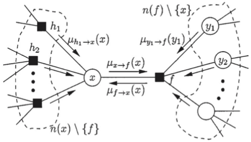
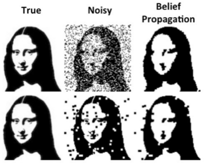
STA 414/2104: Probabilistic Machine Learning

Week 4: Message Passing

Murat A. Erdogdu





Announcements

Office hours for A1:

- Instructor OH: M 5-7pm in person
- TA OH 1: W 10-11am online (link on course webpage)
- TA OH 2: Th 4-5pm in person at MY480

Overview

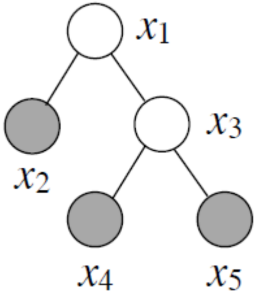
Today:

- Message passing
- Decision theory

Variable Elimination and Trees

- **Last week**: we covered exact inference by variable elimination:
  - We compute a conditional distribution like $p(x_3|\bar{x}_2, \bar{x}_4, \bar{x}_5)$ by starting from the full joint $p(x_1, \bar{x}_2, x_3, \bar{x}_4, \bar{x}_5, ...)$.
- Graph structure determines:
  - the computational cost,
  - and the optimal elimination ordering,
  - which is usually very hard to find anyways.
- Fortunately, for trees, any elimination ordering that goes from the leaves towards any root will be optimal.

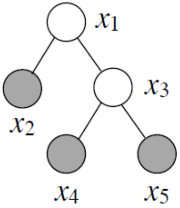
Inference in Trees (MRF with no cycles)



- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ the set of edges
- For $i, j \in \mathcal{V}$, we have $(i, j) \in \mathcal{E}$ if there is an edge between i and j.
- For a node in graph $i \in \mathcal{V}$, $N(i)$ denotes the neighbors of i: $N(i) = \{j : (i, j) \in \mathcal{E}\}$.

In the figure: $\mathcal{V} = \{1, 2, 3, 4, 5\}$ (or $\mathcal{V} = \{x_1, x_2, x_3, x_4, x_5\}$) and $\mathcal{E} = \{(1, 2), (1, 3), (3, 4), (3, 5)\}$. Shaded nodes are observed: $\bar{x}_2, \bar{x}_4, \bar{x}_5$.

Inference in Trees



- Joint distribution is

$$p(x_1, ..., x_5) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \psi_i(x_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j).$$

- We want to compute $p(x_3 | \bar{x}_2, \bar{x}_4, \bar{x}_5)$

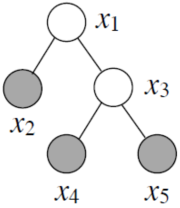
- We have

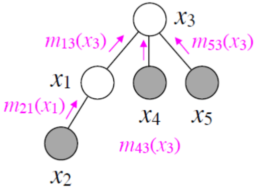
$$p(x_3 | \bar{x}_2, \bar{x}_4, \bar{x}_5) \propto p(x_3, \bar{x}_2, \bar{x}_4, \bar{x}_5) = \sum_{x_1} p(x_1, \bar{x}_2, x_3, \bar{x}_4, \bar{x}_5).$$

$$p(x_3 \mid \bar{x}_2, \bar{x}_4, \bar{x}_5) = \frac{1}{Z^E} \sum_{x_1} \psi_1(x_1)\psi_3(x_3)\psi_2(\bar{x}_2)\psi_4(\bar{x}_4)\psi_5(\bar{x}_5)\psi_{12}(\bar{x}_2, x_1)\psi_{34}(\bar{x}_4, x_3)\psi_{35}(\bar{x}_5, x_3)\psi_{13}(x_1, x_3)$$

- Let's write the variable elimination algorithm.

