2.3. Maximum A Posteriori (MAP) Inference

Jeong Hwan, Lee

KAIST Dept. of Mathematical Sciences

August 17, 2018

Contents

1. Introduction.

Let's recall the **inference problems** in graphical models. Given a probabilistic model (such as a Bayesian network or an MRF), we are interested in using to answer *useful questions*. We will be focusing on two types of questions. :

1. Marginal inference : What is the probability of a given variable in our model after we sum everything else out?
2. **Maximum A Posteriori (MAP) inference** : What is the most likely assignment to the variables in the model (possibly conditioned on *evidence*).

1. Introduction.

$*$ **MAP Inference.**

First, note that any Bayesian network can be viewed as an MRF since they can be written in the form of **Gibbs distribution**. :

$$p(x) = \frac{1}{Z} \cdot \prod_{c \in \mathcal{C}} \phi_c(x_c),$$

where $\mathcal{C}$ denotes the set of all cliques of the given graph G and $Z := \sum_{x \in \mathcal{X}} \left[\prod_{c \in \mathcal{C}} \phi_c(x_c) \right]$ denotes the partition function.

1. Introduction.

$\therefore$ The MAP inference in the above MRF corresponds to the following optimization problem. :

$$\max \{\log p(x) : x \in \mathcal{X}\} = \max \left\{ \sum_{c \in \mathcal{C}} \theta_c(x_c) : x \in \mathcal{X} \right\} - \log Z,$$

where $\theta_c(x_c) := \log \phi_c(x_c)$. In this case, the *computationally intractable* partition constant $\log Z$ does not depend on x and thus can be **ignored** in the above optimization problem. Thus, MAP inference is easier than marginal inference. :

$$\arg\max \left\{ \sum_{c \in \mathcal{C}} \theta_c(x_c) : x \in \mathcal{X} \right\}.$$

1. Introduction.

In the previous seminar, we briefly studied how to solve the inference problems in graphical models within the message-passing framework, which is called the **belief propagation algorithms**.

1. Marginal inference : Sum-product message passing.
2. MAP inference : Max-product message passing.

From now on, we will look at more efficient specialized methods for the MAP inference.

2. The Challenge of MAP Inference.

* **Method 1. Enumeration-based Approach.**
  - Marginal inference : Computing and summing all assignments to the model!
  - MAP inference : Just replace summation with maximization from the approach for marginal inference.

However, there exist more efficient methods than these enumeration-based approaches.

2. The Challenge of MAP Inference.

Remark

1. MAP inference is still not an *easy problem* in the general case.

: The objective function $\sum_{c \in \mathcal{C}} \theta_c(x_c)$ includes many intractable problems as special cases, *e.g.* 3-SAT.

- Construct for each clause $c = (x \vee y \vee \neg z)$, a factor $\theta_c(x, y, z)$ which is defined by

$$\theta_c(x, y, z) := \begin{cases} 1 & \text{if} \quad x, y, z \text{ satisfy clause } c \\ 0 & \text{if} \quad \text{otherwise} \end{cases}$$

For this case, the 3-SAT instance is **satisfiable** if and only if the value of the MAP assignment equals the number of clauses.

2. The Challenge of MAP Inference.

∗ **Boolean Satisfiability Problem (= SAT).**

Definition

(1) A propositional logic formula consists of *variables*, *operators* as following and *parentheses*. :

- AND : conjunction, also denoted by $\wedge$.
- OR : disjunction, $\vee$.
- NOT : negation, $\neg$.

(2) A formula is satisfiable if it can be made TRUE by assigning appropriate logical values (= TRUE, FALSE).

(3) The Boolean satisfiability problem (SAT) is, given a formula, to check whether it is satisfiable.

2. The Challenge of MAP Inference.

The followings are several structures in a formula. :

(4) A literal is either a variable, called positive literal, or the negation of a variable, called negative literal.
(5) A clause is a disjunction of finitely many literals. Also, a clause is called a Horn clause if it contains **at most one** positive literal.
(6) A formula is in conjunctive normal form (CNF) if it is a conjunction of finitely many clauses.

It is well-known that SAT is the first problem that was proven to be NP-hard (Cook-Levin Theorem).

