exponential naïve storage (2^d for binary r.v.)
  - hard to interpret (conditional independence)
- inference $p(X_Q \mid X_E)$
  - Often can't afford to do it by brute force
- If $p(x_1, \ldots, x_d)$ not given, estimate it from data

It is difficult to reason with uncertainty

- joint distribution $p(x_1, \ldots, x_d)$
  - exponential naïve storage (2^d for binary r.v.)
  - hard to interpret (conditional independence)
- inference $p(X_Q \mid X_E)$
  - Often can't afford to do it by brute force
- If $p(x_1, \ldots, x_d)$ not given, estimate it from data
  - Often can't afford to do it by brute force

It is difficult to reason with uncertainty

- joint distribution $p(x_1, \ldots, x_d)$
  - exponential naïve storage (2^d for binary r.v.)
  - hard to interpret (conditional independence)
- inference $p(X_Q \mid X_E)$
  - Often can't afford to do it by brute force
- If $p(x_1, \ldots, x_d)$ not given, estimate it from data
  - Often can't afford to do it by brute force
- Graphical model: efficient representation, inference, and learning on $p(x_1, \ldots, x_d)$, exactly or approximately

What are graphical models?

- Graphical model = joint distribution $p(x_1, \ldots, x_d)$

What are graphical models?

- Graphical model = joint distribution $p(x_1, \ldots, x_d)$
  - Bayesian network or Markov random field

What are graphical models?

- Graphical model = joint distribution $p(x_1, \ldots, x_d)$
  - Bayesian network or Markov random field
  - conditional independence

What are graphical models?

- Graphical model = joint distribution $p(x_1, \dots, x_d)$
  - Bayesian network or Markov random field
  - conditional independence
- Inference = $p(X_Q \mid X_E)$, in general $X_Q \cup X_E \subset \{x_1 \dots x_d\}$

What are graphical models?

- Graphical model = joint distribution $p(x_1, \ldots, x_d)$
  - Bayesian network or Markov random field
  - conditional independence
- Inference = $p(X_Q \mid X_E)$, in general $X_Q \cup X_E \subset \{x_1 \ldots x_d\}$
  - exact, MCMC, variational

What are graphical models?

- Graphical model = joint distribution $p(x_1, \ldots, x_d)$
  - Bayesian network or Markov random field
  - conditional independence
- Inference = $p(X_Q \mid X_E)$, in general $X_Q \cup X_E \subset \{x_1 \ldots x_d\}$
  - exact, MCMC, variational
- If $p(x_1, \ldots, x_d)$ not given, estimate it from data

What are graphical models?

- Graphical model = joint distribution $p(x_1, \ldots, x_d)$
  - Bayesian network or Markov random field
  - conditional independence
- Inference = $p(X_Q \mid X_E)$, in general $X_Q \cup X_E \subset \{x_1 \ldots x_d\}$
  - exact, MCMC, variational
- If $p(x_1, \ldots, x_d)$ not given, estimate it from data
  - parameter and structure learning

Graphical-Model-Nots

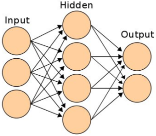
- Graphical model is the study of *probabilistic models*

Graphical-Model-Nots

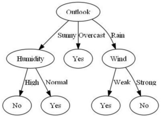
- Graphical model is the study of *probabilistic models*
- Just because there are nodes and edges doesn't mean it's a graphical model

Graphical-Model-Nots

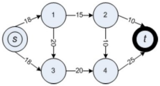
- Graphical model is the study of *probabilistic models*
- Just because there are nodes and edges doesn't mean it's a graphical model
- These are not graphical models:



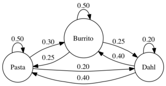
neural network



decision tree



network flow



HMM template
(but HMMs are!)

Outline

Directed graphical models

Directed graphical models

- Also called Bayesian networks

Directed graphical models

- Also called Bayesian networks
- A directed graph has nodes $x_1, \ldots, x_d$, some of them connected by directed edges $x_i \to x_j$

Directed graphical models

- Also called Bayesian networks
- A directed graph has nodes $x_1, \ldots, x_d$, some of them connected by directed edges $x_i \to x_j$
- A cycle is a directed path $x_1 \to \ldots \to x_k$ where $x_1 = x_k$

Directed graphical models

- Also called Bayesian networks
- A directed graph has nodes $x_1, \ldots, x_d$, some of them connected by directed edges $x_i \to x_j$
- A cycle is a directed path $x_1 \to \ldots \to x_k$ where $x_1 = x_k$
- A directed acyclic graph (DAG) contains no cycles

Directed graphical models

- A Bayesian network on the DAG is a family of distributions satisfying

$$\{p \mid p(x_1, \ldots, x_d) = \prod_i p(x_i \mid Pa(x_i))\}$$

where $Pa(x_i)$ is the set of parents of x_i.

Directed graphical models

- A Bayesian network on the DAG is a family of distributions satisfying

$$\{p \mid p(x_1, \ldots, x_d) = \prod_i p(x_i \mid Pa(x_i))\}$$

  where $Pa(x_i)$ is the set of parents of x_i.
- $p(x_i \mid Pa(x_i))$ is the conditional probability distribution (CPD) at x_i

Directed graphical models

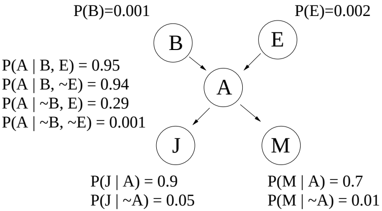
- A Bayesian network on the DAG is a family of distributions satisfying

$$\{p \mid p(x_1, \ldots, x_d) = \prod_i p(x_i \mid Pa(x_i))\}$$

  where $Pa(x_i)$ is the set of parents of x_i.
- $p(x_i \mid Pa(x_i))$ is the conditional probability distribution (CPD) at x_i
- By specifying the CPDs for all i, we specify a joint distribution $p(x_1, \ldots, x_d)$

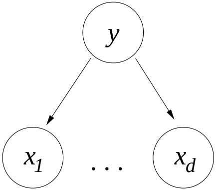
Example: Burglary, Earthquake, Alarm, John and Marry

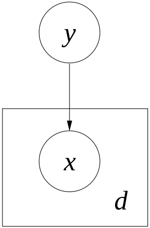
Binary variables



$$
\begin{aligned}
& P(B, \sim E, A, J, \sim M) \\
= \; & P(B)P(\sim E)P(A \mid B, \sim E)P(J \mid A)P(\sim M \mid A) \\
= \; & 0.001 \times (1 - 0.002) \times 0.94 \times 0.9 \times (1 - 0.7) \\
\approx \; & .000253
\end{aligned}
$$

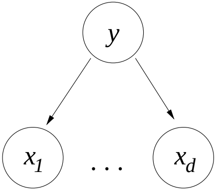
Example: Naive Bayes

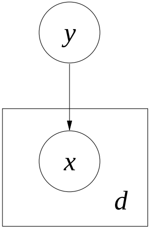




- $p(y, x_1, \ldots x_d) = p(y) \prod_{i=1}^{d} p(x_i \mid y)$

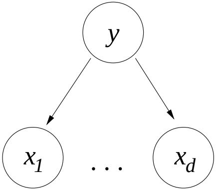
Example: Naive Bayes

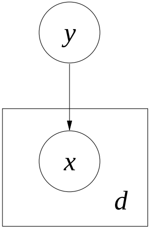




- $p(y, x_1, \ldots x_d) = p(y) \prod_{i=1}^{d} p(x_i \mid y)$
- Plate representation on the right

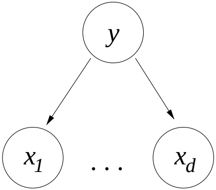
Example: Naive Bayes

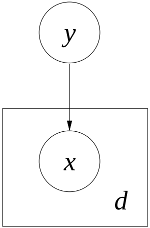




- $p(y, x_1, \ldots x_d) = p(y) \prod_{i=1}^{d} p(x_i \mid y)$
- Plate representation on the right
- $p(y)$ multinomial

Example: Naive Bayes

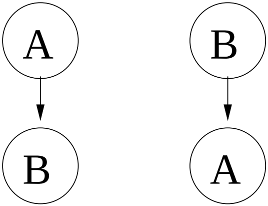




- $p(y, x_1, \ldots x_d) = p(y) \prod_{i=1}^{d} p(x_i \mid y)$
- Plate representation on the right
- $p(y)$ multinomial
- $p(x_i \mid y)$ depends on the feature type: multinomial (count x_i), Gaussian (continuous x_i), etc.

No Causality Whatsoever

$P(A)=a$
$P(B|A)=b$
$P(B|\sim A)=c$



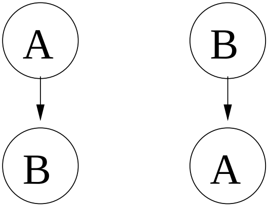
$P(B)=ab+(1-a)c$
$P(A|B)=ab/(ab+(1-a)c)$
$P(A|\sim B)=a(1-b)/(1-ab-(1-a)c)$

The two BNs are equivalent in all respects

- Do not *read* causality from Bayesian networks

No Causality Whatsoever

$P(A)=a$
$P(B|A)=b$
$P(B|\sim A)=c$



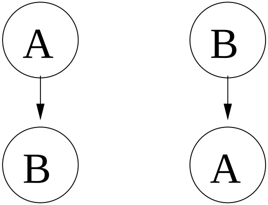
$P(B)=ab+(1-a)c$
$P(A|B)=ab/(ab+(1-a)c)$
$P(A|\sim B)=a(1-b)/(1-ab-(1-a)c)$

The two BNs are equivalent in all respects

- Do not *read* causality from Bayesian networks
- They only represent correlation (joint probability distribution)

No Causality Whatsoever

$P(A)=a$
$P(B|A)=b$
$P(B|\sim A)=c$



$P(B)=ab+(1-a)c$
$P(A|B)=ab/(ab+(1-a)c)$
$P(A|\sim B)=a(1-b)/(1-ab-(1-a)c)$

The two BNs are equivalent in all respects

- Do not *read* causality from Bayesian networks
- They only represent correlation (joint probability distribution)
- However, it is perfectly fine to *design* BNs causally

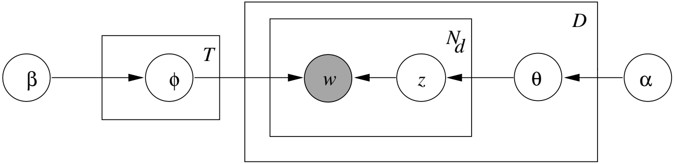
What do we need probabilistic models for?

- Make predictions. $p(y \mid x)$ plus decision theory

What do we need probabilistic models for?

- Make predictions. $p(y \mid x)$ plus decision theory
- Interpret models. Very natural to include latent variables

Example: Latent Dirichlet Allocation (LDA)



A generative model for $p(\phi, \theta, z, w \mid \alpha, \beta)$:
For each topic t

$\phi_t \sim$ Dirichlet(β)

For each document d

$\theta \sim$ Dirichlet(α)

For each word position in d

topic $z \sim$ Multinomial(θ)

word $w \sim$ Multinomial(ϕ_z)

Inference goals: $p(z \mid w, \alpha, \beta), \operatorname{argmax}_{\phi,\theta} p(\phi, \theta \mid w, \alpha, \beta)$

Conditional Independence

- Two r.v.s A, B are independent if

$$
\begin{aligned}
P(A, B) &= P(A)P(B) \\
P(A|B) &= P(A) \\
P(B|A) &= P(B)
\end{aligned}
$$

Conditional Independence

- Two r.v.s A, B are independent if

$$
\begin{aligned}
P(A, B) &= P(A)P(B) \\
P(A|B) &= P(A) \\
P(B|A) &= P(B)
\end{aligned}
$$

- Two r.v.s A, B are conditionally independent given C if

$$
\begin{aligned}
P(A, B \mid C) &= P(A \mid C)P(B \mid C) \\
P(A \mid B, C) &= P(A \mid C) \\
P(B \mid A, C) &= P(B \mid C)
\end{aligned}
$$

Conditional Independence

- Two r.v.s A, B are independent if

$$
\begin{aligned}
P(A,B) &= P(A)P(B) \\
P(A|B) &= P(A) \\
P(B|A) &= P(B)
\end{aligned}
$$

- Two r.v.s A, B are conditionally independent given C if

$$
\begin{aligned}
P(A,B \mid C) &= P(A \mid C)P(B \mid C) \\
P(A \mid B,C) &= P(A \mid C) \\
P(B \mid A,C) &= P(B \mid C)
\end{aligned}
$$

- This extends to groups of r.v.s

Conditional Independence

- Two r.v.s A, B are independent if

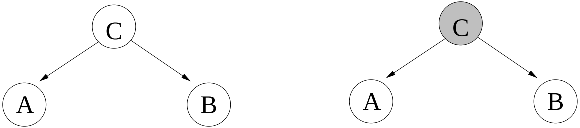
$$
\begin{aligned}
P(A,B) &= P(A)P(B) \\
P(A|B) &= P(A) \\
P(B|A) &= P(B)
\end{aligned}
$$

- Two r.v.s A, B are conditionally independent given C if

$$
\begin{aligned}
P(A,B \mid C) &= P(A \mid C)P(B \mid C) \\
P(A \mid B,C) &= P(A \mid C) \\
P(B \mid A,C) &= P(B \mid C)
\end{aligned}
$$

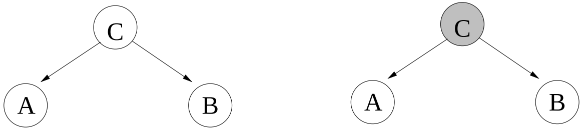
- This extends to groups of r.v.s
- Conditional independence in a BN is precisely specified by d-separation ("directed separation")

d-Separation Case 1: Tail-to-Tail



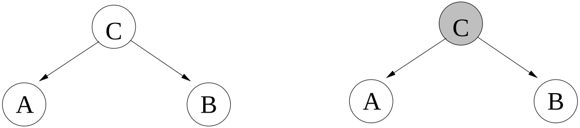
- A, B in general dependent

d-Separation Case 1: Tail-to-Tail



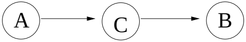
- A, B in general dependent
- A, B conditionally independent given C (observed nodes are shaded)

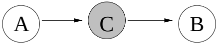
d-Separation Case 1: Tail-to-Tail



- A, B in general dependent
- A, B conditionally independent given C (observed nodes are shaded)
- An observed C is a tail-to-tail node, blocks the undirected path A-B

d-Separation Case 2: Head-to-Tail





- A, B in general dependent

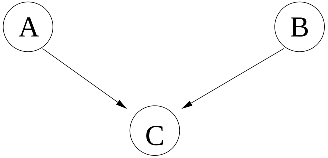
d-Separation Case 2: Head-to-Tail

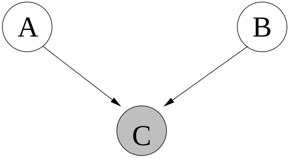
- A, B in general dependent
- A, B conditionally independent given C

d-Separation Case 2: Head-to-Tail

- A, B in general dependent
- A, B conditionally independent given C
- An observed C is a head-to-tail node, blocks the path A-B

d-Separation Case 3: Head-to-Head



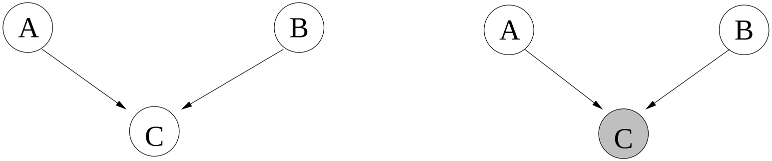


- A, B in general independent

d-Separation Case 3: Head-to-Head

- A, B in general independent
- A, B conditionally dependent given C, or any of C's descendants

d-Separation Case 3: Head-to-Head



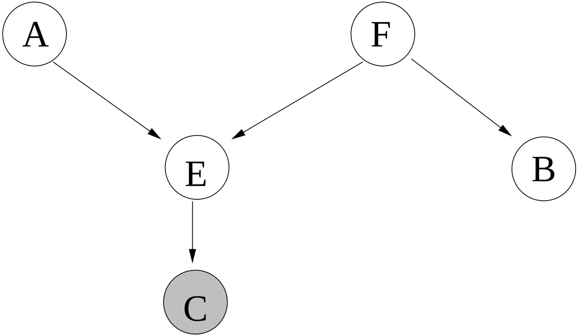
- A, B in general independent
- A, B conditionally dependent given C, or any of C's descendants
- An observed C is a head-to-head node, unblocks the path A-B

d-Separation

- Variable groups A and B are conditionally independent given C, if all undirected paths from nodes in A to nodes in B are *blocked*

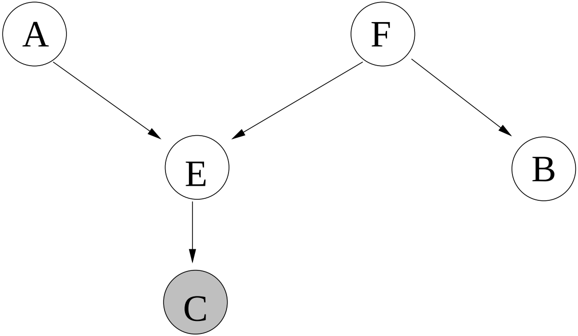
d-Separation Example 1

- The undirected path from A to B is unblocked by E (because of C), and is not blocked by F



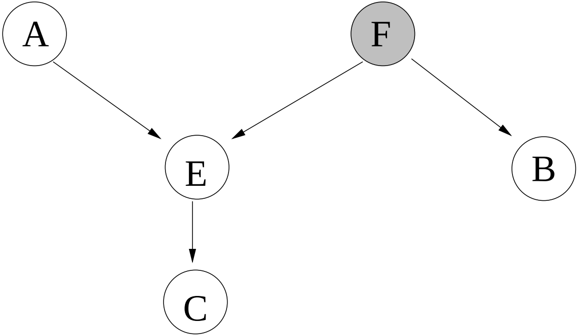
d-Separation Example 1

- The undirected path from A to B is unblocked by E (because of C), and is not blocked by F
- A, B dependent given C



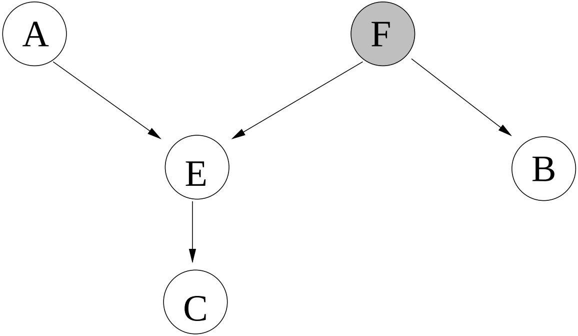
d-Separation Example 2

- The path from A to B is blocked both at E and F



d-Separation Example 2

- The path from A to B is blocked both at E and F
- A, B conditionally independent given F



Undirected graphical models

Undirected graphical models

- Also known as Markov Random Fields

Undirected graphical models

- Also known as Markov Random Fields
- Recall directed graphical models require a DAG and locally normalized CPDs

Undirected graphical models

- Also known as Markov Random Fields
- Recall directed graphical models require a DAG and locally normalized CPDs
  - efficient computation

Undirected graphical models

- Also known as Markov Random Fields
- Recall directed graphical models require a DAG and locally normalized CPDs
  - efficient computation
  - but restrictive

Undirected graphical models

- Also known as Markov Random Fields
- Recall directed graphical models require a DAG and locally normalized CPDs
  - efficient computation
  - but restrictive
- A clique C in an undirected graph is a set of fully connected nodes (full of loops!)