Inference in Trees





$$\begin{aligned}
p(x_3 \mid \bar{x}_2, \bar{x}_4, \bar{x}_5) &= \frac{1}{Z^E}\sum_{x_1}\psi_1(x_1)\psi_3(x_3)\psi_2(\bar{x}_2)\psi_4(\bar{x}_4)\psi_5(\bar{x}_5)\psi_{12}(\bar{x}_2, x_1)\psi_{34}(\bar{x}_4, x_3)\psi_{35}(\bar{x}_5, x_3)\psi_{13}(x_1, x_3)\\
&= \frac{1}{Z^E}\underbrace{\psi_4(\bar{x}_4)\psi_{34}(\bar{x}_4, x_3)}_{m_{43}(x_3)}\underbrace{\psi_5(\bar{x}_5)\psi_{35}(\bar{x}_5, x_3)}_{m_{53}(x_3)}\psi_3(x_3)\sum_{x_1}\psi_1(x_1)\psi_{13}(x_1, x_3)\underbrace{\psi_2(\bar{x}_2)\psi_{12}(\bar{x}_2, x_1)}_{m_{21}(x_1)}\\
&= \frac{1}{Z^E}\psi_3(x_3)m_{43}(x_3)m_{53}(x_3)\underbrace{\sum_{x_1}\psi_1(x_1)\psi_{13}(x_1, x_3)m_{21}(x_1)}_{m_{13}(x_3)}\\
&= \frac{1}{Z^E}\psi_3(x_3)m_{43}(x_3)m_{53}(x_3)m_{13}(x_3) = \frac{\psi_3(x_3)m_{43}(x_3)m_{53}(x_3)m_{13}(x_3)}{\sum_{x_3}\psi_3(x_3)m_{43}(x_3)m_{53}(x_3)m_{13}(x_3)}
\end{aligned}$$

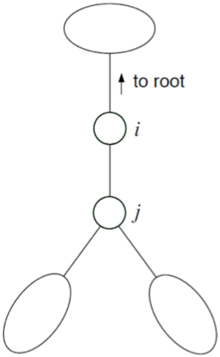
Slide credit: S. Ermon

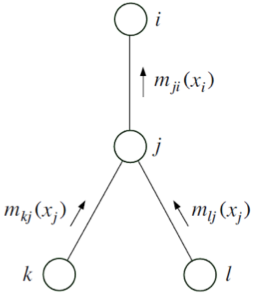
Message Passing on Trees

Belief propagation on trees:

- The message sent from variable j to $i \in N(j)$:
  - If x_j is not observed
$$m_{j\to i}(x_i) = \sum_{x_j} \psi_j(x_j)\psi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(x_j)$$
  - If x_j is observed
$$m_{j\to i}(x_i) = \psi_j(\bar{x}_j)\psi_{ij}(x_i, \bar{x}_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(\bar{x}_j)$$
- Once the message passing stage is complete, we can compute our beliefs as
$$b(x_i) \propto \psi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i).$$
- In this example, beliefs are the marginals we wanted to compute!
$$b(x_i) = p(x_i|\bar{x}_j, \bar{x}_k, ...).$$

Message Passing on Trees





The message sent from variable j to $i \in N(j)$ is

$$m_{j \to i}(x_i) = \sum_{x_j} \psi_j(x_j)\psi_{ij}(x_i, x_j) \prod_{k \in N(j) \setminus i} m_{k \to j}(x_j)$$

Inference in Trees: Compute $p(x_3|\bar{x}_2, \bar{x}_4, \bar{x}_5)$

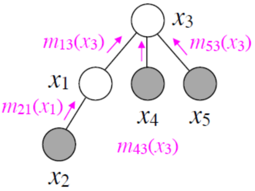
The message sent from variable j to $i \in N(j)$:

- If x_j is not observed

$$m_{j\to i}(x_i) = \sum_{x_j} \psi_j(x_j)\psi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(x_j)$$

- If x_j is observed

$$m_{j\to i}(x_i) = \psi_j(\bar{x}_j)\psi_{ij}(x_i, \bar{x}_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(\bar{x}_j)$$

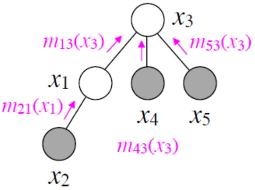


- $m_{5\to 3}(x_3) = \psi_5(\bar{x}_5)\psi_{35}(x_3, \bar{x}_5)$
- $m_{2\to 1}(x_1) = \psi_2(\bar{x}_2)\psi_{12}(x_1, \bar{x}_2)$
- $m_{4\to 3}(x_3) = \psi_4(\bar{x}_4)\psi_{34}(x_3, \bar{x}_4)$
- $m_{1\to 3}(x_3) = \sum_{x_1} \psi_1(x_1)\psi_{13}(x_1, x_3)m_{2\to 1}(x_1)$

Inference in Trees: Compute $p(x_3|\bar{x}_2, \bar{x}_4, \bar{x}_5)$

Beliefs are calculated in the end, after passing all messages:

$$b(x_i) \propto \psi_i(x_i) \prod_{j \in N(i)} m_{j\to i}(x_i).$$



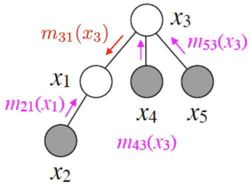
- Now that we computed the messages we need $m_{5\to 3}(x_3), m_{2\to 1}(x_1), m_{4\to 3}(x_3), m_{1\to 3}(x_3)$,
- $b(x_3) \propto \psi_3(x_3) m_{1\to 3}(x_3) m_{4\to 3}(x_3) m_{5\to 3}(x_3)$

This is the same as variable elimination, so

$$p(x_3|\bar{x}_2, \bar{x}_4, \bar{x}_5) = b(x_3)$$

Inference in Trees: Compute $p(x_1|\bar{x}_2, \bar{x}_4, \bar{x}_5)$

Single pass only gives us the correct marginal at the root node.