2. The Challenge of MAP Inference.

Example

Inference in a conditional random field (CRF) model $p(y|x)$. :

$$p(y|x) = \frac{1}{Z(x)} \prod_{c \in \mathcal{C}} \phi_c(x_c, y_c).$$

The MAP inference problem for this model can be formulated as

$$\arg\max \{p(y|x) : y \in \mathcal{Y}\} = \arg\max \left\{ \sum_{c \in \mathcal{C}} \theta_c(x_c, y_c) : y \in \mathcal{Y} \right\},$$

where $\theta_c(x_c, y_c) := \log \phi_c(x_c, y_c)$.

2. The Challenge of MAP Inference.

Many interesting examples of MAP inference comes from instances of **structured prediction** (= an umbrella term for supervised learning techniques that involves predicting *structured objects*).

- Handwriting recognition :

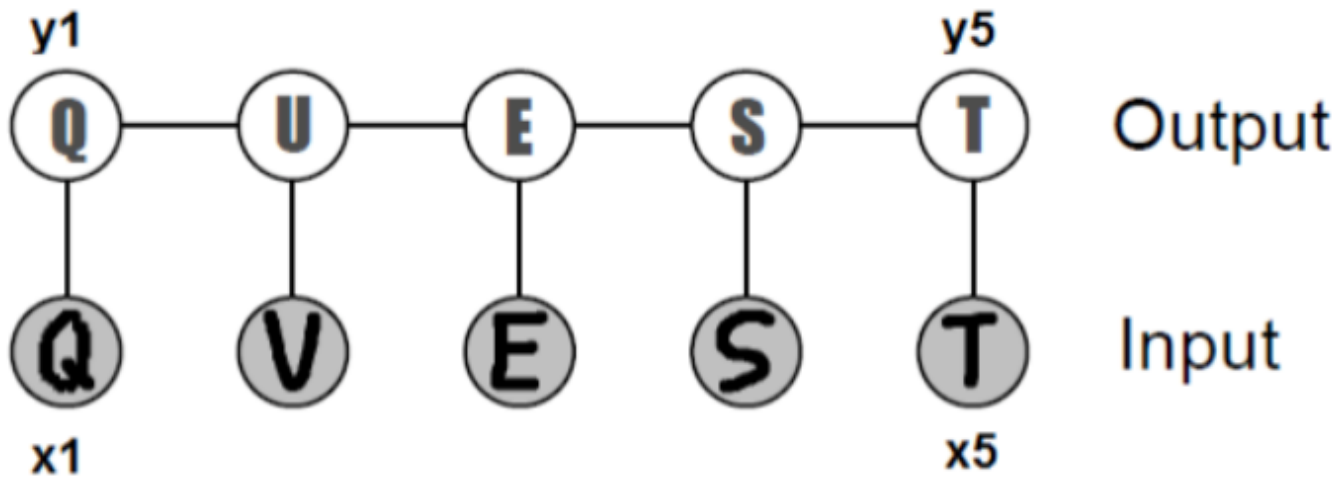

Figure: Chain-structured conditional random field for optical character recognition.

2. The Challenge of MAP Inference.

· Input : A sequence of character images $x_i \in [0,1]^{d \times d}$ in the form of *pixel matrices*.

· Output : A sequence of alphabet letters $y_i \in \{\mathrm{a}, \mathrm{b}, \cdots, \mathrm{z}\}$.

$\rightarrow$ MAP inference in this setting amounts to jointly recognizing the sequence of most likely words $(y_i)_{i=1}^n$ encoded by character images.

- Image segmentation : We are interested in locating an entity in a given image and label all its pixels.

  · Input : A matrix of image pixels $x \in [0,1]^{d \times d}$.

  · Output : The label matrix $y \in \{0,1\}^{d \times d}$, indicating whether each pixel encodes the object we want to recover.