Undirected graphical models

- Also known as Markov Random Fields
- Recall directed graphical models require a DAG and locally normalized CPDs
  - efficient computation
  - but restrictive
- A clique C in an undirected graph is a set of fully connected nodes (full of loops!)
- Define a nonnegative potential function $\psi_C : X_C \mapsto \mathbb{R}_+$

Undirected graphical models

- Also known as Markov Random Fields
- Recall directed graphical models require a DAG and locally normalized CPDs
  - efficient computation
  - but restrictive
- A clique C in an undirected graph is a set of fully connected nodes (full of loops!)
- Define a nonnegative potential function $\psi_C : X_C \mapsto \mathbb{R}_+$
- An undirected graphical model is a family of distributions satisfying

$$\left\{ p \mid p(X) = \frac{1}{Z} \prod_C \psi_C(X_C) \right\}$$

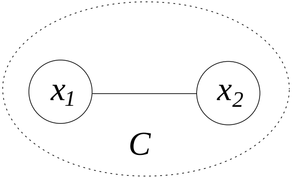
Undirected graphical models

- Also known as Markov Random Fields
- Recall directed graphical models require a DAG and locally normalized CPDs
  - efficient computation
  - but restrictive
- A clique C in an undirected graph is a set of fully connected nodes (full of loops!)
- Define a nonnegative potential function $\psi_C : X_C \mapsto \mathbb{R}_+$
- An undirected graphical model is a family of distributions satisfying

$$\left\{ p \mid p(X) = \frac{1}{Z} \prod_C \psi_C(X_C) \right\}$$

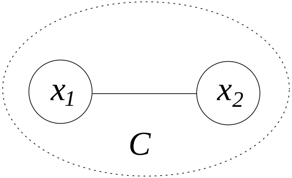
- $Z = \int \prod_C \psi_C(X_C) dX$ is the partition function

Example: A Tiny Markov Random Field



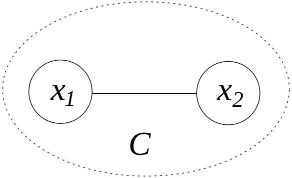
- $x_1, x_2 \in \{-1, 1\}$

Example: A Tiny Markov Random Field



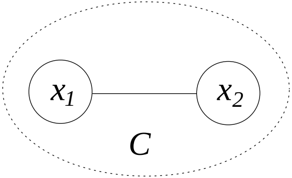
- $x_1, x_2 \in \{-1, 1\}$
- A single clique $\psi_C(x_1, x_2) = e^{a x_1 x_2}$

Example: A Tiny Markov Random Field



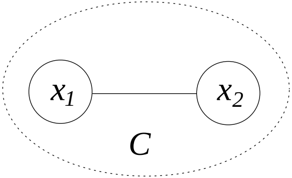
- $x_1, x_2 \in \{-1, 1\}$
- A single clique $\psi_C(x_1, x_2) = e^{ax_1x_2}$
- $p(x_1, x_2) = \frac{1}{Z}e^{ax_1x_2}$

Example: A Tiny Markov Random Field



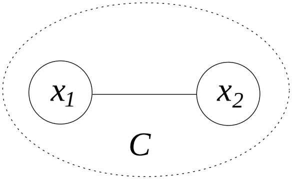
- $x_1, x_2 \in \{-1, 1\}$
- A single clique $\psi_C(x_1, x_2) = e^{ax_1x_2}$
- $p(x_1, x_2) = \frac{1}{Z} e^{ax_1x_2}$
- $Z = (e^a + e^{-a} + e^{-a} + e^a)$

Example: A Tiny Markov Random Field



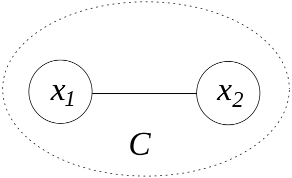
- $x_1, x_2 \in \{-1, 1\}$
- A single clique $\psi_C(x_1, x_2) = e^{ax_1x_2}$
- $p(x_1, x_2) = \frac{1}{Z}e^{ax_1x_2}$
- $Z = (e^a + e^{-a} + e^{-a} + e^a)$
- $p(1,1) = p(-1,-1) = e^a/(2e^a + 2e^{-a})$

Example: A Tiny Markov Random Field



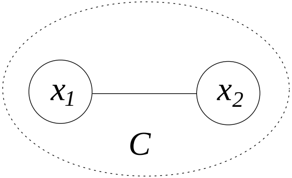
- $x_1, x_2 \in \{-1, 1\}$
- A single clique $\psi_C(x_1, x_2) = e^{ax_1x_2}$
- $p(x_1, x_2) = \frac{1}{Z} e^{ax_1x_2}$
- $Z = (e^a + e^{-a} + e^{-a} + e^a)$
- $p(1, 1) = p(-1, -1) = e^a/(2e^a + 2e^{-a})$
- $p(-1, 1) = p(1, -1) = e^{-a}/(2e^a + 2e^{-a})$

Example: A Tiny Markov Random Field



- $x_1, x_2 \in \{-1, 1\}$
- A single clique $\psi_C(x_1, x_2) = e^{a x_1 x_2}$
- $p(x_1, x_2) = \frac{1}{Z} e^{a x_1 x_2}$
- $Z = (e^a + e^{-a} + e^{-a} + e^a)$
- $p(1, 1) = p(-1, -1) = e^a / (2e^a + 2e^{-a})$
- $p(-1, 1) = p(1, -1) = e^{-a} / (2e^a + 2e^{-a})$
- When the parameter $a > 0$, favor homogeneous chains

Example: A Tiny Markov Random Field



- $x_1, x_2 \in \{-1, 1\}$
- A single clique $\psi_C(x_1, x_2) = e^{ax_1x_2}$
- $p(x_1, x_2) = \frac{1}{Z} e^{ax_1x_2}$
- $Z = (e^a + e^{-a} + e^{-a} + e^a)$
- $p(1, 1) = p(-1, -1) = e^a / (2e^a + 2e^{-a})$
- $p(-1, 1) = p(1, -1) = e^{-a} / (2e^a + 2e^{-a})$
- When the parameter $a > 0$, favor homogeneous chains
- When the parameter $a < 0$, favor inhomogeneous chains

Log-Linear Models

- Real-valued feature functions $f_1(X), \ldots, f_k(X)$

Log-Linear Models

- Real-valued feature functions $f_1(X), \ldots, f_k(X)$
- Real-valued weights $w_1, \ldots, w_k$

$$p(X) = \frac{1}{Z} \exp\left(\sum_{i=1}^{k} w_i f_i(X)\right)$$

Log-Linear Models

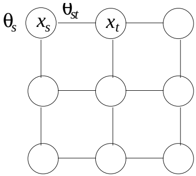
- Real-valued feature functions $f_1(X), \ldots, f_k(X)$
- Real-valued weights $w_1, \ldots, w_k$

$$p(X) = \frac{1}{Z} \exp\left(\sum_{i=1}^{k} w_i f_i(X)\right)$$

- Equivalent to MRF $p(X) = \frac{1}{Z} \prod_C \psi_C(X_C)$ with

$$\psi_C(X_C) = \exp\left(w_C f_C(X)\right)$$

Example: Ising Model

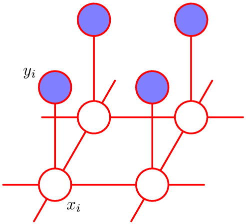


This is an undirected model with $x \in \{0, 1\}$.

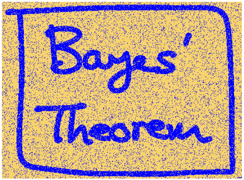
$$p_\theta(x) = \frac{1}{Z} \exp\left(\sum_{s \in V} \theta_s x_s + \sum_{(s,t) \in E} \theta_{st} x_s x_t \right)$$

- $f_s(X) = x_s$, $f_{st}(X) = x_s x_t$

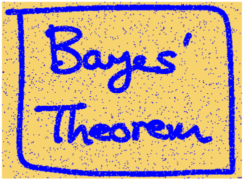
Example: Image Denoising



[From Bishop PRML]



noisy image



$\operatorname{argmax}_X P(X|Y)$

$$p_\theta(X \mid Y) = \frac{1}{Z} \exp\left(\sum_{s \in V} \theta_s x_s + \sum_{(s,t) \in E} \theta_{st} x_s x_t \right)$$

$$\theta_s = \begin{cases} c & y_s = 1 \\ -c & y_s = 0 \end{cases}, \quad \theta_{st} > 0$$

Example: Gaussian Random Field

$$p(X) \sim N(\mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu)^\top \Sigma^{-1}(X-\mu)\right)$$

- Multivariate Gaussian

Example: Gaussian Random Field

$$p(X) \sim N(\mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu)^\top \Sigma^{-1}(X-\mu)\right)$$

- Multivariate Gaussian
- The $n \times n$ covariance matrix Σ positive semi-definite

Example: Gaussian Random Field

$$p(X) \sim N(\mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu)^\top \Sigma^{-1}(X-\mu)\right)$$

- Multivariate Gaussian
- The $n \times n$ covariance matrix Σ positive semi-definite
- Let $\Omega = \Sigma^{-1}$ be the precision matrix

Example: Gaussian Random Field

$$p(X) \sim N(\mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu)^\top \Sigma^{-1}(X-\mu)\right)$$

- Multivariate Gaussian
- The $n \times n$ covariance matrix Σ positive semi-definite
- Let $\Omega = \Sigma^{-1}$ be the precision matrix
- x_i, x_j are conditionally independent given all other variables, if and only if $\Omega_{ij} = 0$

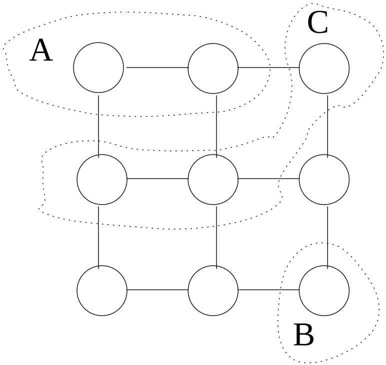
Example: Gaussian Random Field

$$p(X) \sim N(\mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu)^\top \Sigma^{-1}(X-\mu)\right)$$

- Multivariate Gaussian
- The $n \times n$ covariance matrix Σ positive semi-definite
- Let $\Omega = \Sigma^{-1}$ be the precision matrix
- x_i, x_j are conditionally independent given all other variables, if and only if $\Omega_{ij} = 0$
- When $\Omega_{ij} \neq 0$, there is an edge between x_i, x_j

Conditional Independence in Markov Random Fields

- Two group of variables A, B are conditionally independent given another group C, if A, B become disconnected by removing C and all edges involving C



Outline

Exact Inference

Inference by Enumeration

- Let $X = (X_Q, X_E, X_O)$ for query, evidence, and other variables.

Inference by Enumeration

- Let $X = (X_Q, X_E, X_O)$ for query, evidence, and other variables.
- Goal: $P(X_Q \mid X_E)$

Inference by Enumeration

- Let $X = (X_Q, X_E, X_O)$ for query, evidence, and other variables.
- Goal: $P(X_Q \mid X_E)$
- $$P(X_Q \mid X_E) = \frac{P(X_Q, X_E)}{P(X_E)} = \frac{\sum_{X_O} P(X_Q, X_E, X_O)}{\sum_{X_Q, X_O} P(X_Q, X_E, X_O)}$$

Inference by Enumeration

- Let $X = (X_Q, X_E, X_O)$ for query, evidence, and other variables.
- Goal: $P(X_Q \mid X_E)$
- $$P(X_Q \mid X_E) = \frac{P(X_Q, X_E)}{P(X_E)} = \frac{\sum_{X_O} P(X_Q, X_E, X_O)}{\sum_{X_Q, X_O} P(X_Q, X_E, X_O)}$$
- Summing exponential number of terms: with k variables in X_O each taking r values, there are r^k terms

Inference by Enumeration

- Let $X = (X_Q, X_E, X_O)$ for query, evidence, and other variables.
- Goal: $P(X_Q \mid X_E)$
- $$P(X_Q \mid X_E) = \frac{P(X_Q, X_E)}{P(X_E)} = \frac{\sum_{X_O} P(X_Q, X_E, X_O)}{\sum_{X_Q, X_O} P(X_Q, X_E, X_O)}$$
- Summing exponential number of terms: with k variables in X_O each taking r values, there are r^k terms
- Not covered: Variable elimination and junction tree (aka clique tree)

Markov Chain Monte Carlo

Markov Chain Monte Carlo

- Forward sampling

Markov Chain Monte Carlo

- Forward sampling
- Gibbs sampling

Markov Chain Monte Carlo

- Forward sampling
- Gibbs sampling
- Collapsed Gibbs sampling

Markov Chain Monte Carlo

- Forward sampling
- Gibbs sampling
- Collapsed Gibbs sampling
- Not covered: block Gibbs, Metropolis-Hastings, etc.

Markov Chain Monte Carlo

- Forward sampling
- Gibbs sampling
- Collapsed Gibbs sampling
- Not covered: block Gibbs, Metropolis-Hastings, etc.
- Unbiased (after burn-in), but can have high variance

Monte Carlo Methods

- Consider the inference problem $p(X_Q = c_Q \mid X_E)$ where $X_Q \cup X_E \subseteq \{x_1 \dots x_d\}$

$$p(X_Q = c_Q \mid X_E) = \int 1_{(x_Q = c_Q)} p(x_Q \mid X_E) dx_Q$$

Monte Carlo Methods

- Consider the inference problem $p(X_Q = c_Q \mid X_E)$ where $X_Q \cup X_E \subseteq \{x_1 \ldots x_d\}$

$$p(X_Q = c_Q \mid X_E) = \int 1_{(x_Q = c_Q)} p(x_Q \mid X_E) dx_Q$$

- If we can draw samples $x_Q^{(1)}, \ldots x_Q^{(m)} \sim p(x_Q \mid X_E)$, an unbiased estimator is

$$p(X_Q = c_Q \mid X_E) \approx \frac{1}{m} \sum_{i=1}^{m} 1_{(x_Q^{(i)} = c_Q)}$$

Monte Carlo Methods

- Consider the inference problem $p(X_Q = c_Q \mid X_E)$ where $X_Q \cup X_E \subseteq \{x_1 \ldots x_d\}$

$$p(X_Q = c_Q \mid X_E) = \int 1_{(x_Q = c_Q)} p(x_Q \mid X_E) dx_Q$$

- If we can draw samples $x_Q^{(1)}, \ldots x_Q^{(m)} \sim p(x_Q \mid X_E)$, an unbiased estimator is

$$p(X_Q = c_Q \mid X_E) \approx \frac{1}{m} \sum_{i=1}^{m} 1_{(x_Q^{(i)} = c_Q)}$$

- The variance of the estimator decreases as $O(1/m)$

Monte Carlo Methods

- Consider the inference problem $p(X_Q = c_Q \mid X_E)$ where $X_Q \cup X_E \subseteq \{x_1 \ldots x_d\}$

$$p(X_Q = c_Q \mid X_E) = \int 1_{(x_Q = c_Q)} p(x_Q \mid X_E) dx_Q$$

- If we can draw samples $x_Q^{(1)}, \ldots x_Q^{(m)} \sim p(x_Q \mid X_E)$, an unbiased estimator is

$$p(X_Q = c_Q \mid X_E) \approx \frac{1}{m} \sum_{i=1}^{m} 1_{(x_Q^{(i)} = c_Q)}$$

- The variance of the estimator decreases as $O(1/m)$
- Inference reduces to sampling from $p(x_Q \mid X_E)$

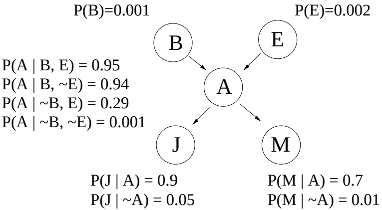
Forward Sampling

- Draw $X \sim P(X)$

Forward Sampling

- Draw $X \sim P(X)$
- Throw away X if it doesn't match the evidence X_E

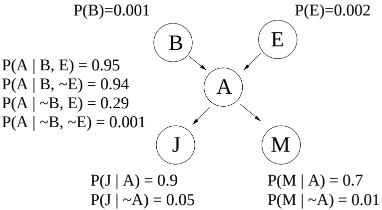
Forward Sampling: Example



To generate a sample $X = (B, E, A, J, M)$:

1. Sample $B \sim \mathrm{Ber}(0.001)$: $r \sim U(0, 1)$. If $(r < 0.001)$ then $B = 1$ else $B = 0$

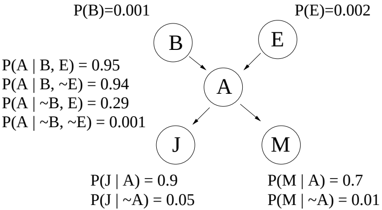
Forward Sampling: Example



To generate a sample $X = (B, E, A, J, M)$:

1. Sample $B \sim \mathrm{Ber}(0.001)$: $r \sim U(0,1)$. If $(r < 0.001)$ then $B = 1$ else $B = 0$
2. Sample $E \sim \mathrm{Ber}(0.002)$

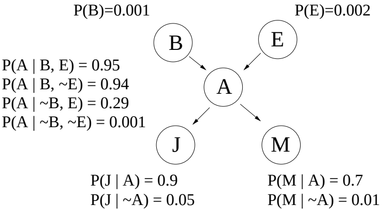
Forward Sampling: Example



To generate a sample $X = (B, E, A, J, M)$:

1. Sample $B \sim \mathrm{Ber}(0.001)$: $r \sim U(0,1)$. If $(r < 0.001)$ then $B = 1$ else $B = 0$
2. Sample $E \sim \mathrm{Ber}(0.002)$
3. If $B = 1$ and $E = 1$, sample $A \sim \mathrm{Ber}(0.95)$, and so on

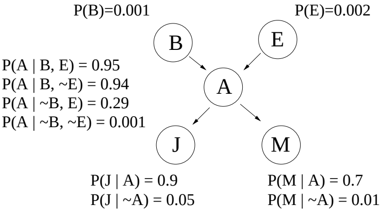
Forward Sampling: Example



To generate a sample $X = (B, E, A, J, M)$:

1. Sample $B \sim \mathrm{Ber}(0.001)$: $r \sim U(0, 1)$. If $(r < 0.001)$ then $B = 1$ else $B = 0$
2. Sample $E \sim \mathrm{Ber}(0.002)$
3. If $B = 1$ and $E = 1$, sample $A \sim \mathrm{Ber}(0.95)$, and so on
4. If $A = 1$ sample $J \sim \mathrm{Ber}(0.9)$ else $J \sim \mathrm{Ber}(0.05)$

Forward Sampling: Example



To generate a sample $X = (B, E, A, J, M)$:

1. Sample $B \sim \mathrm{Ber}(0.001)$: $r \sim U(0, 1)$. If $(r < 0.001)$ then $B = 1$ else $B = 0$
2. Sample $E \sim \mathrm{Ber}(0.002)$
3. If $B = 1$ and $E = 1$, sample $A \sim \mathrm{Ber}(0.95)$, and so on
4. If $A = 1$ sample $J \sim \mathrm{Ber}(0.9)$ else $J \sim \mathrm{Ber}(0.05)$
5. If $A = 1$ sample $M \sim \mathrm{Ber}(0.7)$ else $M \sim \mathrm{Ber}(0.01)$

Inference with Forward Sampling

- Say the inference task is $P(B = 1 \mid E = 1, M = 1)$

Inference with Forward Sampling

- Say the inference task is $P(B = 1 \mid E = 1, M = 1)$
- Throw away all samples except those with $(E = 1, M = 1)$

$$p(B = 1 \mid E = 1, M = 1) \approx \frac{1}{m} \sum_{i=1}^{m} 1_{(B^{(i)}=1)}$$

where m is the number of surviving samples

Inference with Forward Sampling

- Say the inference task is $P(B = 1 \mid E = 1, M = 1)$
- Throw away all samples except those with $(E = 1, M = 1)$

$$p(B = 1 \mid E = 1, M = 1) \approx \frac{1}{m} \sum_{i=1}^{m} 1_{(B^{(i)}=1)}$$

  where m is the number of surviving samples
- Can be highly inefficient (note $P(E = 1)$ tiny)

Inference with Forward Sampling

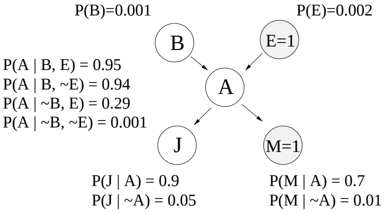
- Say the inference task is $P(B = 1 \mid E = 1, M = 1)$
- Throw away all samples except those with $(E = 1, M = 1)$

$$p(B = 1 \mid E = 1, M = 1) \approx \frac{1}{m} \sum_{i=1}^{m} 1_{(B^{(i)}=1)}$$

  where m is the number of surviving samples
- Can be highly inefficient (note $P(E = 1)$ tiny)
- Does not work for Markov Random Fields (can't sample from $P(X)$)

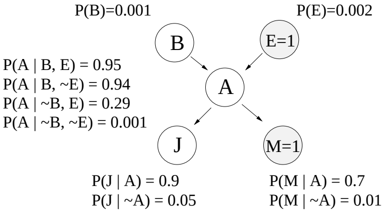
Gibbs Sampling: Example $P(B = 1 \mid E = 1, M = 1)$

- Gibbs sampling is a Markov Chain Monte Carlo (MCMC) method.



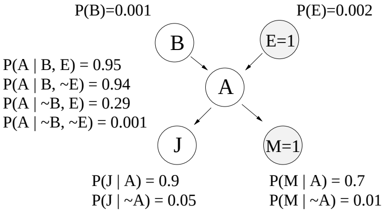
Gibbs Sampling: Example $P(B = 1 \mid E = 1, M = 1)$

- Gibbs sampling is a Markov Chain Monte Carlo (MCMC) method.
- Directly sample from $p(x_Q \mid X_E)$



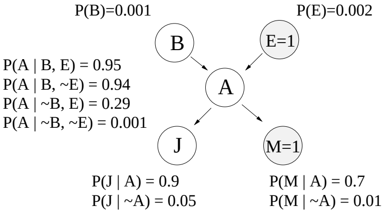
Gibbs Sampling: Example $P(B = 1 \mid E = 1, M = 1)$

- Gibbs sampling is a Markov Chain Monte Carlo (MCMC) method.
- Directly sample from $p(x_Q \mid X_E)$
- Works for both graphical models



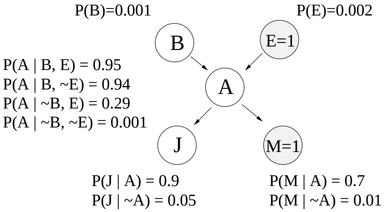
Gibbs Sampling: Example $P(B = 1 \mid E = 1, M = 1)$

- Gibbs sampling is a Markov Chain Monte Carlo (MCMC) method.
- Directly sample from $p(x_Q \mid X_E)$
- Works for both graphical models
- Initialization:



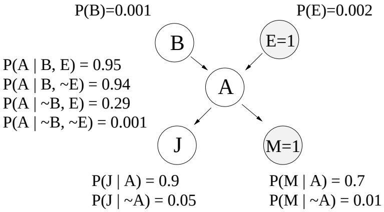
Gibbs Sampling: Example $P(B = 1 \mid E = 1, M = 1)$

- Gibbs sampling is a Markov Chain Monte Carlo (MCMC) method.
- Directly sample from $p(x_Q \mid X_E)$
- Works for both graphical models
- Initialization:
  - Fix evidence; randomly set other variables



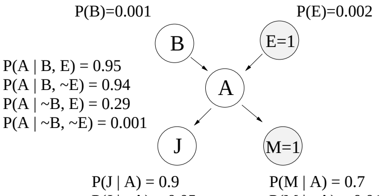
Gibbs Sampling: Example $P(B = 1 \mid E = 1, M = 1)$

- Gibbs sampling is a Markov Chain Monte Carlo (MCMC) method.
- Directly sample from $p(x_Q \mid X_E)$
- Works for both graphical models
- Initialization:
  - Fix evidence; randomly set other variables
  - e.g. $X^{(0)} = (B = 0, E = 1, A = 0, J = 0, M = 1)$



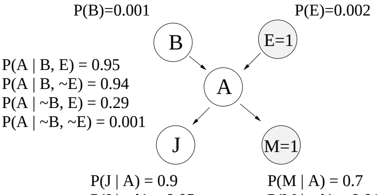
Gibbs Sampling

- For each non-evidence variable x_i, fixing all other nodes X_{-i}, resample its value $x_i \sim P(x_i \mid X_{-i})$



Gibbs Sampling

- For each non-evidence variable x_i, fixing all other nodes X_{-i}, resample its value $x_i \sim P(x_i \mid X_{-i})$
- This is equivalent to $x_i \sim P(x_i \mid \text{MarkovBlanket}(x_i))$

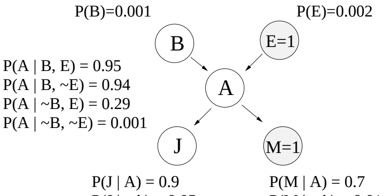


Gibbs Sampling

- For each non-evidence variable x_i, fixing all other nodes X_{-i}, resample its value $x_i \sim P(x_i \mid X_{-i})$
- This is equivalent to $x_i \sim P(x_i \mid \text{MarkovBlanket}(x_i))$
- For a Bayesian network MarkovBlanket(x_i) includes x_i's parents, spouses, and children

$$P(x_i \mid \text{MarkovBlanket}(x_i)) \propto P(x_i \mid Pa(x_i)) \prod_{y \in C(x_i)} P(y \mid Pa(y))$$

where $Pa(x)$ are the parents of x, and $C(x)$ the children of x.

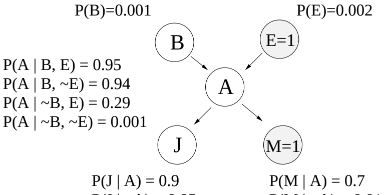


Gibbs Sampling

- For each non-evidence variable x_i, fixing all other nodes X_{-i}, resample its value $x_i \sim P(x_i \mid X_{-i})$
- This is equivalent to $x_i \sim P(x_i \mid \mathsf{MarkovBlanket}(x_i))$
- For a Bayesian network $\mathsf{MarkovBlanket}(x_i)$ includes x_i's parents, spouses, and children

  $$P(x_i \mid \mathsf{MarkovBlanket}(x_i)) \propto P(x_i \mid Pa(x_i)) \prod_{y \in C(x_i)} P(y \mid Pa(y))$$

  where $Pa(x)$ are the parents of x, and $C(x)$ the children of x.
- For many graphical models the Markov Blanket is small.

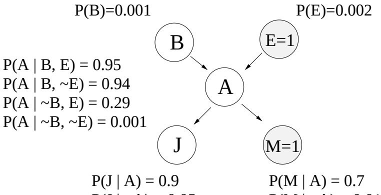


Gibbs Sampling

- For each non-evidence variable x_i, fixing all other nodes X_{-i}, resample its value $x_i \sim P(x_i \mid X_{-i})$
- This is equivalent to $x_i \sim P(x_i \mid \text{MarkovBlanket}(x_i))$
- For a Bayesian network $\text{MarkovBlanket}(x_i)$ includes x_i's parents, spouses, and children

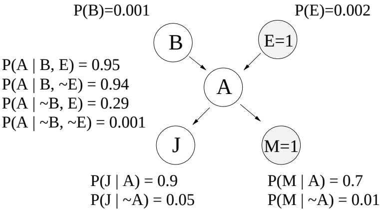
  $$P(x_i \mid \text{MarkovBlanket}(x_i)) \propto P(x_i \mid Pa(x_i)) \prod_{y \in C(x_i)} P(y \mid Pa(y))$$

  where $Pa(x)$ are the parents of x, and $C(x)$ the children of x.
- For many graphical models the Markov Blanket is small.
- For example,
  $B \sim P(B \mid E = 1, A = 0) \propto P(B)P(A = 0 \mid B, E = 1)$



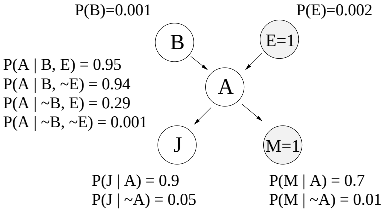
Gibbs Sampling

- Say we sampled $B = 1$. Then $X^{(1)} = (B = 1, E = 1, A = 0, J = 0, M = 1)$



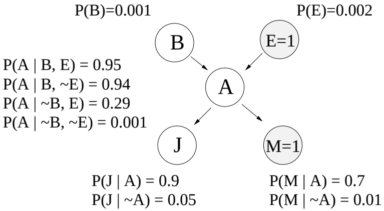
Gibbs Sampling

- Say we sampled $B = 1$. Then $X^{(1)} = (B = 1, E = 1, A = 0, J = 0, M = 1)$
- Starting from $X^{(1)}$, sample $A \sim P(A \mid B = 1, E = 1, J = 0, M = 1)$ to get $X^{(2)}$



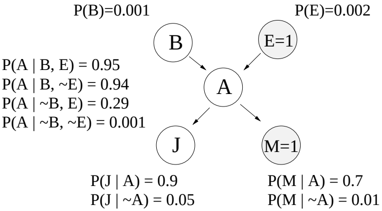
Gibbs Sampling

- Say we sampled $B = 1$. Then $X^{(1)} = (B = 1, \color{red}{E = 1}, A = 0, J = 0, \color{red}{M = 1})$
- Starting from $X^{(1)}$, sample $A \sim P(A \mid B = 1, E = 1, J = 0, M = 1)$ to get $X^{(2)}$
- Move on to J, then repeat $B, A, J, B, A, J \ldots$

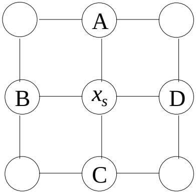


Gibbs Sampling

- Say we sampled $B = 1$. Then $X^{(1)} = (B = 1, E = 1, A = 0, J = 0, M = 1)$
- Starting from $X^{(1)}$, sample $A \sim P(A \mid B = 1, E = 1, J = 0, M = 1)$ to get $X^{(2)}$
- Move on to J, then repeat $B, A, J, B, A, J \ldots$
- Keep all samples after *burn in*. $P(B = 1 \mid E = 1, M = 1)$ is the fraction of samples with $B = 1$.



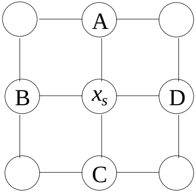
Gibbs Sampling Example 2: The Ising Model



This is an undirected model with $x \in \{0, 1\}$.

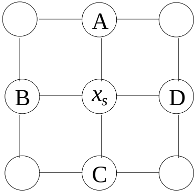
$$p_\theta(x) = \frac{1}{Z} \exp \left(\sum_{s \in V} \theta_s x_s + \sum_{(s,t) \in E} \theta_{st} x_s x_t \right)$$

Gibbs Example 2: The Ising Model



- The Markov blanket of x_s is A, B, C, D

Gibbs Example 2: The Ising Model

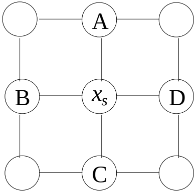


- The Markov blanket of x_s is A, B, C, D
- In general for undirected graphical models

$$p(x_s \mid x_{-s}) = p(x_s \mid x_{N(s)})$$

$N(s)$ is the neighbors of s.

Gibbs Example 2: The Ising Model



- The Markov blanket of x_s is A, B, C, D
- In general for undirected graphical models

$$p(x_s \mid x_{-s}) = p(x_s \mid x_{N(s)})$$

$N(s)$ is the neighbors of s.

- The Gibbs update is

$$p(x_s = 1 \mid x_{N(s)}) = \frac{1}{\exp(-(\theta_s + \sum_{t \in N(s)} \theta_{st} x_t)) + 1}$$

Gibbs Sampling as a Markov Chain

- A Markov chain is defined by a transition matrix $T(X' \mid X)$

Gibbs Sampling as a Markov Chain

- A Markov chain is defined by a transition matrix $T(X' \mid X)$
- Certain Markov chains have a stationary distribution π such that $\pi = T\pi$

Gibbs Sampling as a Markov Chain

- A Markov chain is defined by a transition matrix $T(X' \mid X)$
- Certain Markov chains have a stationary distribution π such that $\pi = T\pi$
- Gibbs sampler is such a Markov chain with $T_i((X_{-i}, x_i') \mid (X_{-i}, x_i)) = p(x_i' \mid X_{-i})$, and stationary distribution $p(x_Q \mid X_E)$

Gibbs Sampling as a Markov Chain

- A Markov chain is defined by a transition matrix $T(X' \mid X)$
- Certain Markov chains have a stationary distribution π such that $\pi = T\pi$
- Gibbs sampler is such a Markov chain with $T_i((X_{-i}, x_i') \mid (X_{-i}, x_i)) = p(x_i' \mid X_{-i})$, and stationary distribution $p(x_Q \mid X_E)$
- But it takes time for the chain to reach stationary distribution (mix)

Gibbs Sampling as a Markov Chain

- A Markov chain is defined by a transition matrix $T(X' \mid X)$
- Certain Markov chains have a stationary distribution π such that $\pi = T\pi$
- Gibbs sampler is such a Markov chain with $T_i((X_{-i}, x_i') \mid (X_{-i}, x_i)) = p(x_i' \mid X_{-i})$, and stationary distribution $p(x_Q \mid X_E)$
- But it takes time for the chain to reach stationary distribution (mix)
  - Can be difficult to assert mixing

Gibbs Sampling as a Markov Chain

- A Markov chain is defined by a transition matrix $T(X' \mid X)$
- Certain Markov chains have a stationary distribution π such that $\pi = T\pi$
- Gibbs sampler is such a Markov chain with $T_i((X_{-i}, x_i') \mid (X_{-i}, x_i)) = p(x_i' \mid X_{-i})$, and stationary distribution $p(x_Q \mid X_E)$
- But it takes time for the chain to reach stationary distribution (mix)
  - Can be difficult to assert mixing
  - In practice "burn in": discard $X^{(0)}, \ldots, X^{(T)}$

Gibbs Sampling as a Markov Chain

- A Markov chain is defined by a transition matrix $T(X' \mid X)$
- Certain Markov chains have a stationary distribution π such that $\pi = T\pi$
- Gibbs sampler is such a Markov chain with $T_i((X_{-i}, x_i') \mid (X_{-i}, x_i)) = p(x_i' \mid X_{-i})$, and stationary distribution $p(x_Q \mid X_E)$
- But it takes time for the chain to reach stationary distribution (mix)
  - Can be difficult to assert mixing
  - In practice "burn in": discard $X^{(0)}, \ldots, X^{(T)}$
  - Use all of $X^{(T+1)}, \ldots$ for inference (they are correlated); Do not thin