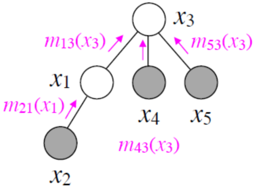
- If we wanted to compute $p(x_1|\bar{x}_2, \bar{x}_4, \bar{x}_5)$, we would also need $m_{3\to1}(x_1) = \sum_{x_3} \psi_3(x_3)\psi_{13}(x_1, x_3)m_{4\to3}(x_3)m_{5\to3}(x_3)$
- $b(x_1) \propto \psi_1(x_1)m_{3\to1}(x_1)m_{2\to1}(x_1)$

We would obtain

$$p(x_1|\bar{x}_2, \bar{x}_4, \bar{x}_5) = b(x_1)$$

Belief Propagation on Trees

Belief Propagation Algorithm on Trees



- Choose root r arbitrarily
- Pass messages from leafs to r
- Pass messages from r to leafs
- These two passes are sufficient on trees!
- Finally, compute the beliefs

- One pass is enough to compute the marginal of the root node.
- Two passes would give marginals of all the nodes.

Loopy Belief Propagation

- What if the graph (MRF) we have is not a tree and have cycles?
- Keep passing messages until “convergence”.
- This is called **Loopy Belief Propagation**.
- We won’t get the exact marginals.
- But turns out it is still a very good approximation!

Loopy Belief Propagation

Loopy BP:

- Initialize all messages uniformly:

$$m_{j\to i}(x_i) = [1/k, ..., 1/k]^\top$$

  where k is the number of states x_j can take.

- Keep running BP updates until they "converge":

$$m_{j\to i}(x_i) = \sum_{x_j} \psi_j(x_j)\psi_{ij}(x_i, x_j) \prod_{k\in N(j)\backslash i} m_{k\to j}(x_j)$$

  and (sometimes) normalized for stability.

- It will generally not converge, but that's ok.
- Compute beliefs

$$b(x_i) \propto \psi_i(x_i) \prod_{j\in\mathcal{N}(i)} m_{j\to i}(x_i).$$

Sum-product vs. Max-product

- The algorithm we learned is called **sum-product BP** and approximately computes the **marginals** at each node.
- For MAP inference, we maximize over x_j instead of summing over them. This is called **max-product BP**.
- BP updates take the form

$$m_{j\to i}(x_i) = \max_{x_j}\left\{\psi_j(x_j)\psi_{ij}(x_i,x_j)\prod_{k\in N(j)\backslash i} m_{k\to j}(x_j)\right\}$$

- After BP algorithm converges, the beliefs are **max-marginals**

$$b(x_i) \propto \psi_i(x_i)\prod_{j\in\mathcal{N}(i)} m_{j\to i}(x_i).$$

Inference

- Beliefs are approximations to marginals

$$b(x_i) = p(x_i | \bar{x}_j, \bar{x}_k, ...)$$

- After computing all the beliefs, we can predict the value of each variable as

$$\hat{x}_i = \arg\max_{x_i} b(x_i)$$

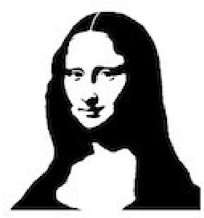
 or the same with max-product BP.

Summary: Loopy BP

- Loopy BP is not exact but it is still very useful in practice, without much theoretical guarantee (other than tree-like graphs).
- Loopy BP is often over-confident since it amplifies the effect of each potential.
- Loopy BP can oscillate, but this is generally ok.
- Loopy BP often works better if we normalize messages, and use momentum in the updates.
- The algorithm we learned is called **sum-product BP**. If we are interested in MAP inference, we can maximize over x_j instead of summing over them. This is called **max-product BP**.