2. The Challenge of MAP Inference.

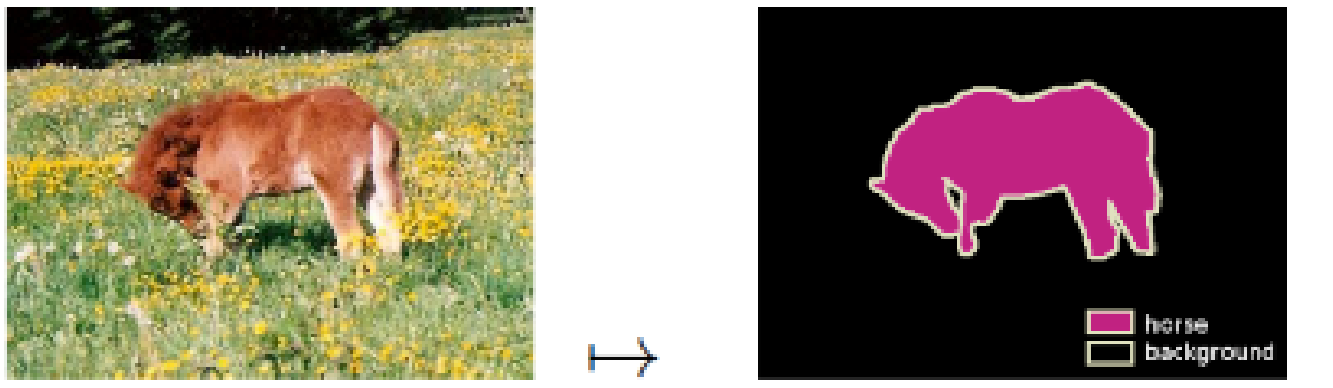

Figure: An illustration of the image segmentation problem.

3. Graph-cuts.

$*$ **Method 2. Graph-theoretical Approach.**

We will start our discussion with an efficient exact MAP inference algorithm for certain **Potts models** (= generalization of the Ising model).

$\rightarrow$ This algorithm will be computationally tractable even when the model has *large treewidth*.

Let's consider an undirected graph $G := (\mathcal{V}, \mathcal{E})$ and a binary Gibbs $p(x)$ distribution over G,

$$p(x) = \frac{1}{Z} \exp\{-E(x)\}, \; x = (x_v)_{v \in \mathcal{V}} \in \mathcal{X} := \{0,1\}^{\mathcal{V}}.$$

3. Graph-cuts.

Suppose the energy function $E : \mathcal{X} \to \mathbb{R}$ is given by

$$E(x) := \sum_{i \in \mathcal{V}} \epsilon_i(x_i) + \sum_{\{i,\, j\} \in \mathcal{E}} \epsilon_{ij}(x_i, x_j), \;\; \epsilon_{ij}(x_i, x_j) := \lambda_{ij} \cdot (1 - \delta_{x_i x_j}),$$

where $\lambda_{ij} \geq 0$ is a cost that penalizes edge mismatches. WLOG, we further assume that [either $\epsilon_i(0) = 0$ or $\epsilon_i(1) = 0$] and $\epsilon_i \geq 0$ for every $i \in \mathcal{V}$.

$\to$ To solve the MAP inference problem for this model, it suffices to find a variable assignment that minimizes the energy $E(x)$.

3. Graph-cuts.

§ Some Basic Notions in Graph Theory.

Definition

Let $G := (\mathcal{V}, \mathcal{E})$ be an undirected weighted graph with a weight $w : \mathcal{E} \to \mathbb{R}$.

1. A cut of G is a partition $C := (S, T)$ of V into two non-empty subsets S and T.

2. The cut-set of a cut $C = (S, T)$ is the set of edges that have one endpoint in S and the other endpoint in T. :

$$\{\{s, t\} \in \mathcal{E} : s \in S, t \in T\}$$

3. The weight or cost of a cut $C = (S, T)$ is the sum of weights of edges belong to the cut-set of C.

3. Graph-cuts.

4. The minimum cut problem is defined as following. :

· Input : An undirected weighted graph $G = (\mathcal{V}, \mathcal{E}, w)$.

· Output : The minimum cut $C = (S, T)$ (= the cut of G with the minimum cost).

5. If s and t are specified vertices of G, then an s-t cut is a cut of G that separates s and t.

3. Graph-cuts.

§ Several Algorithms for Minimum Cut Problem.

1. **Karger's algorithm** : With high probability, we can find all minimum cuts in the running time of $O((|V|^2 \log |V|) \cdot |E|)$.

2. **Karger-Stein algorithm** : With high probability, we can find all minimum cuts in the running time of $O(|V|^2 (\log |V|)^3))$.