Collapsed Gibbs Sampling

- In general, $\mathbb{E}_p[f(X)] \approx \frac{1}{m}\sum_{i=1}^{m} f(X^{(i)})$ for $X^{(i)} \sim p$

Collapsed Gibbs Sampling

- In general, $\mathbb{E}_p[f(X)] \approx \frac{1}{m}\sum_{i=1}^{m} f(X^{(i)})$ for $X^{(i)} \sim p$
- Sometimes $X = (Y, Z)$ where $\mathbb{E}_{Z|Y}$ has a closed-form

Collapsed Gibbs Sampling

- In general, $\mathbb{E}_p[f(X)] \approx \frac{1}{m}\sum_{i=1}^{m} f(X^{(i)})$ for $X^{(i)} \sim p$
- Sometimes $X = (Y, Z)$ where $\mathbb{E}_{Z|Y}$ has a closed-form
- If so,

$$\begin{aligned}\mathbb{E}_p[f(X)] &= \mathbb{E}_{p(Y)}\mathbb{E}_{p(Z|Y)}[f(Y,Z)] \\ &\approx \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{p(Z|Y^{(i)})}[f(Y^{(i)},Z)]\end{aligned}$$

for $Y^{(i)} \sim p(Y)$

Collapsed Gibbs Sampling

- In general, $\mathbb{E}_p[f(X)] \approx \frac{1}{m}\sum_{i=1}^m f(X^{(i)})$ for $X^{(i)} \sim p$
- Sometimes $X = (Y, Z)$ where $\mathbb{E}_{Z|Y}$ has a closed-form
- If so,

$$\begin{aligned} \mathbb{E}_p[f(X)] &= \mathbb{E}_{p(Y)}\mathbb{E}_{p(Z|Y)}[f(Y,Z)] \\ &\approx \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{p(Z|Y^{(i)})}[f(Y^{(i)}, Z)] \end{aligned}$$

  for $Y^{(i)} \sim p(Y)$
- No need to sample Z: it is collapsed

Collapsed Gibbs Sampling

- In general, $\mathbb{E}_p[f(X)] \approx \frac{1}{m}\sum_{i=1}^m f(X^{(i)})$ for $X^{(i)} \sim p$
- Sometimes $X = (Y, Z)$ where $\mathbb{E}_{Z|Y}$ has a closed-form
- If so,

$$\begin{aligned} \mathbb{E}_p[f(X)] &= \mathbb{E}_{p(Y)}\mathbb{E}_{p(Z|Y)}[f(Y,Z)] \\ &\approx \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{p(Z|Y^{(i)})}[f(Y^{(i)},Z)] \end{aligned}$$

  for $Y^{(i)} \sim p(Y)$
- No need to sample Z: it is collapsed
- Collapsed Gibbs sampler $T_i((Y_{-i}, y_i') \mid (Y_{-i}, y_i)) = p(y_i' \mid Y_{-i})$

Collapsed Gibbs Sampling

- In general, $\mathbb{E}_p[f(X)] \approx \frac{1}{m}\sum_{i=1}^m f(X^{(i)})$ for $X^{(i)} \sim p$
- Sometimes $X = (Y, Z)$ where $\mathbb{E}_{Z|Y}$ has a closed-form
- If so,

$$\begin{aligned}\mathbb{E}_p[f(X)] &= \mathbb{E}_{p(Y)}\mathbb{E}_{p(Z|Y)}[f(Y,Z)] \\ &\approx \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{p(Z|Y^{(i)})}[f(Y^{(i)}, Z)]\end{aligned}$$

  for $Y^{(i)} \sim p(Y)$
- No need to sample Z: it is collapsed
- Collapsed Gibbs sampler $T_i((Y_{-i}, y_i') \mid (Y_{-i}, y_i)) = p(y_i' \mid Y_{-i})$
- Note $p(y_i' \mid Y_{-i}) = \int p(y_i', Z \mid Y_{-i}) dZ$

Example: Collapsed Gibbs Sampling for LDA

Collapse θ, ϕ, Gibbs update:

$$P(z_i = j \mid \mathbf{z}_{-i}, \mathbf{w}) \propto \frac{n^{(w_i)}_{-i,j} + \beta n^{(d_i)}_{-i,j} + \alpha}{n^{(\cdot)}_{-i,j} + W\beta n^{(d_i)}_{-i,\cdot} + T\alpha}$$

- $n^{(w_i)}_{-i,j}$: number of times word w_i has been assigned to topic j, excluding the current position

Example: Collapsed Gibbs Sampling for LDA

Collapse θ, ϕ, Gibbs update:

$$P(z_i = j \mid \mathbf{z}_{-i}, \mathbf{w}) \propto \frac{n^{(w_i)}_{-i,j} + \beta n^{(d_i)}_{-i,j} + \alpha}{n^{(\cdot)}_{-i,j} + W\beta n^{(d_i)}_{-i,\cdot} + T\alpha}$$

- $n^{(w_i)}_{-i,j}$: number of times word w_i has been assigned to topic j, excluding the current position
- $n^{(d_i)}_{-i,j}$: number of times a word from document d_i has been assigned to topic j, excluding the current position

Example: Collapsed Gibbs Sampling for LDA

Collapse θ, ϕ, Gibbs update:

$$P(z_i = j \mid \mathbf{z}_{-i}, \mathbf{w}) \propto \frac{n^{(w_i)}_{-i,j} + \beta n^{(d_i)}_{-i,j} + \alpha}{n^{(\cdot)}_{-i,j} + W\beta n^{(d_i)}_{-i,\cdot} + T\alpha}$$

- $n^{(w_i)}_{-i,j}$: number of times word w_i has been assigned to topic j, excluding the current position
- $n^{(d_i)}_{-i,j}$: number of times a word from document d_i has been assigned to topic j, excluding the current position
- $n^{(\cdot)}_{-i,j}$: number of times any word has been assigned to topic j, excluding the current position

Example: Collapsed Gibbs Sampling for LDA

Collapse θ, ϕ, Gibbs update:

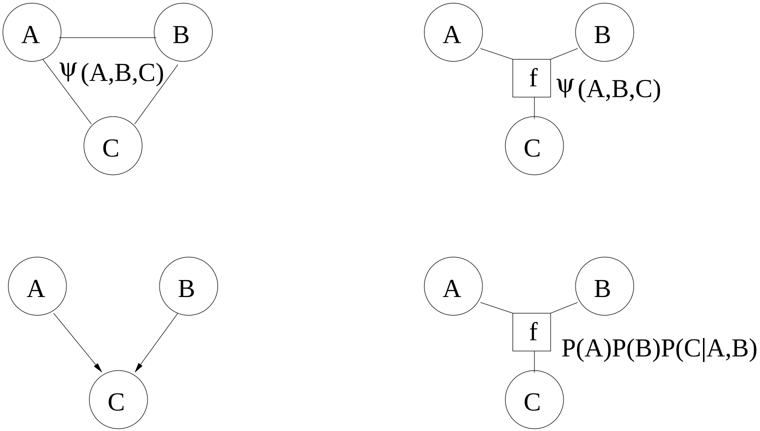
$$P(z_i = j \mid \mathbf{z}_{-i}, \mathbf{w}) \propto \frac{n_{-i,j}^{(w_i)} + \beta n_{-i,j}^{(d_i)} + \alpha}{n_{-i,j}^{(\cdot)} + W\beta n_{-i,\cdot}^{(d_i)} + T\alpha}$$

- $n_{-i,j}^{(w_i)}$: number of times word w_i has been assigned to topic j, excluding the current position
- $n_{-i,j}^{(d_i)}$: number of times a word from document d_i has been assigned to topic j, excluding the current position
- $n_{-i,j}^{(\cdot)}$: number of times any word has been assigned to topic j, excluding the current position
- $n_{-i,\cdot}^{(d_i)}$: length of document d_i, excluding the current position

Belief Propagation

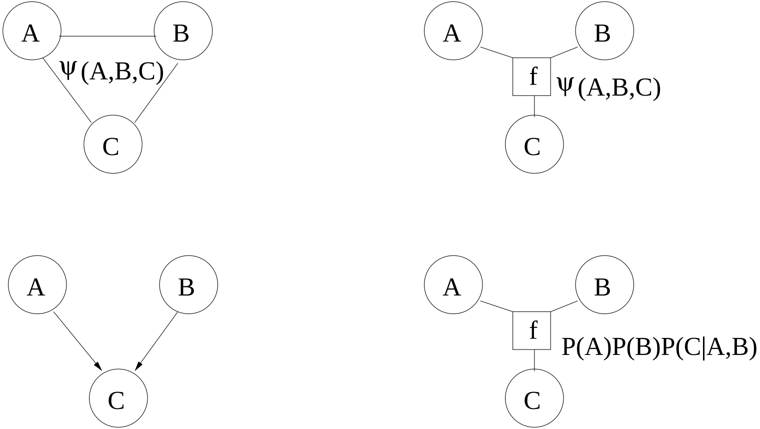
Factor Graph

- For both directed and undirected graphical models



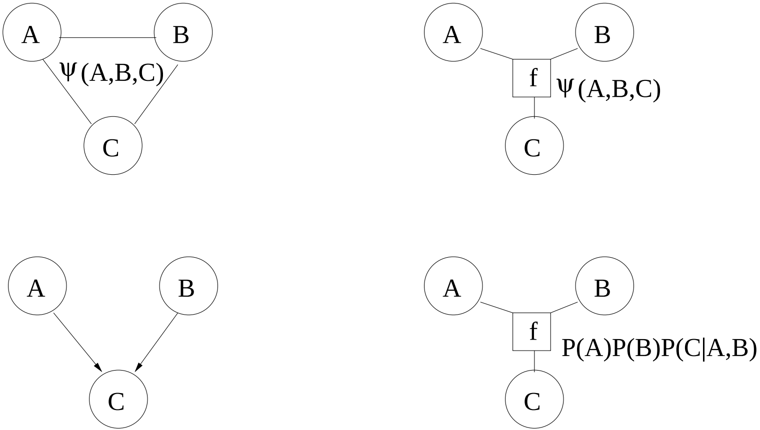
Factor Graph

- For both directed and undirected graphical models
- Bipartite: edges between a variable node and a factor node



Factor Graph

- For both directed and undirected graphical models
- Bipartite: edges between a variable node and a factor node
- Factors represent computation



The Sum-Product Algorithm

- Also known as belief propagation (BP)

The Sum-Product Algorithm

- Also known as belief propagation (BP)
- Exact if the graph is a tree; otherwise known as “loopy BP”, approximate

The Sum-Product Algorithm

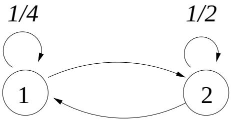
- Also known as belief propagation (BP)
- Exact if the graph is a tree; otherwise known as "loopy BP", approximate
- The algorithm involves passing *messages* on the factor graph

The Sum-Product Algorithm

- Also known as belief propagation (BP)
- Exact if the graph is a tree; otherwise known as "loopy BP", approximate
- The algorithm involves passing *messages* on the factor graph
- Alternative view: variational approximation (more later)

Example: A Simple HMM

- The Hidden Markov Model template (not a graphical model)



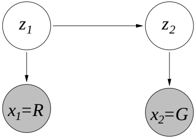
$P(x \mid z=1) = (1/2, 1/4, 1/4)$ — R G B

$P(x \mid z=2) = (1/4, 1/2, 1/4)$ — R G B

$\pi_1 = \pi_2 = 1/2$

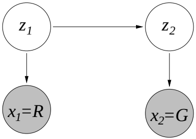
Example: A Simple HMM

- Observing $x_1 = R, x_2 = G$, the directed graphical model

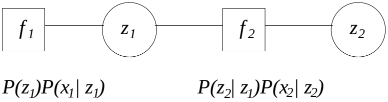


Example: A Simple HMM

- Observing $x_1 = R, x_2 = G$, the directed graphical model



- Factor graph



Messages

- A message is a vector of length K, where K is the number of values x takes.

Messages

- A message is a vector of length K, where K is the number of values x takes.
- There are two types of messages:

Messages

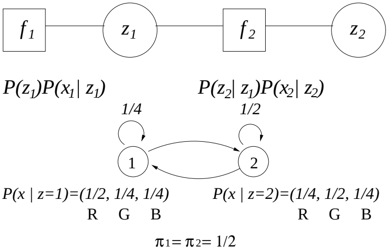
- A message is a vector of length K, where K is the number of values x takes.
- There are two types of messages:
    1. $\mu_{f \to x}$: message from a factor node f to a variable node x
       $\mu_{f \to x}(i)$ is the ith element, $i = 1 \ldots K$.

Messages

- A message is a vector of length K, where K is the number of values x takes.
- There are two types of messages:
  1. $\mu_{f \to x}$: message from a factor node f to a variable node x
     $\mu_{f \to x}(i)$ is the ith element, $i = 1 \ldots K$.
  2. $\mu_{x \to f}$: message from a variable node x to a factor node f

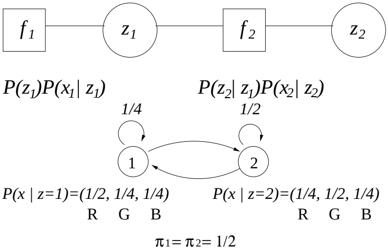
Leaf Messages

- Assume tree factor graph. Pick an arbitrary root, say z_2



Leaf Messages

- Assume tree factor graph. Pick an arbitrary root, say z_2
- Start messages at leaves.

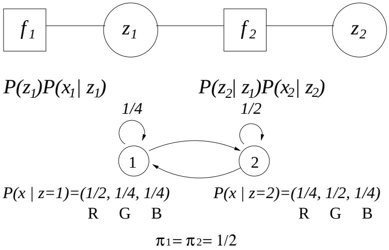


Leaf Messages

- Assume tree factor graph. Pick an arbitrary root, say z_2
- Start messages at leaves.
- If a leaf is a factor node f, $\mu_{f\to x}(x) = f(x)$

$$\mu_{f_1\to z_1}(z_1 = 1) = P(z_1 = 1)P(R|z_1 = 1) = 1/2 \cdot 1/2 = 1/4$$

$$\mu_{f_1\to z_1}(z_1 = 2) = P(z_1 = 2)P(R|z_1 = 2) = 1/2 \cdot 1/4 = 1/8$$



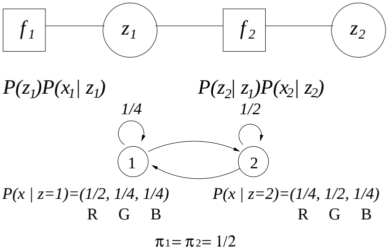
Leaf Messages

- Assume tree factor graph. Pick an arbitrary root, say z_2
- Start messages at leaves.
- If a leaf is a factor node f, $\mu_{f \to x}(x) = f(x)$

$$\mu_{f_1 \to z_1}(z_1 = 1) = P(z_1 = 1)P(R|z_1 = 1) = 1/2 \cdot 1/2 = 1/4$$

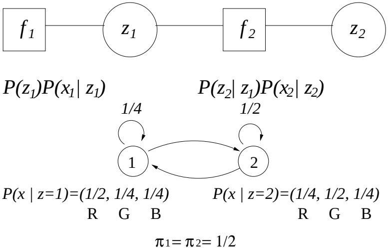
$$\mu_{f_1 \to z_1}(z_1 = 2) = P(z_1 = 2)P(R|z_1 = 2) = 1/2 \cdot 1/4 = 1/8$$

- If a leaf is a variable node x, $\mu_{x \to f}(x) = 1$



Message from Variable to Factor

- A node (factor or variable) can send out a message if all other incoming messages have arrived



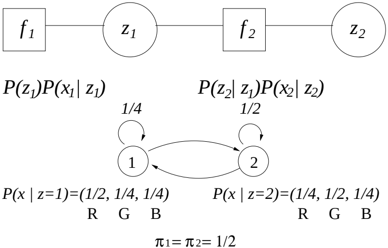
Message from Variable to Factor

- A node (factor or variable) can send out a message if all other incoming messages have arrived
- Let x be in factor f_s. $ne(x)\backslash f_s$ are factors connected to x excluding f_s.

$$\mu_{x \to f_s}(x) = \prod_{f \in ne(x)\backslash f_s} \mu_{f \to x}(x)$$

$$\mu_{z_1 \to f_2}(z_1 = 1) = 1/4$$

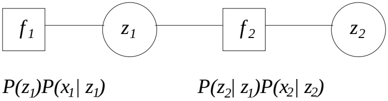
$$\mu_{z_1 \to f_2}(z_1 = 2) = 1/8$$



Message from Factor to Variable

- Let x be in factor f_s. Let the other variables in f_s be $x_{1:M}$.

$$\mu_{f_s \to x}(x) = \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m=1}^{M} \mu_{x_m \to f_s}(x_m)$$



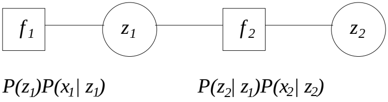
Message from Factor to Variable

- Let x be in factor f_s. Let the other variables in f_s be $x_{1:M}$.

$$\mu_{f_s \to x}(x) = \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m=1}^{M} \mu_{x_m \to f_s}(x_m)$$

- In this example

$$\begin{aligned}
\mu_{f_2 \to z_2}(s) &= \sum_{s'=1}^{2} \mu_{z_1 \to f_2}(s') f_2(z_1 = s', z_2 = s) \\
&= 1/4 P(z_2 = s | z_1 = 1) P(x_2 = G | z_2 = s) \\
&\quad +1/8 P(z_2 = s | z_1 = 2) P(x_2 = G | z_2 = s)
\end{aligned}$$



Message from Factor to Variable

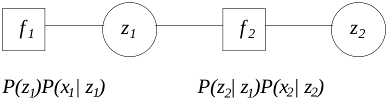
- Let x be in factor f_s. Let the other variables in f_s be $x_{1:M}$.

$$\mu_{f_s \to x}(x) = \sum_{x_1} \dots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m=1}^{M} \mu_{x_m \to f_s}(x_m)$$

- In this example

$$\begin{aligned} \mu_{f_2 \to z_2}(s) &= \sum_{s'=1}^{2} \mu_{z_1 \to f_2}(s') f_2(z_1 = s', z_2 = s) \\ &= 1/4 P(z_2 = s | z_1 = 1) P(x_2 = G | z_2 = s) \\ &\quad +1/8 P(z_2 = s | z_1 = 2) P(x_2 = G | z_2 = s) \end{aligned}$$

- We get $\mu_{f_2 \to z_2}(z_2 = 1) = 1/32$, $\mu_{f_2 \to z_2}(z_2 = 2) = 1/8$

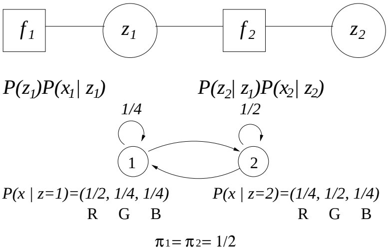


Up to Root, Back Down

- The message has reached the root, pass it back down

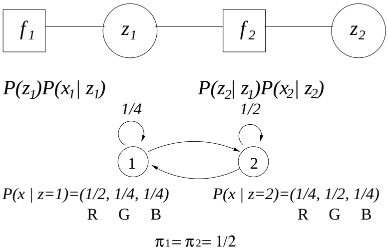
$$\mu_{z_2 \to f_2}(z_2 = 1) = 1$$

$$\mu_{z_2 \to f_2}(z_2 = 2) = 1$$



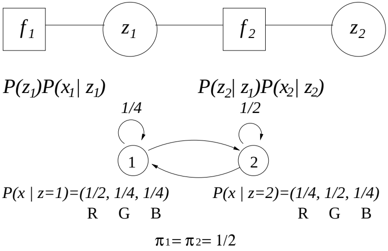
Keep Passing Down

- $\mu_{f_2 \to z_1}(s) = \sum_{s'=1}^{2} \mu_{z_2 \to f_2}(s') f_2(z_1 = s, z_2 = s')$
  $= 1P(z_2 = 1|z_1 = s)P(x_2 = G|z_2 = 1)$
  $+ 1P(z_2 = 2|z_1 = s)P(x_2 = G|z_2 = 2)$.