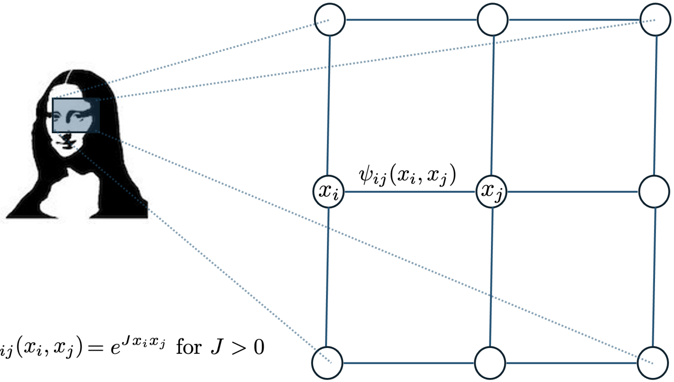
Image Denoising

- A binary image is a $\sqrt{n} \times \sqrt{n}$ matrix where each entry is $+1$ or -1.
- We vectorize this matrix and denote the image as $\boldsymbol{x} \in \mathbb{R}^n$.
- For example, the Mona Lisa below is a 128×128 image, vectorized to be $\boldsymbol{x} \in \mathbb{R}^{16384}$.



An MRF for Mona Lisa

- The set of nodes $\mathcal{V} = \{1, 2, ..., 128^2\}$
- The set of edges $\mathcal{E} = \{(i, j) : \ i \text{ and } j \text{ has an edge based on grid}\}$
- $x_i \in \{\pm 1\}$

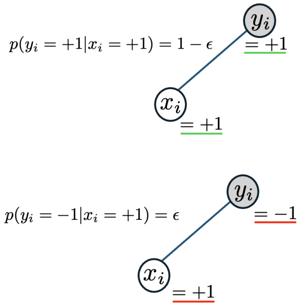


$\psi_{ij}(x_i, x_j) = e^{J x_i x_j}$ for $J > 0$

$$p(x_1, ..., x_n) \propto \prod_{(i,j)\in\mathcal{E}} \psi_{ij}(x_i, x_j) = e^{\sum_{(i,j)\in\mathcal{E}} J x_i x_j}$$

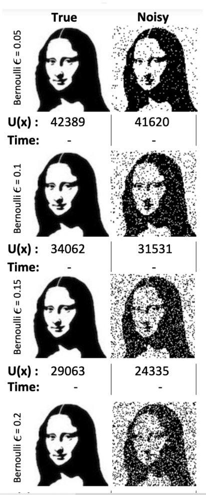
Noisy Image

- Assume that the image has been sent through a noisy channel, where each pixel is flipped with a small probability ϵ.
- True pixels x_i's are unobserved, we only observe noisy pixels y_i's.

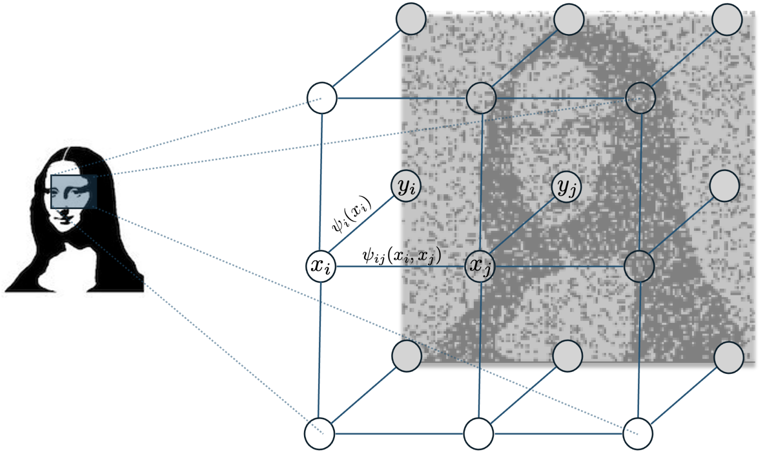


$p(y_i = +1 | x_i = +1) = 1 - \epsilon$
y_i
$= +1$
x_i
$= +1$
$p(y_i = -1 | x_i = +1) = \epsilon$
y_i
$= -1$
x_i
$= +1$

$$p(y_i | x_i) = (1-\epsilon)^{\frac{1+y_i x_i}{2}} \epsilon^{\frac{1-y_i x_i}{2}}$$
$$:= \psi_i(x_i)$$



An MRF for Denoising Mona Lisa



Inference Task: Image Denoising

1. We have the joint over x_i's induced by the MRF:

$$p(x_1, ..., x_n) \propto \prod_{(i,j)\in\mathcal{E}} \psi_{i,j}(x_i, x_j) = \prod_{(i,j)\in\mathcal{E}} e^{Jx_ix_j}.$$

2. We have a Bernoulli ϵ-noise model:

$$p(y_i|x_i) = (1-\epsilon)^{\frac{1+x_iy_i}{2}} \epsilon^{\frac{1-x_iy_i}{2}} := \psi_i(x_i).$$