3. **Stoer-Wagner algorithm** : Total time complexity $=$ $O(|V||E| + |V|^2 \log |V|)$.

Also, there exist algorithms based on the *maximum flow problem* and more...

3. Graph-cuts.

Definition (Augmented graph)

Given an undirected weighted graph $G = (\mathcal{V}, \mathcal{E}, w)$, the augmented graph $G' := (\mathcal{V}', \mathcal{E}', w')$ of G is defined as following. :

- $\mathcal{V}' := \mathcal{V} \cup \{s, t\}$: the source node s and the sink node t.
- Let $\mathcal{U} := \{u \in \mathcal{V} : \epsilon_u(0) = 0\}$. Then, we define $\mathcal{E}'$ by

$$\mathcal{E}' := \mathcal{E} \cup \{\{s, u\} : u \in \mathcal{U}\} \cup \{\{t, v\} : v \in \mathcal{V} \setminus \mathcal{U}\}.$$

- First, set $w'|_{\mathcal{E}} = w$. Also, let's define as

$$w'(\{s, u\}) := \epsilon_u(1), u \in \mathcal{U}$$
$$w'(\{t, v\}) := \epsilon_v(0), v \in \mathcal{V} \setminus \mathcal{U}$$

3. Graph-cuts.

§ Formulation of MAP Inference as a Min-Cut Problem.

1. (The cost of a minimum cut in the augmented graph G') = (The minimum energy in the model).
2. Let $C = (S, T)$ be a minimum cut of G'. Then, we have

$$S = \{s\} \cup \{v \in \mathcal{V} : x_v = 0\}, \;\; T = \{t\} \cup \{v \in \mathcal{V} : x_v = 1\}.$$

$\rightarrow$ The edges between nodes that disagree are precisely the ones that are in the minimum cut $C = (S, T)$.

3. Graph-cuts.

Example

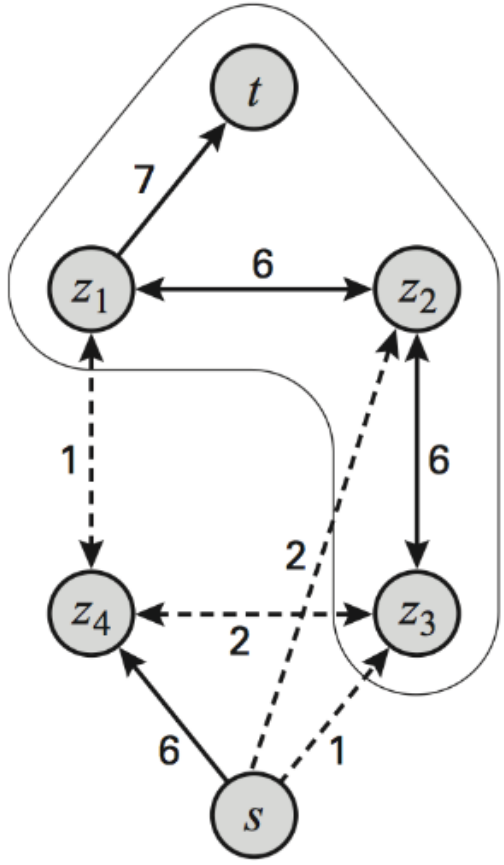

Figure: MAP inference $\rightarrow$ Min-cut problem in the augmented graph.

4. Linear Programming-based Approaches.

4.1. Linear Programming.

- Graphcut-based methods : Solve the MAP inference problem **exactly**, but they are only applicable in certain restricted class of MRFs.
- Linear programming-based methods : Solve the MAP inference problem **approximately**, but they are applicable for much larger classes of graphical models.

$\star$ Our strategy : MAP inference problem $\rightarrow$ Integer LP.

4. Linear Programming-based Approaches.

4.1. Linear Programming.

Definition (Linear programming (LP))

Linear programmings are optimization problems that can be expressed in canonical form as

$$\min \text{ (max) } c^T x$$
$$\text{s.t. } Ax \leq b,$$

where $x \in \mathbb{R}^n$ is a variable (usually we additionally restrict it to be $x \geq 0$), $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$.