Keep Passing Down

- $\mu_{f_2 \to z_1}(s) = \sum_{s'=1}^{2} \mu_{z_2 \to f_2}(s') f_2(z_1 = s, z_2 = s')$
  $= 1P(z_2 = 1 | z_1 = s)P(x_2 = G | z_2 = 1)$
  $\quad + 1P(z_2 = 2 | z_1 = s)P(x_2 = G | z_2 = 2).$
- We get
  $\mu_{f_2 \to z_1}(z_1 = 1) = 7/16$
  $\mu_{f_2 \to z_1}(z_1 = 2) = 3/8$



From Messages to Marginals

- Once a variable receives all incoming messages, we compute its marginal as

$$p(x) \propto \prod_{f \in ne(x)} \mu_{f \to x}(x)$$

From Messages to Marginals

- Once a variable receives all incoming messages, we compute its marginal as
$$p(x) \propto \prod_{f \in ne(x)} \mu_{f \to x}(x)$$
- In this example
$P(z_1|x_1, x_2) \propto \mu_{f_1 \to z_1} \cdot \mu_{f_2 \to z_1} = \binom{1/4}{1/8} \cdot \binom{7/16}{3/8} = \binom{7/64}{3/64} \Rightarrow \binom{0.7}{0.3}$
$P(z_2|x_1, x_2) \propto \mu_{f_2 \to z_2} = \binom{1/32}{1/8} \Rightarrow \binom{0.2}{0.8}$

From Messages to Marginals

- Once a variable receives all incoming messages, we compute its marginal as

$$p(x) \propto \prod_{f \in ne(x)} \mu_{f \to x}(x)$$

- In this example
  $P(z_1|x_1, x_2) \propto \mu_{f_1 \to z_1} \cdot \mu_{f_2 \to z_1} = \binom{1/4}{1/8} \cdot \binom{7/16}{3/8} = \binom{7/64}{3/64} \Rightarrow \binom{0.7}{0.3}$
  $P(z_2|x_1, x_2) \propto \mu_{f_2 \to z_2} = \binom{1/32}{1/8} \Rightarrow \binom{0.2}{0.8}$
- One can also compute the marginal of the *set of* variables x_s involved in a factor f_s

$$p(x_s) \propto f_s(x_s) \prod_{x \in ne(f)} \mu_{x \to f}(x)$$

Handling Evidence

- Observing $x = v$,

Handling Evidence

- Observing $x = v$,
  - we can absorb it in the factor (as we did); or

Handling Evidence

- Observing $x = v$,
  - we can absorb it in the factor (as we did); or
  - set messages $\mu_{x \to f}(x) = 0$ for all $x \neq v$

Handling Evidence

- Observing $x = v$,
  - we can absorb it in the factor (as we did); or
  - set messages $\mu_{x \to f}(x) = 0$ for all $x \neq v$
- Observing X_E,

Handling Evidence

- Observing $x = v$,
  - we can absorb it in the factor (as we did); or
  - set messages $\mu_{x \to f}(x) = 0$ for all $x \neq v$
- Observing X_E,
  - multiplying the incoming messages to $x \notin X_E$ gives the *joint* (not $p(x|X_E)$):

$$p(x, X_E) \propto \prod_{f \in ne(x)} \mu_{f \to x}(x)$$

Handling Evidence

- Observing $x = v$,
  - we can absorb it in the factor (as we did); or
  - set messages $\mu_{x \to f}(x) = 0$ for all $x \neq v$
- Observing X_E,
  - multiplying the incoming messages to $x \notin X_E$ gives the *joint* (not $p(x|X_E)$):

$$p(x, X_E) \propto \prod_{f \in ne(x)} \mu_{f \to x}(x)$$

  - The conditional is easily obtained by normalization

$$p(x|X_E) = \frac{p(x, X_E)}{\sum_{x'} p(x', X_E)}$$

Loopy Belief Propagation

- So far, we assumed a tree graph

Loopy Belief Propagation

- So far, we assumed a tree graph
- When the factor graph contains loops, pass messages indefinitely until convergence

Loopy Belief Propagation

- So far, we assumed a tree graph
- When the factor graph contains loops, pass messages indefinitely until convergence
- Loopy BP may not convergence, but “works” in many cases

Outline

Parameter Learning

- Assume the graph structure is given

Parameter Learning

- Assume the graph structure is given
- Parameters:

Parameter Learning

- Assume the graph structure is given
- Parameters:
  - θ_i in CPDs $p(x_i \mid pa(x_i), \theta_i)$ in directed graphical models

$$p(X) = \prod_i p(x_i \mid Pa(x_i), \theta_i)$$

Parameter Learning

- Assume the graph structure is given
- Parameters:
  - θ_i in CPDs $p(x_i \mid pa(x_i), \theta_i)$ in directed graphical models

$$p(X) = \prod_i p(x_i \mid Pa(x_i), \theta_i)$$

  - Weights w_i in undirected graphical model

$$p(X) = \frac{1}{Z} \exp\left(\sum_{i=1}^{k} w_i f_i(X) \right)$$

Parameter Learning

- Assume the graph structure is given
- Parameters:
  - θ_i in CPDs $p(x_i \mid pa(x_i), \theta_i)$ in directed graphical models

$$p(X) = \prod_i p(x_i \mid Pa(x_i), \theta_i)$$

  - Weights w_i in undirected graphical model

$$p(X) = \frac{1}{Z} \exp\left(\sum_{i=1}^{k} w_i f_i(X)\right)$$

- Principle: maximum likelihood estimate

Parameter Learning: Maximum Likelihood Estimate

- *fully observed*: all dimensions of X are observed

Parameter Learning: Maximum Likelihood Estimate

- *fully observed*: all dimensions of X are observed
  - given $X^1, \ldots, X^n$, the MLE is

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log p(X^i \mid \theta)$$

Parameter Learning: Maximum Likelihood Estimate

- *fully observed*: all dimensions of X are observed
  - given $X^1, \ldots, X^n$, the MLE is

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log p(X^i \mid \theta)$$

  - log likelihood factorizes for directed models (easy)

Parameter Learning: Maximum Likelihood Estimate

- *fully observed*: all dimensions of X are observed
  - given $X^1, \ldots, X^n$, the MLE is

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log p(X^i \mid \theta)$$

  - log likelihood factorizes for directed models (easy)
  - gradient method for undirected models

Parameter Learning: Maximum Likelihood Estimate

- *fully observed*: all dimensions of X are observed
  - given $X^1, \ldots, X^n$, the MLE is

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log p(X^i \mid \theta)$$

  - log likelihood factorizes for directed models (easy)
  - gradient method for undirected models
- *partially observed*: $X = (X_o, X_h)$ where X_h unobserved

Parameter Learning: Maximum Likelihood Estimate

- *fully observed*: all dimensions of X are observed
  - given $X^1, \ldots, X^n$, the MLE is

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log p(X^i \mid \theta)$$

  - log likelihood factorizes for directed models (easy)
  - gradient method for undirected models
- *partially observed*: $X = (X_o, X_h)$ where X_h unobserved
  - given $X_o^1, \ldots, X_o^n$, the MLE is

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log \left(\sum_{X_h} p(X_o^i, X_h \mid \theta) \right)$$

Parameter Learning: Maximum Likelihood Estimate

- *fully observed*: all dimensions of X are observed
  - given $X^1, \ldots, X^n$, the MLE is

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log p(X^i \mid \theta)$$

  - log likelihood factorizes for directed models (easy)
  - gradient method for undirected models
- *partially observed*: $X = (X_o, X_h)$ where X_h unobserved
  - given $X_o^1, \ldots, X_o^n$, the MLE is

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log \left(\sum_{X_h} p(X_o^i, X_h \mid \theta) \right)$$

  - log likelihood does not factorize

Parameter Learning: Maximum Likelihood Estimate

- *fully observed*: all dimensions of X are observed
  - given $X^1, \ldots, X^n$, the MLE is

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log p(X^i \mid \theta)$$

  - log likelihood factorizes for directed models (easy)
  - gradient method for undirected models
- *partially observed*: $X = (X_o, X_h)$ where X_h unobserved
  - given $X_o^1, \ldots, X_o^n$, the MLE is

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log \left(\sum_{X_h} p(X_o^i, X_h \mid \theta) \right)$$

  - log likelihood does not factorize
  - The EM algorithm finds a local maximum

Structure Learning

- Let $\mathcal{M}$ be all allowed candidate features

Structure Learning

- Let $\mathcal{M}$ be all allowed candidate features
- Let $M \subseteq \mathcal{M}$ be the "active subset"

$$P(X \mid M, \theta) = \frac{1}{Z} \exp\left(\sum_{i \in M} \theta_i f_i(X)\right)$$

Structure Learning

- Let $\mathcal{M}$ be all allowed candidate features
- Let $M \subseteq \mathcal{M}$ be the "active subset"

$$P(X \mid M, \theta) = \frac{1}{Z} \exp\left(\sum_{i \in M} \theta_i f_i(X)\right)$$

- $score(M) = \max_\theta \ln P(\text{Data} \mid M, \theta)$

Structure Learning

- Let $\mathcal{M}$ be all allowed candidate features
- Let $M \subseteq \mathcal{M}$ be the "active subset"

$$P(X \mid M, \theta) = \frac{1}{Z} \exp\left(\sum_{i \in M} \theta_i f_i(X)\right)$$

- $score(M) = \max_\theta \ln P(\text{Data} \mid M, \theta)$
- The score is always better for larger M – needs regularization or Bayesian treatment

Structure Learning

- Let $\mathcal{M}$ be all allowed candidate features
- Let $M \subseteq \mathcal{M}$ be the "active subset"

$$P(X \mid M, \theta) = \frac{1}{Z} \exp\left(\sum_{i \in M} \theta_i f_i(X)\right)$$

- $score(M) = \max_\theta \ln P(\text{Data} \mid M, \theta)$
- The score is always better for larger M – needs regularization or Bayesian treatment
- M and θ treated separately; combinatorial search over M

Structure Learning for Gaussian Random Fields

- Consider a d-dimensional multivariate Gaussian $N(\mu, \Sigma)$

Structure Learning for Gaussian Random Fields

- Consider a d-dimensional multivariate Gaussian $N(\mu, \Sigma)$
- The graphical model has p nodes $x_1, \ldots, x_d$

Structure Learning for Gaussian Random Fields

- Consider a d-dimensional multivariate Gaussian $N(\mu, \Sigma)$
- The graphical model has p nodes $x_1, \ldots, x_d$
- The edge between x_i, x_j is absent if and only if $\Omega_{ij} = 0$, where $\Omega = \Sigma^{-1}$

Structure Learning for Gaussian Random Fields

- Consider a d-dimensional multivariate Gaussian $N(\mu, \Sigma)$
- The graphical model has p nodes $x_1, \ldots, x_d$
- The edge between x_i, x_j is absent if and only if $\Omega_{ij} = 0$, where $\Omega = \Sigma^{-1}$
- Equivalently, x_i, x_j are conditionally independent given other variables

Example

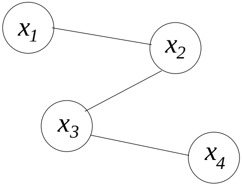
- If we know $\Sigma = \begin{pmatrix} 14 & -16 & 4 & -2 \\ -16 & 32 & -8 & 4 \\ 4 & -8 & 8 & -4 \\ -2 & 4 & -4 & 5 \end{pmatrix}$

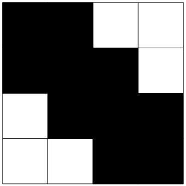
Example

- If we know $\Sigma = \begin{pmatrix} 14 & -16 & 4 & -2 \\ -16 & 32 & -8 & 4 \\ 4 & -8 & 8 & -4 \\ -2 & 4 & -4 & 5 \end{pmatrix}$
- Then $\Omega = \Sigma^{-1} = \begin{pmatrix} 0.1667 & 0.0833 & 0.0000 & 0 \\ 0.0833 & 0.0833 & 0.0417 & 0 \\ 0.0000 & 0.0417 & 0.2500 & 0.1667 \\ 0 & 0 & 0.1667 & 0.3333 \end{pmatrix}$

Example

- If we know $\Sigma = \begin{pmatrix} 14 & -16 & 4 & -2 \\ -16 & 32 & -8 & 4 \\ 4 & -8 & 8 & -4 \\ -2 & 4 & -4 & 5 \end{pmatrix}$
- Then $\Omega = \Sigma^{-1} = \begin{pmatrix} 0.1667 & 0.0833 & 0.0000 & 0 \\ 0.0833 & 0.0833 & 0.0417 & 0 \\ 0.0000 & 0.0417 & 0.2500 & 0.1667 \\ 0 & 0 & 0.1667 & 0.3333 \end{pmatrix}$
- The corresponding graphical model structure is





Structure Learning for Gaussian Random Fields

- Let data be $X^{(1)}, \ldots, X^{(n)} \sim N(\mu, \Sigma)$

Structure Learning for Gaussian Random Fields

- Let data be $X^{(1)}, \ldots, X^{(n)} \sim N(\mu, \Sigma)$
- The log likelihood is $\frac{n}{2} \log |\Omega| - \frac{1}{2} \sum_{i=1}^{n} (X^{(i)} - \mu)^\top \Omega (X^{(i)} - \mu)$

Structure Learning for Gaussian Random Fields

- Let data be $X^{(1)}, \ldots, X^{(n)} \sim N(\mu, \Sigma)$
- The log likelihood is $\frac{n}{2}\log|\Omega| - \frac{1}{2}\sum_{i=1}^{n}(X^{(i)} - \mu)^\top \Omega (X^{(i)} - \mu)$
- The maximum likelihood estimate of Σ is the sample covariance

$$S = \frac{1}{n}\sum_i (X^{(i)} - \bar{X})^\top (X^{(i)} - \bar{X})$$

where $\bar{X}$ is the sample mean

Structure Learning for Gaussian Random Fields

- Let data be $X^{(1)}, \ldots, X^{(n)} \sim N(\mu, \Sigma)$
- The log likelihood is $\frac{n}{2} \log |\Omega| - \frac{1}{2} \sum_{i=1}^{n} (X^{(i)} - \mu)^\top \Omega (X^{(i)} - \mu)$
- The maximum likelihood estimate of Σ is the sample covariance

  $$S = \frac{1}{n} \sum_i (X^{(i)} - \bar{X})^\top (X^{(i)} - \bar{X})$$

  where $\bar{X}$ is the sample mean
- S^{-1} is not a good estimate of Ω when n is small

Structure Learning for Gaussian Random Fields

- For centered data, minimize a regularized problem instead:

$$-\log|\Omega| + \frac{1}{n}\sum_{i=1}^{n} {X^{(i)}}^\top \Omega X^{(i)} + \lambda \sum_{i\neq j} |\Omega_{ij}|$$

Structure Learning for Gaussian Random Fields

- For centered data, minimize a regularized problem instead:

$$-\log |\Omega| + \frac{1}{n} \sum_{i=1}^{n} {X^{(i)}}^\top \Omega X^{(i)} + \lambda \sum_{i \neq j} |\Omega_{ij}|$$

- Known as GLASSO

Outline

Kernel methods

- Traditionally, an item x is a feature vector in $\mathbb{R}^d$

Kernel methods

- Traditionally, an item x is a feature vector in $\mathbb{R}^d$
  - Feature engineering decides what the features are

Kernel methods

- Traditionally, an item x is a feature vector in $\mathbb{R}^d$
  - Feature engineering decides what the features are
  - Learning algorithms work on $x_1, \ldots, x_n \in \mathbb{R}^d$ directly

Kernel methods

- Traditionally, an item x is a feature vector in $\mathbb{R}^d$
  - Feature engineering decides what the features are
  - Learning algorithms work on $x_1, \ldots, x_n \in \mathbb{R}^d$ directly
- Many algorithms actually only use inner products $x_i^\top x_j$

Kernel methods

- Traditionally, an item x is a feature vector in $\mathbb{R}^d$
  - Feature engineering decides what the features are
  - Learning algorithms work on $x_1, \ldots, x_n \in \mathbb{R}^d$ directly
- Many algorithms actually only use inner products $x_i^\top x_j$
  - Data fully defined by $n \times n$ matrix K where $K_{ij} = x_i^\top x_j$

Kernel methods

- Traditionally, an item x is a feature vector in $\mathbb{R}^d$
  - Feature engineering decides what the features are
  - Learning algorithms work on $x_1, \ldots, x_n \in \mathbb{R}^d$ directly
- Many algorithms actually only use inner products $x_i^\top x_j$
  - Data fully defined by $n \times n$ matrix K where $K_{ij} = x_i^\top x_j$
  - We can just give K to these algorithms

Kernel methods

- Traditionally, an item x is a feature vector in $\mathbb{R}^d$
  - Feature engineering decides what the features are
  - Learning algorithms work on $x_1, \ldots, x_n \in \mathbb{R}^d$ directly
- Many algorithms actually only use inner products $x_i^\top x_j$
  - Data fully defined by $n \times n$ matrix K where $K_{ij} = x_i^\top x_j$
  - We can just give K to these algorithms
- What if we give any matrix K' to these algorithms?