3. Inference task:

$$\text{Compute} \quad p(x_i|y_1, ..., y_n) \text{ for all } i$$

Image Denoising

$$\begin{aligned}
p(y_i|x_i) =& (1-\epsilon)^{\frac{1+y_ix_i}{2}}\epsilon^{\frac{1-y_ix_i}{2}} \qquad \text{for all } i. \\
=& \exp\left\{\frac{1+y_ix_i}{2}\log(1-\epsilon)+\frac{1-y_ix_i}{2}\log(\epsilon)\right\} \\
\propto& \exp\left\{y_ix_i\frac{1}{2}\log\left(\frac{1-\epsilon}{\epsilon}\right)\right\} \\
=& e^{\beta y_i x_i} \quad \text{where} \;\; \beta \coloneqq \frac{1}{2}\log\left(\frac{1-\epsilon}{\epsilon}\right)
\end{aligned}$$

Back to Ising Model

- Therefore, we have

$$\begin{aligned} p(x_1, ..., x_n | y_1, ..., y_n) &\propto p(x_1, ..., x_n, y_1, ..., y_n) \\ &= p(x_1, ..., x_n) \prod_{i \in \mathcal{V}} p(y_i | x_i) \\ &\propto \exp\left\{ J \sum_{(i,j) \in \mathcal{E}} x_i x_j + \beta \sum_{i \in \mathcal{V}} y_i x_i \right\} \\ &= \prod_{(i,j) \in \mathcal{E}} \psi_{i,j}(x_i, x_j) \prod_{i \in \mathcal{V}} \psi_i(x_i) \end{aligned}$$

- Now, it is clear that node potentials are given by

$$\psi_i(x_i) = \exp(\beta y_i x_i)$$

Hyperparameters

- The posterior looks like

$$p(x_1, ..., x_n | y_1, ..., y_n) \propto \exp\left\{ J \sum_{(i,j)\in\mathcal{E}} x_i x_j + \beta \sum_{i\in\mathcal{V}} y_i x_i \right\}$$

- Parameters:
  - J controls how similar each pixel should be compared to its neighbors.
  - β controls how similar each pixel should be to its noisy observation.
  - Recall: $\beta = \frac{1}{2}\log\left(\frac{1-\epsilon}{\epsilon}\right)$
  - so when ϵ is small, β should be large.

Inference Task: Image Denoising

Recall Loopy BP

1. Choose an arbitrary root
2. Repeat:
    - Pass messages from leaves to root
    - Pass messages from root to leaves
3. Compute beliefs.

Loopy BP: Message Passing step

We will vectorize the functions:

- Function dimension is the same as the number of states its argument takes.
- Each message $m_{j\to i}(x_i)$ is stored as a 2-dimensional vector, where its first and second coordinates are $m_{j\to i}(+1)$ and $m_{j\to i}(-1)$, respectively.
- Initialize all messages with

$$m_{j\to i}(x_i) = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$$

Loopy BP: Message Passing step

1. Vectorize all potentials corresponding to edges and nodes:

$$\psi_i(x_i) = \begin{bmatrix} e^{+\beta y_i} \\ e^{-\beta y_i} \end{bmatrix} \quad \psi_{i,j}(x_i, x_j) = \begin{bmatrix} e^{+J} & e^{-J} \\ e^{-J} & e^{+J} \end{bmatrix},$$

2. Message update

$$m_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i, x_j)\psi_j(x_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(x_j),$$

- Product: compute the elementwise product of $\psi_j(x_j)$ and $m_{k\to j}(x_j)$ for all $k \in N(j) \setminus \{i\}$, which will be a vector.
- Summation: Multiple the matrix $\psi_{i,j}(x_i, x_j)$ with the vector above.

Loopy BP: Stability Improvements

After computing $m^{\text{new}}_{j\to i}(x_i)$, we can improve stability via

- normalization after each step:

$$m^{\text{new}}_{j\to i}(x_i) \leftarrow m^{\text{new}}_{j\to i}(x_i) / \sum_{x_i} m^{\text{new}}_{j\to i}(x_i)$$

- momentum in the message passing step: For some momentum parameter $\eta \in (0,1)$,

$$m^{\text{new}}_{j\to i}(x_i) \leftarrow \eta \; m^{\text{new}}_{j\to i}(x_i) + (1-\eta) \; m^{\text{old}}_{j\to i}(x_i)$$