4. Linear Programming-based Approaches.

4.1. Linear Programming.

Definition (Integer linear programming (ILP))

Integer linear programmings are optimization problems that can be expressed in canonical form as

$$\min \text{ (max) } c^T x$$

$$\text{s.t. } Ax \leq b,$$

where $x \in \mathbb{Z}^n$ is a variable (usually we additionally restrict it to be $x \geq 0$), $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$.

4. Linear Programming-based Approaches.

4.1. Linear Programming.

Remark

1. In many cases, we consider a particular problem which is called a 0-1 integer linear programming. : We additionally require that $x \in \{0,1\}^n$ to a general ILP.
2. The above additional requirement makes the optimization considerably more difficult. $\rightarrow$ ILP is NP-complete in general.
3. **Rounding** : One of the main techniques to solve ILP problems. First, relax the requirement $x \in \{0,1\}^n$ into $0 \leq x \leq 1$ (LP relaxation), solve the resulting LP problem, and then round the LP solution to its nearest integer value.

4. Linear Programming-based Approaches.

4.2. Formulating MAP Inference as ILP.

For simplicity, let's consider MAP in a binary pairwise MRF with the energy $E(x)$ and the corresponding Gibbs distribution $p(x)$. :

$$E(x) := \sum_{i \in \mathcal{V}} \epsilon_i(x_i) + \sum_{\{i,\, j\} \in \mathcal{E}} \epsilon_{ij}(x_i, x_j), \;\; x \in \mathcal{X} := \{0,1\}^{\mathcal{V}}$$

$$p(x) := \frac{1}{Z(\beta)} \exp\{-\beta E(x)\}$$

In this case, the MAP inference problem reduces to the following optimization problem. :

$$\max \left\{ \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{\{i,\, j\} \in \mathcal{E}} \theta_{ij}(x_i, x_j) : x \in \mathcal{X} \right\}.$$

4. Linear Programming-based Approaches.

4.2. Formulating MAP Inference as ILP.

Let's introduce two types of decision variables. :

- A variable $\mu_i(x_i)$ for each $i \in \mathcal{V}$ and state $x_i \in \{0, 1\}$.
- A variable $\mu_{ij}(x_i, x_j)$ for each edge $\{i, j\} \in \mathcal{E}$ and pair of states $x_i, x_j \in \{0, 1\}$.

Then, we can rewrite the objective function in MAP problem in terms of these variables. :

$$\max \left\{ \sum_{i \in \mathcal{V}} \sum_{x_i} \theta_i(x_i)\mu_i(x_i) + \sum_{\{i,\, j\} \in \mathcal{E}} \sum_{x_i, x_j} \theta_{ij}(x_i, x_j)\mu_{ij}(x_i, x_j) : \mu \right\}$$

In this case, what is the *appropriate mathematical formulation* of constraints for $\mu_i(x_i)$ and $\mu_{ij}(x_i, x_j)$?

4. Linear Programming-based Approaches.

4.2. Formulating MAP Inference as ILP.

The constraints for variables $\mu_i(x_i)$ and $\mu_{ij}(x_i, x_j)$ are as following. :

- We need to force each cluster to choose a *local assignment*.
  1. $\mu_i(x_i) \in \{0, 1\},\ \forall i \in \mathcal{V}, x_i \in \{0, 1\}.$
  2. $\sum_{x_i=0}^{1} \mu_i(x_i) = 1,\ \forall i \in \mathcal{V}.$
  3. $\mu_{ij}(x_i, x_j) \in \{0, 1\},\ \forall \{i, j\} \in \mathcal{E}, x_i, x_j \in \{0, 1\}.$
  4. $\sum_{x_i=0}^{1} \sum_{x_j=0}^{1} \mu_{ij}(x_i, x_j) = 1,\ \forall \{i, j\} \in \mathcal{E}.$
- These assignments must be *consistent*.
  1. $\sum_{x_i=0}^{1} \mu_{ij}(x_i, x_j) = \mu_j(x_j),\ \forall \{i, j\} \in \mathcal{E}, x_j \in \{0, 1\}.$
  2. $\sum_{x_j=0}^{1} \mu_{ij}(x_i, x_j) = \mu_i(x_i),\ \forall \{i, j\} \in \mathcal{E}, x_i \in \{0, 1\}.$

Together, these constraints along with the above objective function yield an ILP, whose solution equals the **MAP assignment**.