Kernel methods

- Traditionally, an item x is a feature vector in $\mathbb{R}^d$
  - Feature engineering decides what the features are
  - Learning algorithms work on $x_1, \ldots, x_n \in \mathbb{R}^d$ directly
- Many algorithms actually only use inner products $x_i^\top x_j$
  - Data fully defined by $n \times n$ matrix K where $K_{ij} = x_i^\top x_j$
  - We can just give K to these algorithms
- What if we give any matrix K' to these algorithms?
  - They work if K' is positive semi-definition (kernel matrix)

Kernel methods

- Traditionally, an item x is a feature vector in $\mathbb{R}^d$
  - Feature engineering decides what the features are
  - Learning algorithms work on $x_1, \ldots, x_n \in \mathbb{R}^d$ directly
- Many algorithms actually only use inner products $x_i^\top x_j$
  - Data fully defined by $n \times n$ matrix K where $K_{ij} = x_i^\top x_j$
  - We can just give K to these algorithms
- What if we give any matrix K' to these algorithms?
  - They work if K' is positive semi-definition (kernel matrix)
  - There are feature vectors $\phi(x) \in \mathbb{R}^D$ such that $K'_{ij} = \phi(x_i)^\top \phi(x_j)$

Kernel methods

- Traditionally, an item x is a feature vector in $\mathbb{R}^d$
  - Feature engineering decides what the features are
  - Learning algorithms work on $x_1, \ldots, x_n \in \mathbb{R}^d$ directly
- Many algorithms actually only use inner products $x_i^\top x_j$
  - Data fully defined by $n \times n$ matrix K where $K_{ij} = x_i^\top x_j$
  - We can just give K to these algorithms
- What if we give any matrix K' to these algorithms?
  - They work if K' is positive semi-definition (kernel matrix)
  - There are feature vectors $\phi(x) \in \mathbb{R}^D$ such that $K'_{ij} = \phi(x_i)^\top \phi(x_j)$
  - $\phi(x)$ implicit feature engineering

Kernel methods

- Traditionally, an item x is a feature vector in $\mathbb{R}^d$
  - Feature engineering decides what the features are
  - Learning algorithms work on $x_1, \ldots, x_n \in \mathbb{R}^d$ directly
- Many algorithms actually only use inner products $x_i^\top x_j$
  - Data fully defined by $n \times n$ matrix K where $K_{ij} = x_i^\top x_j$
  - We can just give K to these algorithms
- What if we give any matrix K' to these algorithms?
  - They work if K' is positive semi-definition (kernel matrix)
  - There are feature vectors $\phi(x) \in \mathbb{R}^D$ such that $K'_{ij} = \phi(x_i)^\top \phi(x_j)$
  - $\phi(x)$ implicit feature engineering
- Precise definition: Reproducing Kernel Hilbert Space (RKHS)

Outline

The Linearly Separable Case

- $x \in R^d$, $y \in \{-1, 1\}$

The Linearly Separable Case

- $x \in R^d$, $y \in \{-1, 1\}$
- discriminant function $f(x) = w^\top x + b$

The Linearly Separable Case

- $x \in R^d$, $y \in \{-1, 1\}$
- discriminant function $f(x) = w^\top x + b$
- classification rule sign$(f(x))$

The Linearly Separable Case

- $x \in R^d$, $y \in \{-1, 1\}$
- discriminant function $f(x) = w^\top x + b$
- classification rule sign$(f(x))$
- linear decision boundary $\{x \in \mathbb{R}^d \mid f(x) = 0\}$ orthogonal to w

The Linearly Separable Case

- Distance between a correctly classified x and the decision boundary:
$$\frac{yf(x)}{\|w\|}$$

The Linearly Separable Case

- Training task: given $\{(x, y)_{1:n}\}$, find a decision boundary w, b to maximize the distance to the closest point

$$\max_{w,b} \min_{i=1}^{n} \frac{y_i(w^\top x_i + b)}{\|w\|}$$

The Linearly Separable Case

- Equivalently,

$$\max_{w,b} \quad \frac{1}{\|w\|}$$
$$s.t. \quad y_i(w^\top x_i + b) \geq 1 \;\; i = 1 \dots n$$

The Linearly Separable Case

- Equivalently,

$$\begin{aligned} \min_{w,b} \quad & \tfrac{1}{2}\|w\|^2 \\ s.t. \quad & y_i(w^\top x_i + b) \geq 1 \quad i = 1 \dots n \end{aligned}$$

The Linearly Separable Case

- Equivalently,

$$\begin{aligned} \min_{w,b} \quad & \tfrac{1}{2}\|w\|^2 \\ s.t. \quad & y_i(w^\top x_i + b) \geq 1 \;\; i = 1 \dots n \end{aligned}$$

- Primal problem, uses feature vectors $x_i \in \mathbb{R}^d$

The Linearly Separable Case

- Equivalently,

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2$$
$$s.t. \quad y_i(w^\top x_i + b) \geq 1 \;\; i = 1 \ldots n$$

- Primal problem, uses feature vectors $x_i \in \mathbb{R}^d$
- The equivalent dual problem will involve only inner products $x_i^\top x_j$

The Linearly Separable Case

- The dual problem

$$\begin{array}{ll} \max\limits_{\alpha} & -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{n} \alpha_i \\ s.t. & \alpha_i \geq 0 \;\; i = 1 \ldots n \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \end{array}$$

The Linearly Separable Case

- The dual problem

$$\begin{array}{cc} \max\limits_{\alpha} & -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{n}\alpha_i \\ s.t. & \alpha_i \geq 0 \;\; i = 1\ldots n \\ & \sum_{i=1}^{n}\alpha_i y_i = 0 \end{array}$$

- $d + 1$ primal variables w, b

The Linearly Separable Case

- The dual problem

$$\begin{array}{ll} \max\limits_{\alpha} & -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{n}\alpha_i \\ s.t. & \alpha_i \geq 0 \;\; i = 1\ldots n \\ & \sum_{i=1}^{n}\alpha_i y_i = 0 \end{array}$$

- $d+1$ primal variables w, b
- n dual variables α (interesting when $d \gg n$)

The Linearly Separable Case

To classify a test point x

- primal discriminant function $f(x) = w^\top x + b$

The Linearly Separable Case

To classify a test point x

- primal discriminant function $f(x) = w^\top x + b$
- dual discriminant function $f(x) = \sum_{i=1}^{n} \alpha_i y_i x_i^\top x + b$

The Linearly Separable Case

To classify a test point x

- primal discriminant function $f(x) = w^\top x + b$
- dual discriminant function $f(x) = \sum_{i=1}^{n} \alpha_i y_i x_i^\top x + b$
- another inner-product

Support vectors

- The Karush-Kuhn-Tucker complementarity condition:
  $\alpha_i(y_i(w^\top x_i + b) - 1) = 0, \; i = 1 \dots n$

Support vectors

- The Karush-Kuhn-Tucker complementarity condition: $\alpha_i(y_i(w^\top x_i + b) - 1) = 0,\ i = 1 \dots n$
- $y_i(w^\top x_i + b) - 1 > 0$ (x_i outside the margin) $\Rightarrow \alpha_i = 0$ (x_i not support vector)

Support vectors

- The Karush-Kuhn-Tucker complementarity condition: $\alpha_i(y_i(w^\top x_i + b) - 1) = 0,\ i = 1 \dots n$
- $y_i(w^\top x_i + b) - 1 > 0$ (x_i outside the margin) $\Rightarrow \alpha_i = 0$ (x_i not support vector)
- $\alpha_i \neq 0$ (x_i is support vector) $\Rightarrow y_i(w^\top x_i + b) = 1$ (x_i on the margin)

The Non-Separable Case

- Relax margin constraints

$$y_i(w^\top x_i + b) \geq 1 - \xi_i$$

The Non-Separable Case

- Relax margin constraints

$$y_i(w^\top x_i + b) \geq 1 - \xi_i$$

- Slack variables $\xi_i \geq 0$

The Non-Separable Case

- Relax margin constraints

$$y_i(w^\top x_i + b) \geq 1 - \xi_i$$

- Slack variables $\xi_i \geq 0$
- Large enough ξ_i allows x_i on the wrong side of the decision boundary

The Non-Separable Case

- Primal problem

$$\begin{aligned}
\min_{w,b,\xi} \quad & \tfrac{1}{2}\|w\|^2 + C\textstyle\sum_{i=1}^n \xi_i \\
s.t. \quad & y_i(w^\top x_i + b) \geq 1 - \xi_i \;\; i = 1 \ldots n \\
& \xi_i \geq 0
\end{aligned}$$

The Non-Separable Case

- Dual problem

$$
\begin{array}{ll}
\max\limits_{\alpha} & -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{n}\alpha_i \\
s.t. & 0 \leq \alpha_i \leq C \;\; i = 1 \ldots n \\
 & \sum_{i=1}^{n}\alpha_i y_i = 0
\end{array}
$$

The Non-Separable Case

- Dual problem

$$\begin{array}{cc} \max\limits_{\alpha} & -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{n}\alpha_i \\ s.t. & 0 \le \alpha_i \le C \;\; i = 1 \ldots n \\ & \sum_{i=1}^{n}\alpha_i y_i = 0 \end{array}$$

- Again, data enter optimization as inner products

The Non-Separable Case

- Dual problem

$$\begin{array}{ll} \max\limits_{\alpha} & -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{n}\alpha_i \\ s.t. & 0 \le \alpha_i \le C \;\; i = 1 \dots n \\ & \sum_{i=1}^{n}\alpha_i y_i = 0 \end{array}$$

- Again, data enter optimization as inner products
- Support vectors:

The Non-Separable Case

- Dual problem

$$\begin{array}{ll} \max\limits_{\alpha} & -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{n}\alpha_i \\ s.t. & 0 \leq \alpha_i \leq C \quad i = 1\ldots n \\ & \sum_{i=1}^{n}\alpha_i y_i = 0 \end{array}$$

- Again, data enter optimization as inner products
- Support vectors:
  - $\alpha_i = 0 \Rightarrow x_i$ not a support vector

The Non-Separable Case

- Dual problem

$$\begin{array}{ll} \max\limits_{\alpha} & -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{n}\alpha_i \\ s.t. & 0 \leq \alpha_i \leq C \;\; i = 1\ldots n \\ & \sum_{i=1}^{n}\alpha_i y_i = 0 \end{array}$$

- Again, data enter optimization as inner products
- Support vectors:
  - $\alpha_i = 0 \Rightarrow x_i$ not a support vector
  - $0 < \alpha_i < C \Rightarrow \xi = 0$, support vector x_i on the margin

The Non-Separable Case

- Dual problem

$$
\begin{array}{cc}
\max\limits_{\alpha} & -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{n}\alpha_i \\
s.t. & 0 \le \alpha_i \le C \quad i = 1\dots n \\
 & \sum_{i=1}^{n}\alpha_i y_i = 0
\end{array}
$$

- Again, data enter optimization as inner products
- Support vectors:
  - $\alpha_i = 0 \Rightarrow x_i$ not a support vector
  - $0 < \alpha_i < C \Rightarrow \xi = 0$, support vector x_i on the margin
  - $\alpha = C \Rightarrow x_i$ inside the margin if $\xi \le 1$, or on the wrong side of the decision boundary if $\xi > 1$

The Non-Separable Case

- The discriminant function is

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i x_i^\top x + b$$

The Non-Separable Case

- The discriminant function is

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i x_i^\top x + b$$

- Inner product again

The Kernel Trick

- SVM dual problem only involves inner products $x_i^\top x_j$

The Kernel Trick

- SVM dual problem only involves inner products $x_i^\top x_j$
- Let $K(x_i, x_j) = x_i^\top x_j$

The Kernel Trick

- SVM dual problem only involves inner products $x_i^\top x_j$
- Let $K(x_i, x_j) = x_i^\top x_j$
- Replace $x_i^\top x_j$ with $K(x_i, x_j)$ everywhere

The Kernel Trick

- SVM dual problem only involves inner products $x_i^\top x_j$
- Let $K(x_i, x_j) = x_i^\top x_j$
- Replace $x_i^\top x_j$ with $K(x_i, x_j)$ everywhere
- Tautology

The Kernel Trick

- Instead of $K(x_i, x_j) = x_i^\top x_j$, let K be any positive definite function

The Kernel Trick

- Instead of $K(x_i, x_j) = x_i^\top x_j$, let K be any positive definite function
- K p.d. if $\forall n, \forall x_1 \ldots x_n$ the matrix

$$K_n = \begin{bmatrix} K(x_1, x_1) & \ldots & K(x_1, x_n) \\ & \vdots & \\ K(x_n, x_1) & \ldots & K(x_n, x_n) \end{bmatrix}$$

is positive semi-definite.

The Kernel Trick

- Instead of $K(x_i, x_j) = x_i^\top x_j$, let K be any positive definite function
- K p.d. if $\forall n, \forall x_1 \dots x_n$ the matrix

$$K_n = \begin{bmatrix} K(x_1, x_1) & \dots & K(x_1, x_n) \\ & \vdots & \\ K(x_n, x_1) & \dots & K(x_n, x_n) \end{bmatrix}$$

  is positive semi-definite.
- K_n positive semi-definite if $\forall \mathbf{z} = (z_1, \dots, z_n)^\top \in \mathbb{R}^n$,

$$\mathbf{z}^\top K_n \mathbf{z} \geq 0$$

The Kernel Trick

P.d. K examples:

- Linear kernel

$$k(x_i, x_j) = x_i^\top x_j$$

The Kernel Trick

P.d. K examples:

- Linear kernel

$$k(x_i, x_j) = x_i^\top x_j$$

- Polynomial kernel

$$k(x_i, x_j) = (1 + x_i^\top x_j)^p$$

The Kernel Trick

P.d. K examples:

- Linear kernel

$$k(x_i, x_j) = x_i^\top x_j$$

- Polynomial kernel

$$k(x_i, x_j) = (1 + x_i^\top x_j)^p$$

- Radial Basis Function (RBF) kernel

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

The Kernel Trick

- SVM dual problem can use any p.d. K (kernelize)

The Kernel Trick

- SVM dual problem can use any p.d. K (kernelize)
- There exists a feature mapping $\phi()$ such that $K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$

The Kernel Trick

- SVM dual problem can use any p.d. K (kernelize)
- There exists a feature mapping $\phi()$ such that $K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$
  - $\phi()$ may not be finite dimensional

The Kernel Trick

- SVM dual problem can use any p.d. K (kernelize)
- There exists a feature mapping $\phi()$ such that $K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$
  - $\phi()$ may not be finite dimensional
  - $\phi()$ may not be unique

The Kernel Trick

- SVM dual problem can use any p.d. K (kernelize)
- There exists a feature mapping $\phi()$ such that $K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$
  - $\phi()$ may not be finite dimensional
  - $\phi()$ may not be unique
- What does the kernel trick buy us?

The Kernel Trick

- $x_1 = -1(+), x_2 = 0(-), x_3 = 1(+)$

The Kernel Trick

- $x_1 = -1(+), x_2 = 0(-), x_3 = 1(+)$
- Not a linearly separable dataset

The Kernel Trick

- $x_1 = -1(+), x_2 = 0(-), x_3 = 1(+)$
- Not a linearly separable dataset
- But we can map x to $\mathbb{R}^3$

$$\phi(x) = (1, \sqrt{2}x, x^2)^\top$$

and separate them with a hyperplane

The Kernel Trick

- $x_1 = -1(+), x_2 = 0(-), x_3 = 1(+)$
- Not a linearly separable dataset
- But we can map x to $\mathbb{R}^3$

$$\phi(x) = (1, \sqrt{2}x, x^2)^\top$$

  and separate them with a hyperplane
- Non-linear decision boundary in the original space

The Kernel Trick

- $x_1 = -1(+), x_2 = 0(-), x_3 = 1(+)$
- Not a linearly separable dataset
- But we can map x to $\mathbb{R}^3$

$$\phi(x) = (1, \sqrt{2}x, x^2)^\top$$

  and separate them with a hyperplane
- Non-linear decision boundary in the original space
- Equivalently, we used a kernel

$$K(x_i, x_j) = \phi(x_i)^\top \phi(x_j) = (1 + x_i x_j)^2$$

  in *linear* SVM without slack variables.