Loopy BP: Message Passing step

1. For some number of iterations, keep passing messages. It will generally not converge, but change in messages will be small.
2. Compute beliefs

$$b(x_i) \propto \psi_i(x_i) \prod_{j \in N(i)} m_{j \to i}(x_i).$$

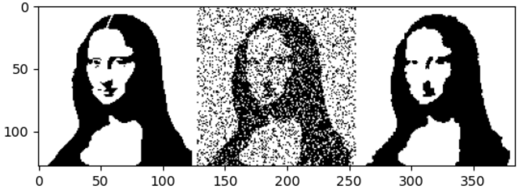
- Beliefs $b(x_i)$ are 2-dimensional vectors, with $b(+1)$ and $b(-1)$ being the first and second coordinates.
- Compute the elementwise product of $\psi_i(x_i)$ and $m_{j \to i}(x_i)$ for all $j \in N(i)$.
- Normalize them so they are probabilities: $b(x_i) \leftarrow b(x_i) / \sum_{x_i} b(x_i)$.

Loopy BP for Image Denoising

- While loopy BP may not converge, 10-20 iterations suffice to perform approximate inference on the posterior.
- The computed beliefs correspond to $p(x_i|y_1, ..., y_n)$.
- One decision rule to estimate x_i is

$$\hat{x}_i = \operatorname{argmax}_{x_i} b(x_i).$$

- The result is quite nice!



Loopy BP: Issues

- Convergence is always a problem.
- Over-confident; thus, less accurate.
- If the noise is not iid, results deteriorate fast.
- There are advanced versions to remedy some of these, e.g. Generalized Belief Propagation.

Summary

- Belief propagation (BP) is an exact variable elimination method on trees.
  - One pass is enough to get the marginal at the root node.
  - Two passes would give the marginals at all nodes.
- Loopy BP is the same algorithm applied iteratively on a general graph, without any convergence guarantees.
  - Message passing stage may seem unstable.
  - We use momentum and normalization to improve stability.
  - Beliefs will often provide overconfident estimates of marginals.
- We worked with a single image so there was **no learning** from data. In other words, we did not do ’machine learning’.

Decision making

We develop a small amount of theory that provides a framework for understanding many of the models we consider.

- Suppose we have a real-valued input vector x and a corresponding target (output) value c with joint probability distribution: $p(x, c)$.
- Our goal is to predict the output label c given a new value for x.
- For now, we focus on classification so c is a categorical variable, but the same reasoning applies to regression (continuous target).

The joint probability distribution $p(x, c)$ provides a complete summary of uncertainties associated with these random variables.

Inference

Estimating/approximating $p(x, c)$ is an example of an **inference** task.

Decision rule

In the Mona Lisa example, we used the following rule to predict pixel values

$$\hat{x}_i = \text{argmax}_{x_i}\, b(x_i) = p(x_i|y_1, ..., y_n).$$

This decision rule predicts $x_i = 1$ in both cases below:

- $b(x_i = +1) = 0.9$
- $b(x_i = +1) = 0.51$

Example: Cancer screening from chest X-ray

Based on the X-ray image, we would like determine whether the patient has cancer or not.

- The input vector x is pixel intensities, and the output c represents the presence of cancer, class $\mathcal{C}_1$, or absence of cancer, class $\mathcal{C}_2$.

- $\mathcal{C}_1$ cancer present
- $\mathcal{C}_2$ cancer absent

We can use an "arbitrary" encoding for these classes $\mathcal{C}_1$ and $\mathcal{C}_2$.

Inference

Inference Problem

Let's assume we estimated the joint distribution $p(x, \mathcal{C}_k)$ using some ML method. In the end, we must make a decision of whether to give treatment to the patient or not.

- Given a new X-ray image, our goal is to decide which of the two classes that image should be assigned to. We could compute conditional probabilities of the two classes, given the input image:

$$p(\mathcal{C}_k|x) = \frac{p(x|\mathcal{C}_k)p(\mathcal{C}_k)}{p(x)} \quad \text{Bayes' rule.}$$

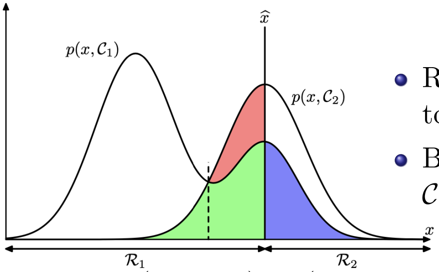
- If we minimize the expected number of mistakes, we can minimize the probability of assigning x to the wrong class.
  This suggests we minimize the **misclassification rate**.

Misclassification rate

Goal

Make as few misclassifications as possible. We need a rule that assigns each value of x to one of the available classes.

Divide the input space into regions $\mathcal{R}_k$ (decision regions) such that all points in $\mathcal{R}_k$ are assigned to class $\mathcal{C}_k$.