4. Linear Programming-based Approaches.

4.2. Formulating MAP Inference as ILP.

Remark

1. This ILP is still NP-hard, but we can obtain an approximate solution by transform this ILP into an (easy to solve) LP via *relaxation*.
2. If the underlying graph $G = (\mathcal{V}, \mathcal{E})$ of a given binary pairwise MRF is a **tree**, then it is well-known that the relaxed LP is guaranteed to always return *integer solutions* and it becomes an optimal solution of the original ILP.

5. Dual Decomposition.

5.1. Introduction.

In this section, we will look at another way to transform the MAP objective into a more *amenable* optimization problem.

Consider n discrete variables $x_1, \cdots, x_n$, where $\mathcal{X}$ is their state space. Also, let F be a set of non-empty subsets of $\mathcal{V} = [n]$. Assume that the factors $\theta_i(x_i)$, $i \in \mathcal{V}$ and $\theta_f(x_f)$, $f \in F$ are given, where $x_f := (x_i : i \in f)$, $f \in F$. Then, our task is solving the following optimization problem :

$$\max \left\{ \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{f \in F} \theta_f(x_f) : x \in \mathcal{X}^n \right\}.$$

5. Dual Decomposition.

5.1. Introduction.

Let p^* denote the optimal value of this objective function and x^* denote the optimal assignment. This objective function is *difficult* to optimize because the factors are coupled. Thus, we want to consider an alternative objective functions when we optimize the factors separately! :

$$\max\ \theta_i(x_i),\ i \in \mathcal{V}\ /\ \max\ \theta_f\left(x_f^f\right),\ f \in F$$

subject to

- $x_i \in \mathcal{X}, \forall i \in \mathcal{V}$.
- $x_f^f \in \mathcal{X}^{|f|}, \forall f \in F$.
- $x_i^f = x_i, \forall i \in f, f \in F$.

5. Dual Decomposition.

5.1. Introduction.

We can solve this optimization problem by using the **Lagrange multiplier method**. Then, the *Lagrangian* for the problem is given by

$$\mathcal{L}\left(\delta, \mathbf{x}^F, \mathbf{x}\right) := \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{f \in F} \theta_f\left(x_f^f\right) + \sum_{f \in F} \sum_{i \in f} \sum_{\hat{x}_i \in \mathcal{X}} \delta_{fi}\left(\hat{x}_i\right) \left[1_{\{\hat{x}_i = x_i\}} - 1_{\left\{\hat{x}_i = x_i^f\right\}}\right],$$

where $\delta = \left(\delta_{fi}\left(\hat{x}_i\right) : f \in F, i \in f, \hat{x}_i \in \mathcal{X}\right)$ and $\mathbf{x}^F = \left(x_f^f : f \in F\right)$, $\mathbf{x} = \left(x_i : i \in \mathcal{V}\right)$. The variables $\delta_{fi}\left(\hat{x}_i\right)$ are called **Lagrange multipliers**.

5. Dual Decomposition.

5.1. Introduction.

Observation. $\mathcal{L}(\delta) := \max\left\{\mathcal{L}\left(\delta, \mathbf{x}^F, \mathbf{x}\right) : \mathbf{x}^F, \mathbf{x}\right\} \geq p^*$ for all $\delta \in \mathbb{R}^{|\mathcal{X}|\cdot\left(\sum_{f\in F}|f|\right)} =: \Delta$ (= Weak Duality).

Here, the function $\mathcal{L}(\delta)$ is called the **relaxed Lagrange dual function**. In order to get the *tightest* such bound, we need to optimize the relaxed dual function $\mathcal{L}(\delta)$ over δ. Hence, we can consider the following *dual problem* :

$$\min_{\delta\in\Delta}\mathcal{L}(\delta) = \sum_{i\in\mathcal{V}}\max\left\{\theta_i(x_i) + \sum_{f\in F: i\in f}\delta_{fi}(x_i) : x_i \in \mathcal{X}\right\}$$
$$+\sum_{f\in F}\max\left\{\theta_f\left(x_f^f\right) - \sum_{i\in f}\delta_{fi}\left(x_i^f\right) : x_f^f \in \mathcal{X}^{|f|}\right\}.$$

5. Dual Decomposition.

5.1. Introduction.

$\star$ Re-parametrization Interpretation of the Dual Function.