Outline

The Kernel Trick is not just for SVMs

Summary of the kernel trick:

- data as inner products

The Kernel Trick is not just for SVMs

Summary of the kernel trick:

- data as inner products
- p.d. K kernel

The Kernel Trick is not just for SVMs

Summary of the kernel trick:

- data as inner products
- p.d. K kernel
- induced feature map $\phi()$ such that $K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$

The Kernel Trick is not just for SVMs

Summary of the kernel trick:

- data as inner products
- p.d. K kernel
- induced feature map $\phi()$ such that $K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$
- choosing the kernel K equivalent to feature engineering

The Kernel Trick is not just for SVMs

Summary of the kernel trick:

- data as inner products
- p.d. K kernel
- induced feature map $\phi()$ such that $K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$
- choosing the kernel K equivalent to feature engineering
- many algorithms can be kernelized

Principal Component Analysis (PCA)

- Unsupervised learning

Principal Component Analysis (PCA)

- Unsupervised learning
- Given $x_1 \ldots x_n \in \mathbb{R}^d$, finds directions of maximum spread

Principal Component Analysis (PCA)

- Unsupervised learning
- Given $x_1 \dots x_n \in \mathbb{R}^d$, finds directions of maximum spread
- Centering data:

$$x_i \leftarrow x_i - \bar{x}$$

  where $\bar{x} = \frac{1}{n} \sum_j x_j$

Principal Component Analysis (PCA)

- Unsupervised learning
- Given $x_1 \dots x_n \in \mathbb{R}^d$, finds directions of maximum spread
- Centering data:

$$x_i \leftarrow x_i - \bar{x}$$

  where $\bar{x} = \frac{1}{n} \sum_j x_j$
- $d \times d$ sample covariance matrix

$$C = \frac{1}{n} \sum_i x_i x_i^\top$$

PCA

- Eigendecomposition

$$C = U\Lambda U^\top = \sum_{j=1}^{d} \lambda_j u_j u_j^\top$$

PCA

- Eigendecomposition

$$C = U\Lambda U^\top = \sum_{j=1}^{d} \lambda_j u_j u_j^\top$$

- Eigenvalues $\lambda_1 \geq \ldots \geq \lambda_d \geq 0$ the variances

PCA

- Eigendecomposition

$$C = U \Lambda U^\top = \sum_{j=1}^{d} \lambda_j u_j u_j^\top$$

- Eigenvalues $\lambda_1 \geq \ldots \geq \lambda_d \geq 0$ the variances
- Eigenvectors $u_1 \ldots u_d$ the principal components with decreasing importance

$$C u_j = \lambda_j u_j, \quad j = 1 \ldots d$$

PCA

- Eigendecomposition

$$C = U\Lambda U^\top = \sum_{j=1}^{d} \lambda_j u_j u_j^\top$$

- Eigenvalues $\lambda_1 \geq \ldots \geq \lambda_d \geq 0$ the variances
- Eigenvectors $u_1 \ldots u_d$ the principal components with decreasing importance

$$Cu_j = \lambda_j u_j, \quad j = 1 \ldots d$$

- $u_1 \ldots u_d$ orthonormal basis of $\mathbb{R}^d$, rotated axes

PCA

- Dimension reduction: project to the top $k \leq d$ directions

PCA

- Dimension reduction: project to the top $k \leq d$ directions
- U_k the first k columns of $U = [u_1 \mid u_2 \mid \dots \mid u_d]$

PCA

- Dimension reduction: project to the top $k \le d$ directions
- U_k the first k columns of $U = [u_1 \mid u_2 \mid \ldots \mid u_d]$
- $x \in \mathbb{R}^d$ projected to $\mathbb{R}^k$ by

$$U_k^\top x = \begin{bmatrix} u_1^\top x \\ \vdots \\ u_k^\top x \end{bmatrix}$$

PCA

- Dimension reduction: project to the top $k \leq d$ directions
- U_k the first k columns of $U = [u_1 \mid u_2 \mid \dots \mid u_d]$
- $x \in \mathbb{R}^d$ projected to $\mathbb{R}^k$ by

$$U_k^\top x = \begin{bmatrix} u_1^\top x \\ \vdots \\ u_k^\top x \end{bmatrix}$$

- U_k minimizes training set ℓ_2-error among rank-k projections

$$\sum_{i=1}^{n} \|x_i - U_k^\top x_i\|_2^2$$

PCA

- Dimension reduction: project to the top $k \leq d$ directions
- U_k the first k columns of $U = [u_1 \mid u_2 \mid \dots \mid u_d]$
- $x \in \mathbb{R}^d$ projected to $\mathbb{R}^k$ by

$$U_k^\top x = \begin{bmatrix} u_1^\top x \\ \vdots \\ u_k^\top x \end{bmatrix}$$

- U_k minimizes training set ℓ_2-error among rank-k projections

$$\sum_{i=1}^{n} \|x_i - U_k^\top x_i\|_2^2$$

- So far PCA with feature vectors in $\mathbb{R}^d$. Next: PCA with inner products

PCA with inner products

- For $j = 1 \dots d$

$$\begin{aligned} Cu_j &= \lambda_j u_j \\ \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top u_j &= \lambda_j u_j \\ \sum_{i=1}^{n} \frac{(x_i^\top u_j)}{n\lambda_j} x_i &= u_j \end{aligned}$$

PCA with inner products

- For $j = 1 \ldots d$

$$\begin{aligned} Cu_j &= \lambda_j u_j \\ \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top u_j &= \lambda_j u_j \\ \sum_{i=1}^{n} \frac{(x_i^\top u_j)}{n\lambda_j} x_i &= u_j \end{aligned}$$

- Any u_j can be written in the form

$$u_j = \sum_{i=1}^{n} \alpha_{ji} x_i$$

PCA with inner products

- For $j = 1 \dots d$

$$\begin{aligned} Cu_j &= \lambda_j u_j \\ \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top u_j &= \lambda_j u_j \\ \sum_{i=1}^{n} \frac{(x_i^\top u_j)}{n\lambda_j} x_i &= u_j \end{aligned}$$

- Any u_j can be written in the form

$$u_j = \sum_{i=1}^{n} \alpha_{ji} x_i$$

- $\alpha_{ji} \in \mathbb{R}$, value not obvious (involving u_j)

PCA with inner products

- $n \times n$ matrix K with $K_{ij} = x_i^\top x_j$

PCA with inner products

- $n \times n$ matrix K with $K_{ij} = x_i^\top x_j$
- $\alpha_j = (\alpha_{j1}, \ldots, \alpha_{jn})^\top$ satisfy the eigenvalue equation

$$K\alpha_j = n\lambda_j\alpha_j$$

Why?

$$
\begin{aligned}
Cu_j &= \lambda_j u_j \\
x_i^\top C u_j &= x_i^\top \lambda_j u_j, \quad i = 1 \ldots n \\
x_i^\top \left(\frac{1}{n} \sum_{k=1}^{n} x_k x_k^\top \right) \left(\sum_{m=1}^{n} \alpha_{jm} x_m \right) &= x_i^\top \lambda_j \sum_{m=1}^{n} \alpha_{jm} x_m \\
\frac{1}{n} \sum_{k=1}^{n} \sum_{m=1}^{n} \alpha_{jm} x_i^\top x_k x_k^\top x_m &= \sum_{m=1}^{n} \lambda_j \alpha_{jm} x_i^\top x_m \\
\frac{1}{n} \sum_{k=1}^{n} \sum_{m=1}^{n} \alpha_{jm} K_{ik} K_{km} &= \sum_{m=1}^{n} \lambda_j \alpha_{jm} K_{im} \\
\frac{1}{n} K_{i.} K \alpha_j &= \lambda_j K_{i.} \alpha_j, \quad i = 1 \ldots n \\
\frac{1}{n} K K \alpha_j &= \lambda_j K \alpha_j \\
K \alpha_j &= n \lambda_j \alpha_j
\end{aligned}
$$

assuming $n \leq d$ and K invertible

PCA with inner products

- $\alpha_j = (\alpha_{j1}, \ldots, \alpha_{jn})^\top$ satisfy the eigenvalue equation

$$K\alpha_j = n\lambda_j\alpha_j$$

PCA with inner products

- $\alpha_j = (\alpha_{j1}, \ldots, \alpha_{jn})^\top$ satisfy the eigenvalue equation
$$K\alpha_j = n\lambda_j \alpha_j$$
- Norm of α_j is also fixed:
$$\begin{aligned}
\|u_j\| &= 1 \\
u_j^\top u_j &= 1 \\
\sum_{k,m=1}^{n} \alpha_{jk} x_k^\top x_m \alpha_{jm} &= 1 \\
\sum_{k,m=1}^{n} \alpha_{jk} K_{km} \alpha_{jm} &= 1 \\
\alpha_j^\top K \alpha_j &= 1 \\
\alpha_j^\top n\lambda_j \alpha_j &= 1 \\
\|\alpha_j\| &= \sqrt{\frac{1}{n\lambda_j}}
\end{aligned}$$

PCA with inner products

- Compute $\alpha_1, \ldots, \alpha_k$ by solving the eigenvalue equation (k largest eigenvalues)

PCA with inner products

- Compute $\alpha_1, \ldots, \alpha_k$ by solving the eigenvalue equation (k largest eigenvalues)
- Project (new) point x to top $k \leq n$ directions

$$\begin{bmatrix} u_1^\top x \\ \vdots \\ u_k^\top x \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n \alpha_{1i} x_i^\top x \\ \vdots \\ \sum_{i=1}^n \alpha_{ki} x_i^\top x \end{bmatrix} = \begin{bmatrix} \alpha_1^\top K_x \\ \vdots \\ \alpha_k^\top K_x \end{bmatrix}$$

where $K_x = (K(x_1, x), \ldots, K(x_n, x))^\top$ and $K(x_i, x) = x_i^\top x$

Kernel PCA

Perhaps replacing $K_{ij} = x_i^\top x_j$ with any kernel $K(x_i, x_j)$?

- Equivalently, we are doing standard PCA in $\phi(x)$ space

Kernel PCA

Perhaps replacing $K_{ij} = x_i^\top x_j$ with any kernel $K(x_i, x_j)$?

- Equivalently, we are doing standard PCA in $\phi(x)$ space
- But... is the training set centered $\sum_{i=1}^n \phi(x_i) = 0$?

Kernel PCA

Perhaps replacing $K_{ij} = x_i^\top x_j$ with any kernel $K(x_i, x_j)$?

- Equivalently, we are doing standard PCA in $\phi(x)$ space
- But... is the training set centered $\sum_{i=1}^n \phi(x_i) = 0$?
- Need to center K

Centering the kernel for training

$$
\begin{aligned}
\phi'(x_i) &= \phi(x_i) - \frac{1}{n}\sum_{k=1}^{n}\phi(x_k) \\
\phi'(x_i)^\top \phi'(x_j) &= \left(\phi(x_i) - \frac{1}{n}\sum_{k=1}^{n}\phi(x_k)\right)^\top \left(\phi(x_j) - \frac{1}{n}\sum_{k=1}^{n}\phi(x_k)\right) \\
K'_{ij} &= K_{ij} - \frac{1}{n}\sum_{k=1}^{n} K_{jk} - \frac{1}{n}\sum_{k=1}^{n} K_{ik} + \frac{1}{n^2}\sum_{k,m=1}^{n} K_{km}
\end{aligned}
$$

Finding α_j by solving the eigenvalue problem

$$
K'\alpha_j = n\lambda_j\alpha_j
$$

Projecting (new) point x with centering

- New point x needs to be centered $\phi'(x) = \phi(x) - \sum_{i=1}^{n} \phi(x_i)$

Projecting (new) point x with centering

- New point x needs to be centered $\phi'(x) = \phi(x) - \sum_{i=1}^{n} \phi(x_i)$
- Note x not involved in computing the training set mean

Projecting (new) point x with centering

- New point x needs to be centered $\phi'(x) = \phi(x) - \sum_{i=1}^{n} \phi(x_i)$
- Note x not involved in computing the training set mean
- Recall j-th projection is $\alpha_j^\top K'_x$

Projecting (new) point x with centering

- New point x needs to be centered $\phi'(x) = \phi(x) - \sum_{i=1}^{n} \phi(x_i)$
- Note x not involved in computing the training set mean
- Recall j-th projection is $\alpha_j^\top K'_x$
- $K'_x = (K'(x_1, x), \ldots, K'(x_n, x))^\top$

Projecting (new) point x with centering

- New point x needs to be centered $\phi'(x) = \phi(x) - \sum_{i=1}^{n} \phi(x_i)$
- Note x not involved in computing the training set mean
- Recall j-th projection is $\alpha_j^\top K'_x$
- $K'_x = (K'(x_1, x), \ldots, K'(x_n, x))^\top$

Projecting (new) point x with centering

- New point x needs to be centered $\phi'(x) = \phi(x) - \sum_{i=1}^{n} \phi(x_i)$
- Note x not involved in computing the training set mean
- Recall j-th projection is $\alpha_j^\top K'_x$
- $K'_x = (K'(x_1, x), \ldots, K'(x_n, x))^\top$

$$K'(x_i, x) = K(x_i, x) - \frac{1}{n} \sum_{k=1}^{n} K(x_k, x) - \frac{1}{n} \sum_{k=1}^{n} K_{ik} + \frac{1}{n^2} \sum_{k,m=1}^{n} K_{km}$$

Outline

Norm

Let $\mathcal{F}$ be a vector space over $\mathbb{R}$. A function $\|\cdot\|_{\mathcal{F}} : \mathcal{F} \mapsto \mathbb{R}_{\geq 0}$ is a norm if

- $\|f\|_{\mathcal{F}} = 0$ iff $f = 0$ (separation)

Norm

Let $\mathcal{F}$ be a vector space over $\mathbb{R}$. A function $\|\cdot\|_{\mathcal{F}} : \mathcal{F} \mapsto \mathbb{R}_{\geq 0}$ is a norm if

- $\|f\|_{\mathcal{F}} = 0$ iff $f = 0$ (separation)
- $\|af\|_{\mathcal{F}} = |a|\|f\|_{\mathcal{F}}$ (positive homogeneity)

Norm

Let $\mathcal{F}$ be a vector space over $\mathbb{R}$. A function $\|\cdot\|_{\mathcal{F}} : \mathcal{F} \mapsto \mathbb{R}_{\geq 0}$ is a norm if

- $\|f\|_{\mathcal{F}} = 0$ iff $f = 0$ (separation)
- $\|af\|_{\mathcal{F}} = |a|\|f\|_{\mathcal{F}}$ (positive homogeneity)
- $\|f+g\|_{\mathcal{F}} \leq \|f\|_{\mathcal{F}} + \|g\|_{\mathcal{F}}$ (triangle inequality)

Norm

Example

- Let μ be a positive measure on $\mathcal{X} \subset \mathbb{R}^d$ and $p \geq 1$

Norm

Example

- Let μ be a positive measure on $\mathcal{X} \subset \mathbb{R}^d$ and $p \geq 1$
- Let $L_p(\mathcal{X}, \mu) = \left\{ f : \mathcal{X} \mapsto \mathbb{R} \text{ measurable} \mid \int_{\mathcal{X}} |f(x)|^p d\mu < \infty \right\}$

Norm

Example

- Let μ be a positive measure on $\mathcal{X} \subset \mathbb{R}^d$ and $p \geq 1$
- Let $L_p(\mathcal{X}, \mu) = \left\{ f : \mathcal{X} \mapsto \mathbb{R} \text{ measurable} \mid \int_{\mathcal{X}} |f(x)|^p d\mu < \infty \right\}$
- $\|f\|_p = \left(\int_{\mathcal{X}} |f(x)|^p d\mu \right)^{\frac{1}{p}}$ is a norm

Cauchy sequence

A sequence $\{f_n\}_{n=1}^{\infty}$ of elements of a normed vector space $(\mathcal{F}, \|\cdot\|_{\mathcal{F}})$ is a Cauchy sequence if:

- $\forall \epsilon > 0, \exists N$

Cauchy sequence

A sequence $\{f_n\}_{n=1}^{\infty}$ of elements of a normed vector space $(\mathcal{F}, \|\cdot\|_{\mathcal{F}})$ is a Cauchy sequence if:

- $\forall \epsilon > 0, \exists N$
- $\forall n, m \geq N, \|f_n - f_m\|_{\mathcal{F}} < \epsilon$

Convergent sequence

A sequence $\{f_n\}_{n=1}^{\infty}$ of elements of a normed vector space $(\mathcal{F}, \|\cdot\|_{\mathcal{F}})$ converges to $f \in \mathcal{F}$ if:

- $\forall \epsilon > 0, \exists N$

Convergent sequence

A sequence $\{f_n\}_{n=1}^{\infty}$ of elements of a normed vector space $(\mathcal{F}, \|\cdot\|_{\mathcal{F}})$ converges to $f \in \mathcal{F}$ if:

- $\forall \epsilon > 0, \exists N$
- $\forall n \geq N, \|f_n - f\|_{\mathcal{F}} < \epsilon$

Convergent sequence

A sequence $\{f_n\}_{n=1}^{\infty}$ of elements of a normed vector space $(\mathcal{F}, \|\cdot\|_{\mathcal{F}})$ converges to $f \in \mathcal{F}$ if:

- $\forall \epsilon > 0, \exists N$
- $\forall n \geq N, \|f_n - f\|_{\mathcal{F}} < \epsilon$
- f must be in $\mathcal{F}$

Cauchy may not converge

- Convergent $\Rightarrow$ Cauchy

Cauchy may not converge

- Convergent $\Rightarrow$ Cauchy
- Cauchy may not converge (in $\mathcal{F}$)

Cauchy may not converge

- Convergent $\Rightarrow$ Cauchy
- Cauchy may not converge (in $\mathcal{F}$)
- Example: $C[0,1]$ bounded continuous functions on $[0,1]$

Cauchy may not converge

- Convergent $\Rightarrow$ Cauchy
- Cauchy may not converge (in $\mathcal{F}$)
- Example: $C[0,1]$ bounded continuous functions on $[0,1]$
- $\|f\| = \sqrt{\int_0^1 f(x)^2 dx}$

Cauchy may not converge

- Convergent $\Rightarrow$ Cauchy
- Cauchy may not converge (in $\mathcal{F}$)
- Example: $C[0,1]$ bounded continuous functions on $[0,1]$
- $\|f\| = \sqrt{\int_0^1 f(x)^2 dx}$
- $f_n(x) = 0$ for $x \in [0, \frac{1}{2} - \frac{1}{n}]$, 1 otherwise

Cauchy may not converge

- Convergent $\Rightarrow$ Cauchy
- Cauchy may not converge (in $\mathcal{F}$)
- Example: $C[0,1]$ bounded continuous functions on $[0,1]$
- $\|f\| = \sqrt{\int_0^1 f(x)^2 dx}$
- $f_n(x) = 0$ for $x \in [0, \frac{1}{2} - \frac{1}{n}]$, 1 otherwise
- $\{f_n(x)\}$ is Cauchy, but not convergent (limit $\notin C[0,1]$)