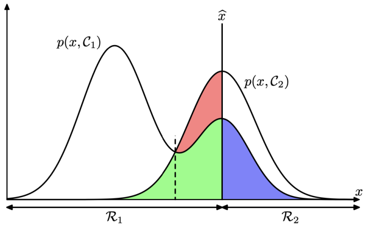


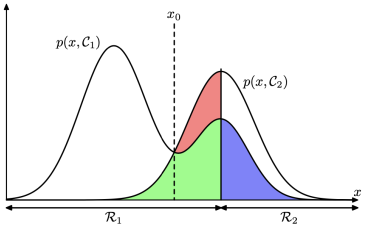
- Red + green regions: input belongs to class $\mathcal{C}_2$, but is assigned to $\mathcal{C}_1$.
- Blue region: input belongs to class $\mathcal{C}_1$, but is assigned to $\mathcal{C}_2$.

$$\begin{aligned} p(\text{mistake}) =& p(x \in \mathcal{R}_1, \mathcal{C}_2) + p(x \in \mathcal{R}_2, \mathcal{C}_1) \\ =& \int_{\mathcal{R}_1} p(x, \mathcal{C}_2) dx + \int_{\mathcal{R}_2} p(x, \mathcal{C}_1) dx \end{aligned}$$

Misclassification rate

Compare the following two decision rules:





- Blue + green area is always included in the $p(\text{mistake})$.
- Therefore, we aim to reduce the red area by moving the threshold $\hat{x}$ to x_0, which turns out to be optimal in this case.

Misclassification error

- Misclassification error:

$$p(\text{mistake}) = \underbrace{\int_{\mathcal{R}_1} p(x, \mathcal{C}_2) dx}_{\text{red+green}} + \underbrace{\int_{\mathcal{R}_2} p(x, \mathcal{C}_1) dx}_{\text{blue}}$$

  and the decision regions $\mathcal{R}_1$ and $\mathcal{R}_2$ are disjoint.
- Therefore, for a particular input x, if $p(x, \mathcal{C}_1) > p(x, \mathcal{C}_2)$, then we assign x to class $\mathcal{C}_1$. I.e. $\mathcal{R}_1 = \{x : p(x, \mathcal{C}_1) > p(x, \mathcal{C}_2)\}$.

Minimizing misclassification

In order to minimize the probability of making mistake, we assign each x to the class for which the probability $p(\mathcal{C}_k, x)$ is largest. This minimizes the misclassification rate.

Expected loss

How realistic is it to minimize the misclassification rate?

- We want a **loss function** to measure the loss incurred by taking any of the available decisions.
- Suppose that for x, the true class is $\mathcal{C}_k$, but we assign x to class $\mathcal{C}_j$ and incur loss of L_{kj} ((k, j)-th element of a loss matrix).

Consider medical diagnosis example: example of a loss matrix:

Incorrectly classify as healthy

$$\text{Truth}\quad \begin{array}{c|cc} & \multicolumn{2}{c}{\text{Decision}} \\ & \text{cancer} & \text{normal} \\ \hline \text{cancer} & 0 & 1000 \\ \text{normal} & 1 & 0 \end{array}$$

Incorrectly classify as cancer

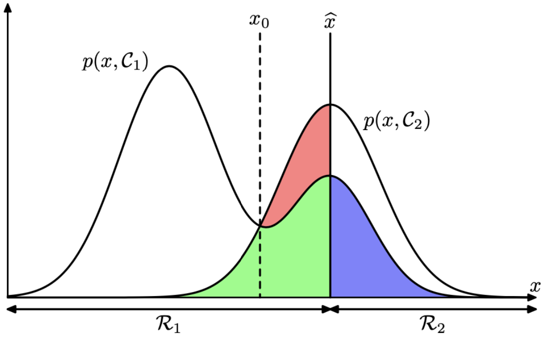
Thus the expected loss is given by

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj}\ p(x, \mathcal{C}_k) dx$$

New goal: Minimize expected loss

New objective:

Choose regions $\mathcal{R}_j$ as to minimize expected loss.



In the above figure, the blue region corresponds to L_{12}: the sample comes from class $\mathcal{C}_1$ but we classified as $\mathcal{C}_2$.