Given a set of dual variables δ, define *new* factors on x_i, $i \in \mathcal{V}$ and x_f, $f \in F$ given by :

$$\bar{\theta}_i^{\delta}(x_i) := \theta_i(x_i) + \sum_{f \in F : i \in f} \delta_{fi}(x_i),$$

$$\bar{\theta}_f^{\delta}(x_f) := \theta_f(x_f) - \sum_{i \in f} \delta_{fi}(x_i).$$

Then, we can see that

$$\sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{f \in F} \theta_f(x_f) = \sum_{i \in \mathcal{V}} \bar{\theta}_i^{\delta}(x_i) + \sum_{f \in F} \bar{\theta}_f^{\delta}(x_f).$$

5. Dual Decomposition.

5.1. Introduction.

Also, we can rewrite the relaxed dual function as following :

$$\mathcal{L}(\delta) = \sum_{i \in \mathcal{V}} \max \left\{ \bar{\theta}_i^{\delta}(x_i) : x_i \in \mathcal{X} \right\} + \sum_{f \in F} \max \left\{ \bar{\theta}_f^{\delta}(x_f) : x_f \in \mathcal{X}^{|f|} \right\},$$

and let $\delta^* := \arg\min \{ \mathcal{L}(\delta) : \delta \in \Delta \}$.

The *weak duality* holds for general dual problems, but the *strong duality* does not hold in general. However, the following theorem says that the *strong duality* holds for some functions $\theta(x)$ in the above Lagrange dual problem.

5. Dual Decomposition.

5.1. Introduction.

Theorem (Strong Duality)

Suppose the functions $\theta(x)$ satisfy the following property. :

There exist $\delta^* \in \Delta, x^* \in \mathcal{X}^n$ such that $x_i^* \in \arg\max_{x_i} \bar{\theta}_i^{\delta^*}(x_i)$, $\forall i \in \mathcal{V}$ and $x_f^* \in \arg\max_{x_f} \bar{\theta}_f^{\delta^*}(x_f)$, $\forall f \in F$.

Then, an upper bound of p^* will be *exactly tight*, *i.e.*, $\mathcal{L}(\delta^*) = p^*$.

5. Dual Decomposition.

5.1. Introduction.

Proof.

From the assumption, we have

$$\begin{aligned}\mathcal{L}(\delta^*) &= \sum_{i\in\mathcal{V}} \bar{\theta}_i^{\delta^*}(x_i^*) + \sum_{f\in F} \bar{\theta}_f^{\delta^*}(x_f^*) \\ &= \sum_{i\in\mathcal{V}} \theta_i(x_i^*) + \sum_{f\in F} \theta_f(x_f^*) \\ &\leq \max\left\{\sum_{i\in\mathcal{V}} \theta_i(x_i) + \sum_{f\in F} \theta_f(x_f) : x\in\mathcal{X}^n\right\} = p^*.\end{aligned}$$

By combining the weak duality of the relaxed Lagrange dual function $\mathcal{L}(\delta)$, we obtain the desired result. ■

5. Dual Decomposition.

5.2. Minimizing the Objective.

There exist several ways of evaluating $\mathcal{L}(\delta^*)$, of which we will give a *brief overview*. :

1. Subgradient Descent Method.

First, note that the dual function $\mathcal{L}(\delta)$ is convex, since it is a pointwise max of a set of *affine functions*. Because the objective function $\mathcal{L}(\delta)$ is continuous and convex, we may minimize it by using the **subgradient descent method**. The subgradient descent method is similar to *gradient descent method*, but is applicable to non-differentiable convex functions.

5. Dual Decomposition.

5.2. Minimizing the Objective.

Definition (Subgradient)

Suppose U is a convex open subset of $\mathbb{R}^n$ and $f : U \to \mathbb{R}$ is a convex function. A vector $v \in \mathbb{R}^n$ is called a **subgradient** of f at $x_0 \in U$ if we have

$$f(x) - f(x_0) \geq v \cdot (x - x_0), \ \forall x \in U.$$

The set of all subgradients of f at $x_0 \in U$ is called the **subdifferential** of f at x_0 and it is denoted by $\partial f(x_0)$. It is well-known that the subdifferential of a convex function is always a non-empty convex compact set in $\mathbb{R}^n$.