Banach space

- One may complete the vector space by adding the limits of all Cauchy sequences

Banach space

- One may complete the vector space by adding the limits of all Cauchy sequences
- A Banach space is a complete normed space

Banach space

- One may complete the vector space by adding the limits of all Cauchy sequences
- A Banach space is a complete normed space
- Example:
  $L_p(\mathcal{X}, \mu) = \left\{ f : \mathcal{X} \mapsto \mathbb{R} \text{ measurable} \mid \int_{\mathcal{X}} |f(x)|^p d\mu < \infty \right\}$
  with norm $\|f\|_p = \left(\int_{\mathcal{X}} |f(x)|^p d\mu \right)^{\frac{1}{p}}$ is a Banach space

Inner product

- Let $\mathcal{F}$ be a vector space over $\mathbb{R}$. A function $\langle \cdot, \cdot \rangle_{\mathcal{F}} : \mathcal{F} \times \mathcal{F} \mapsto \mathbb{R}$ is an inner product if

Inner product

- Let $\mathcal{F}$ be a vector space over $\mathbb{R}$. A function $\langle \cdot, \cdot \rangle_{\mathcal{F}} : \mathcal{F} \times \mathcal{F} \mapsto \mathbb{R}$ is an inner product if
  - $\langle af_1 + bf_2, g \rangle_{\mathcal{F}} = a\langle f_1, g \rangle_{\mathcal{F}} + b\langle f_2, g \rangle_{\mathcal{F}}$

Inner product

- Let $\mathcal{F}$ be a vector space over $\mathbb{R}$. A function $\langle\cdot,\cdot\rangle_{\mathcal{F}} : \mathcal{F} \times \mathcal{F} \mapsto \mathbb{R}$ is an inner product if
  - $\langle af_1 + bf_2, g\rangle_{\mathcal{F}} = a\langle f_1, g\rangle_{\mathcal{F}} + b\langle f_2, g\rangle_{\mathcal{F}}$
  - $\langle f, g\rangle_{\mathcal{F}} = \langle g, f\rangle_{\mathcal{F}}$

Inner product

- Let $\mathcal{F}$ be a vector space over $\mathbb{R}$. A function $\langle \cdot, \cdot \rangle_{\mathcal{F}} : \mathcal{F} \times \mathcal{F} \mapsto \mathbb{R}$ is an inner product if
  - $\langle af_1 + bf_2, g \rangle_{\mathcal{F}} = a\langle f_1, g \rangle_{\mathcal{F}} + b\langle f_2, g \rangle_{\mathcal{F}}$
  - $\langle f, g \rangle_{\mathcal{F}} = \langle g, f \rangle_{\mathcal{F}}$
  - $\langle f, f \rangle_{\mathcal{F}} \geq 0$ with 0 iff $f = 0$

Inner product

- Let $\mathcal{F}$ be a vector space over $\mathbb{R}$. A function $\langle\cdot,\cdot\rangle_{\mathcal{F}} : \mathcal{F} \times \mathcal{F} \mapsto \mathbb{R}$ is an inner product if
  - $\langle af_1 + bf_2, g\rangle_{\mathcal{F}} = a\langle f_1, g\rangle_{\mathcal{F}} + b\langle f_2, g\rangle_{\mathcal{F}}$
  - $\langle f, g\rangle_{\mathcal{F}} = \langle g, f\rangle_{\mathcal{F}}$
  - $\langle f, f\rangle_{\mathcal{F}} \geq 0$ with 0 iff $f = 0$
- An inner product space is a normed space with $\|f\| = \sqrt{\langle f, f\rangle}$

Hilbert space

- A Hilbert space is a complete inner product space, i.e. a Banach space with an inner product

Hilbert space

- A Hilbert space is a complete inner product space, i.e. a Banach space with an inner product
- Example: $L_2(\mathcal{X}, \mu)$ is a Hilbert space with inner product

$$\langle f, g \rangle = \int_{\mathcal{X}} f(x)g(x)d\mu$$

Linear functional

- Let $\mathcal{F}, \mathcal{G}$ be normed vector spaces over $\mathbb{R}$

Linear functional

- Let $\mathcal{F}, \mathcal{G}$ be normed vector spaces over $\mathbb{R}$
- A function $A : \mathcal{F} \mapsto \mathcal{G}$ is a linear operator iff

Linear functional

- Let $\mathcal{F}, \mathcal{G}$ be normed vector spaces over $\mathbb{R}$
- A function $A : \mathcal{F} \mapsto \mathcal{G}$ is a linear operator iff
  - $A(af) = aA(f), \ \forall a \in R, f \in \mathcal{F}$

Linear functional

- Let $\mathcal{F}, \mathcal{G}$ be normed vector spaces over $\mathbb{R}$
- A function $A : \mathcal{F} \mapsto \mathcal{G}$ is a linear operator iff
  - $A(af) = aA(f),\ \forall a \in R, f \in \mathcal{F}$
  - $A(f_1 + f_2) = A(f_1) + A(f_2),\ \forall f_1, f_2 \in \mathcal{F}$

Linear functional

- Let $\mathcal{F}, \mathcal{G}$ be normed vector spaces over $\mathbb{R}$
- A function $A : \mathcal{F} \mapsto \mathcal{G}$ is a linear operator iff
  - $A(af) = aA(f), \ \forall a \in R, f \in \mathcal{F}$
  - $A(f_1 + f_2) = A(f_1) + A(f_2), \ \forall f_1, f_2 \in \mathcal{F}$
- When $\mathcal{G} = \mathbb{R}$, A is a linear functional

Linear functional

- Let $\mathcal{F}, \mathcal{G}$ be normed vector spaces over $\mathbb{R}$
- A function $A : \mathcal{F} \mapsto \mathcal{G}$ is a linear operator iff
  - $A(af) = aA(f), \ \forall a \in R, f \in \mathcal{F}$
  - $A(f_1 + f_2) = A(f_1) + A(f_2), \ \forall f_1, f_2 \in \mathcal{F}$
- When $\mathcal{G} = \mathbb{R}$, A is a linear functional
- Example: For a fixed $h \in \mathcal{F}$,

$$A_h(f) = \langle f, h \rangle_{\mathcal{F}}$$

is a linear functional

Continuity

- $A : \mathcal{F} \mapsto \mathcal{G}$ is continuous at $f_0 \in \mathcal{F}$, if for every $\epsilon > 0$, $\exists \delta$ s.t.

$$\|f - f_0|_{\mathcal{F}} < \delta \;\Rightarrow\; \|Af - Af_0\|_{\mathcal{G}} < \epsilon$$

Continuity

- $A : \mathcal{F} \mapsto \mathcal{G}$ is continuous at $f_0 \in \mathcal{F}$, if for every $\epsilon > 0$, $\exists \delta$ s.t.

$$\|f - f_0|_{\mathcal{F}} < \delta \;\Rightarrow\; \|Af - Af_0\|_{\mathcal{G}} < \epsilon$$

- A is continuous on $\mathcal{F}$ if it is continuous at all $f \in F$

Riesz representation

In a Hilbert space $\mathcal{F}$, all continuous linear functionals are of the form $\langle \cdot, g \rangle_{\mathcal{F}}$, for some $g \in \mathcal{F}$.

Evaluation functional

- Let $\mathcal{X}$ be a non-empty set

Evaluation functional

- Let $\mathcal{X}$ be a non-empty set
- Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \mapsto \mathbb{R}$

Evaluation functional

- Let $\mathcal{X}$ be a non-empty set
- Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \mapsto \mathbb{R}$
- For a fixed $x \in \mathcal{X}$ the functional $\delta_x : \mathcal{H} \mapsto \mathbb{R}$ defined as

$$\delta_x(f) = f(x)$$

  is the Dirac evaluation functional at x

Evaluation functional

- Let $\mathcal{X}$ be a non-empty set
- Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \mapsto \mathbb{R}$
- For a fixed $x \in \mathcal{X}$ the functional $\delta_x : \mathcal{H} \mapsto \mathbb{R}$ defined as

$$\delta_x(f) = f(x)$$

  is the Dirac evaluation functional at x
- δ_x is linear:

$$\delta_x(af + bg) = (af + bg)(x) = af(x) + bg(x) = a\delta_x(f) + b\delta_x(g)$$

Evaluation functional

- Let $\mathcal{X}$ be a non-empty set
- Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \mapsto \mathbb{R}$
- For a fixed $x \in \mathcal{X}$ the functional $\delta_x : \mathcal{H} \mapsto \mathbb{R}$ defined as

  $$\delta_x(f) = f(x)$$

  is the Dirac evaluation functional at x
- δ_x is linear:

  $$\delta_x(af + bg) = (af + bg)(x) = af(x) + bg(x) = a\delta_x(f) + b\delta_x(g)$$

- Is δ_x continuous?

Evaluation functional

- Let $\mathcal{X}$ be a non-empty set
- Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \mapsto \mathbb{R}$
- For a fixed $x \in \mathcal{X}$ the functional $\delta_x : \mathcal{H} \mapsto \mathbb{R}$ defined as

  $$\delta_x(f) = f(x)$$

  is the Dirac evaluation functional at x
- δ_x is linear:

  $$\delta_x(af + bg) = (af + bg)(x) = af(x) + bg(x) = a\delta_x(f) + b\delta_x(g)$$

- Is δ_x continuous?
- ... Not necessarily

Reproducing Kernel Hilbert Space

- A Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \mapsto \mathbb{R}$ defined on a non-empty set $\mathcal{X}$ is a Reproducing Kernel Hilbert Space (RKHS) if δ_x is continuous for all $x \in \mathcal{X}$

Reproducing Kernel Hilbert Space

- A Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \mapsto \mathbb{R}$ defined on a non-empty set $\mathcal{X}$ is a Reproducing Kernel Hilbert Space (RKHS) if δ_x is continuous for all $x \in \mathcal{X}$
- The reproducing kernel of $\mathcal{H}$ is a function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ if it satisfies

Reproducing Kernel Hilbert Space

- A Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \mapsto \mathbb{R}$ defined on a non-empty set $\mathcal{X}$ is a Reproducing Kernel Hilbert Space (RKHS) if δ_x is continuous for all $x \in \mathcal{X}$
- The reproducing kernel of $\mathcal{H}$ is a function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ if it satisfies
  - $k(\cdot, x) \in \mathcal{H}, \forall x \in \mathcal{X}$

Reproducing Kernel Hilbert Space

- A Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \mapsto \mathbb{R}$ defined on a non-empty set $\mathcal{X}$ is a Reproducing Kernel Hilbert Space (RKHS) if δ_x is continuous for all $x \in \mathcal{X}$
- The reproducing kernel of $\mathcal{H}$ is a function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ if it satisfies
  - $k(\cdot, x) \in \mathcal{H}, \forall x \in \mathcal{X}$
  - $\langle f, k(\cdot, x)\rangle_{\mathcal{H}} = f(x), \forall f \in \mathcal{H}, x \in \mathcal{X}$ (reproducing)

Reproducing Kernel Hilbert Space

- A Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \mapsto \mathbb{R}$ defined on a non-empty set $\mathcal{X}$ is a Reproducing Kernel Hilbert Space (RKHS) if δ_x is continuous for all $x \in \mathcal{X}$
- The reproducing kernel of $\mathcal{H}$ is a function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ if it satisfies
  - $k(\cdot, x) \in \mathcal{H}, \forall x \in \mathcal{X}$
  - $\langle f, k(\cdot, x)\rangle_{\mathcal{H}} = f(x), \forall f \in \mathcal{H}, x \in \mathcal{X}$ (reproducing)
- Obviously,

$$\langle k(\cdot, y), k(\cdot, x)\rangle_{\mathcal{H}} = k(x, y)$$

Reproducing Kernel Hilbert Space

- A Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \mapsto \mathbb{R}$ defined on a non-empty set $\mathcal{X}$ is a Reproducing Kernel Hilbert Space (RKHS) if δ_x is continuous for all $x \in \mathcal{X}$
- The reproducing kernel of $\mathcal{H}$ is a function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ if it satisfies
  - $k(\cdot, x) \in \mathcal{H}, \forall x \in \mathcal{X}$
  - $\langle f, k(\cdot, x) \rangle_{\mathcal{H}} = f(x), \forall f \in \mathcal{H}, x \in \mathcal{X}$ (reproducing)
- Obviously,

$$\langle k(\cdot, y), k(\cdot, x) \rangle_{\mathcal{H}} = k(x, y)$$

- $\mathcal{H}$ is an RKHS (i.e. its evaluation functionals δ_x are continuous) iff $\mathcal{H}$ has a reproducing kernel

Positive definiteness

- A symmetric function $h : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is positive definite if $\forall n, \forall a \in \mathbb{R}^n, \forall x_1 \dots x_n \in \mathcal{X}$,

$$a^\top H a \geq 0$$

where H is the $n \times n$ matrix with $H_{ij} = h(x_i, x_j)$

Positive definiteness

- A symmetric function $h : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is positive definite if $\forall n, \forall a \in \mathbb{R}^n, \forall x_1 \dots x_n \in \mathcal{X}$,

$$a^\top H a \geq 0$$

  where H is the $n \times n$ matrix with $H_{ij} = h(x_i, x_j)$
- Reproducing kernels are positive definite

Positive definiteness

- A symmetric function $h : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is positive definite if $\forall n, \forall a \in \mathbb{R}^n, \forall x_1 \ldots x_n \in \mathcal{X}$,

$$a^\top H a \geq 0$$

  where H is the $n \times n$ matrix with $H_{ij} = h(x_i, x_j)$
- Reproducing kernels are positive definite
- Let $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ be positive definite. There is a unique RKHS $\mathcal{H} = \{f : \mathcal{X} \mapsto \mathbb{R}\}$ with reproducing kernel k [Moore-Aronszajn]

Representer Theorem

- Let $\mathcal{X}$ be a non-empty set

Representer Theorem

- Let $\mathcal{X}$ be a non-empty set
- Let k be a positive definite kernel on $\mathcal{X} \times \mathcal{X}$

Representer Theorem

- Let $\mathcal{X}$ be a non-empty set
- Let k be a positive definite kernel on $\mathcal{X} \times \mathcal{X}$
- Let $\mathcal{H}_k$ be the corresponding RKHS

Representer Theorem

- Let $\mathcal{X}$ be a non-empty set
- Let k be a positive definite kernel on $\mathcal{X} \times \mathcal{X}$
- Let $\mathcal{H}_k$ be the corresponding RKHS
- Let training data be $(x_1, y_1) \ldots (x_n, y_n) \in \mathcal{X} \times \mathbb{R}$

Representer Theorem

- Let $\mathcal{X}$ be a non-empty set
- Let k be a positive definite kernel on $\mathcal{X} \times \mathcal{X}$
- Let $\mathcal{H}_k$ be the corresponding RKHS
- Let training data be $(x_1, y_1) \dots (x_n, y_n) \in \mathcal{X} \times \mathbb{R}$
- Let the regularizer function $\Omega : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}$ be strictly monotonically increasing

Representer Theorem

- Let $\mathcal{X}$ be a non-empty set
- Let k be a positive definite kernel on $\mathcal{X} \times \mathcal{X}$
- Let $\mathcal{H}_k$ be the corresponding RKHS
- Let training data be $(x_1, y_1) \dots (x_n, y_n) \in \mathcal{X} \times \mathbb{R}$
- Let the regularizer function $\Omega : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}$ be strictly monotonically increasing
- Let the empirical risk function $\hat{R}$ be arbitrary

Representer Theorem

- Let $\mathcal{X}$ be a non-empty set
- Let k be a positive definite kernel on $\mathcal{X} \times \mathcal{X}$
- Let $\mathcal{H}_k$ be the corresponding RKHS
- Let training data be $(x_1, y_1) \ldots (x_n, y_n) \in \mathcal{X} \times \mathbb{R}$
- Let the regularizer function $\Omega : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}$ be strictly monotonically increasing
- Let the empirical risk function $\hat{R}$ be arbitrary
- Any minimizer

$$\operatorname*{argmin}_{f \in \mathcal{H}_k} \hat{R}((x_1, y_1, f(x_1)), \ldots, (x_n, y_n, f(x_n))) + \Omega(\|f\|)$$

admits the form

$$\sum_{i=1}^{n} \alpha_i k(\cdot, x_i)$$

References

Graphical Models

- Koller & Friedman, *Probabilistic Graphical Models*. MIT 2009

Kernel Methods

References

Graphical Models

- Koller & Friedman, *Probabilistic Graphical Models*. MIT 2009
- Wainwright & Jordan, *Graphical Models, Exponential Families, and Variational Inference*. FTML 2008

Kernel Methods

References

Graphical Models

- Koller & Friedman, *Probabilistic Graphical Models*. MIT 2009
- Wainwright & Jordan, *Graphical Models, Exponential Families, and Variational Inference*. FTML 2008
- Bishop, *Pattern Recognition and Machine Learning*. Springer 2006.

Kernel Methods

References

Graphical Models

- Koller & Friedman, *Probabilistic Graphical Models*. MIT 2009
- Wainwright & Jordan, *Graphical Models, Exponential Families, and Variational Inference*. FTML 2008
- Bishop, *Pattern Recognition and Machine Learning*. Springer 2006.

Kernel Methods

- Schölkopf & Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT 2001

References

Graphical Models

- Koller & Friedman, *Probabilistic Graphical Models*. MIT 2009
- Wainwright & Jordan, *Graphical Models, Exponential Families, and Variational Inference*. FTML 2008
- Bishop, *Pattern Recognition and Machine Learning*. Springer 2006.

Kernel Methods

- Schölkopf & Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT 2001
- Shawe-Taylor & Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge 2004

References

Graphical Models

- Koller & Friedman, *Probabilistic Graphical Models*. MIT 2009
- Wainwright & Jordan, *Graphical Models, Exponential Families, and Variational Inference*. FTML 2008
- Bishop, *Pattern Recognition and Machine Learning*. Springer 2006.

Kernel Methods

- Schölkopf & Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT 2001
- Shawe-Taylor & Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge 2004
- Dino Sejdinovic, Arthur Gretton, *What is an RKHS?* Online notes 2014