Minimize expected loss

Therefore, we want to minimize

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj}\ p(x, \mathcal{C}_k) dx$$
$$= \sum_j \int_{\mathcal{R}_j} \sum_k L_{kj}\ p(x, \mathcal{C}_k) dx.$$

Define $g_j(x) = \sum_k L_{kj}\ p(x, \mathcal{C}_k)$: the overall cost of incorrectly assigning x to class $\mathcal{C}_j$. Then, the expected loss is equal to

$$\mathbb{E}[L] = \sum_j \int_{\mathcal{R}_j} g_j(x) dx$$

Thus, minimizing $\mathbb{E}[L]$ is equivalent to choosing

$$\mathcal{R}_j = \{x \ : \ g_j(x) < g_i(x) \ \text{ for all } \ i \neq j\}.$$

Simplifying further

We can also use the product rule $p(x, \mathcal{C}_1) = p(\mathcal{C}_1|x)p(x)$ and reduce the problem to:

Discriminant rules:

Find regions $\mathcal{R}_j$ such that the following is minimized:

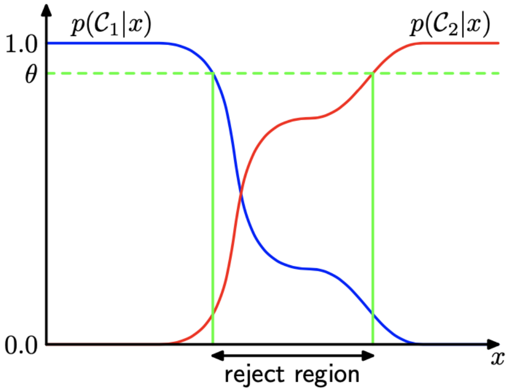
$$\sum_k L_{kj}\ p(\mathcal{C}_k|x).$$

That is

$$\mathcal{R}_j = \left\{x\ : \sum_k L_{kj}\ p(\mathcal{C}_k|x) < \sum_k L_{ki}\ p(\mathcal{C}_k|x) \ \text{ for all }\ i \neq j\right\}.$$

Reject option

For the regions where we are relatively uncertain about class membership, we don't have to make a decision.



Here, notice that we have a threshold θ and the conditional class probabilities fall below this threshold, we refuse to make a decision.

Loss functions for regression

- Now we consider an input/target setup (x, t) where the target (output) is continuous $t \in \mathbb{R}$, and the joint density is $p(x, t)$.
- Instead of decision regions, we aim to find a regression function $y(x) \approx t$ which maps inputs to the outputs.
- Consider the squared loss function L between $y(x)$ and t to assess the quality of our estimate $L(y(x), t) = (y(x) - t)^2$.

Goal:

What is the best function $y(x)$ that minimizes the expected loss?

$$\mathbb{E}[L] = \int \int L(y(x), t) p(x, t) dx dt.$$

Minimizing expected loss: Best regression function

We add and subtract $\mathbb{E}[t|x]$ and write

$$\begin{aligned}
\mathbb{E}[L] &= \int\int (y(x) - t)^2 p(x,t)dxdt \\
&= \int\int (y(x) - \mathbb{E}[t|x] + \mathbb{E}[t|x] - t)^2 p(x,t)dxdt \\
&= \int\int (y(x) - \mathbb{E}[t|x])^2 p(x,t)dxdt + \int\int (\mathbb{E}[t|x] - t)^2 p(x,t)dxdt \\
&\quad + 2\int\int (y(x) - \mathbb{E}[t|x])(\mathbb{E}[t|x] - t)p(x,t)dxdt
\end{aligned}$$

The last term is zero since

$$\begin{aligned}
&\int\int (y(x) - \mathbb{E}[t|x])(\mathbb{E}[t|x] - t)p(x,t)dxdt \\
&= \int\int (y(x) - \mathbb{E}[t|x])(\mathbb{E}[t|x] - t)p(t|x)p(x)dxdt \\
&= \int (y(x) - \mathbb{E}[t|x])\Big\{\underbrace{\int (\mathbb{E}[t|x] - t)p(t|x)dt}_{=0}\Big\}p(x)dx = 0
\end{aligned}$$

Best regression function

- We showed that the expected loss is given by the sum of two **non-negative** terms

$$\mathbb{E}[L] = \int\int (y(x) - \mathbb{E}[t|x])^2 p(x,t)dxdt + \int\int (\mathbb{E}[t|x] - t)^2 p(x,t)dxdt.$$

- The second term does not depend on $y(x)$ thus choosing the best regression function $y(x)$ is equivalent to minimizing the first term on the right hand side.
- Since that term is always non-negative, we can make it zero by choosing

$$y(x) = \mathbb{E}[t|x].$$

- The second term is the expectation of the conditional variance of $t|x$. It represents the intrinsic variability of the target data and can be regarded as noise.

Summary: Decision making

- Depending on the application, one needs to choose an appropriate loss function.
- Loss function can significantly change the optimal decision rule.
- One can always use the reject option and not make a decision.
- In case of regression, one can find the optimal map between x and t if one knows the conditional distribution $t|x$. The optimal map corresponds to the conditional expectation $\mathbb{E}[t|x]$.