5. Dual Decomposition.

5.2. Minimizing the Objective.

Algorithm (Classical Subgradient Descent Method)

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. A classical subgradient descent method iterates

$$x^{(t+1)} = x^{(t)} - \alpha_t g^{(t)},$$

where $g^{(t)} \in \partial f\left(x^{(t)}\right)$ and α_t is a step-size that may depend on t. It may happen that $-g^{(t)}$ is not a *descent direction* for f at $x^{(t)}$. Therefore, we maintain a list f_{best} that keeps track of the lowest(or the highest) objective function value, *i.e.*,

$$f_{best}^{(t+1)} = \min \text{ or } \max\left\{f_{best}^{(t)}, f\left(x^{(t)}\right)\right\}.$$

5. Dual Decomposition.

5.2. Minimizing the Objective.

⋆ Tuning Step-size Parameters.

The five step-size rules which *convergence* proofs are known. :

1. Constant step-size. : $\alpha_t = \alpha$, $\forall t \geq 1$.
2. Constant step length. : $\alpha_t = \frac{\gamma}{||g^{(t)}||_2}$, which gives $||x^{(t+1)} - x^{(t)}||_2 = \gamma$ for all $t \geq 1$.
3. Square-summable but not summable step-sizes. : $\alpha_t \geq 0$, $\forall t \geq 1$ and $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$, $\sum_{t=1}^{\infty} \alpha_t = \infty$.
4. Non-summable diminishing step-sizes. : $\lim_{t\to\infty} \alpha_t = 0$ and $\sum_{t=1}^{\infty} \alpha_t = \infty$.
5. Non-summable diminishing step lengths. : $\alpha_t = \frac{\gamma_t}{||g^{(t)}||_2}$, where $\gamma_t \geq 0$, $\forall t \geq 1$ and $\lim_{t\to\infty} \gamma_t = 0$, $\sum_{t=1}^{\infty} \gamma_t = \infty$.

5. Dual Decomposition.

5.2. Minimizing the Objective.

Now, we need to discuss how to calculate the subgradient of $\mathcal{L}(\delta)$, completing the description of the subgradient algorithm.

Algorithm (Computing Subgradient of the Dual Function)

Let δ^t be the current dual variable.

1. Choose a maximizing assignment for each sub-problem. :

$$\bar{x}_i \in \arg\max_{x_i} \bar{\theta}_i^{\delta^t}(x_i) \text{ and } \bar{x}_f^f \in \arg\max_{x_f} \bar{\theta}_f^{\delta^t}(x_f).$$

2. The subgradient of $\mathcal{L}(\delta)$ at δ^t is given by the following pseudocode. :

5. Dual Decomposition.

5.2. Minimizing the Objective.

For $f \in F$ and $i \in f$:

If $\bar{x}_i^f \neq \bar{x}_i$:

$$g_{fi}^{(t)}(\bar{x}_i) = +1.$$

$$g_{fi}^{(t)}\left(\bar{x}_i^f\right) = -1.$$

Otherwise :

$$g_{fi}^{(t)}(\bar{x}_i) = g_{fi}^{(t)}\left(\bar{x}_i^f\right) = 0.$$

5. Dual Decomposition.

5.2. Minimizing the Objective.

2. Block Coordinate Descent Method.

An alternative way of minimizing $\mathcal{L}(\delta)$ is via **block coordinate descent method**. A typical way of forming *blocks* is to consider all the variables $\delta_{fi}(x_i)$ associated with a fixed factor $f \in F$.

$\rightarrow$ This results in updates that are very similar to *loopy max-product belief propagation* algorithm.

5. Dual Decomposition.

5.2. Minimizing the Objective.

⋆ Advantages of this method. :

- In practice, this method may be faster than the subgradient descent algorithm.
- It is guaranteed to decrease the objective at every step.
- It does not require tuning *step-size parameters*.

⋆ Drawbacks of this method. :

- It does not find the *global minimum*, since the objective function is not **strongly convex** in general.

5. Dual Decomposition.

5.2. Minimizing the Objective.

Definition (Strong Convexity)

A differentiable function $f : U \subseteq \mathbb{R}^n \to \mathbb{R}$ is **strongly convex** if

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2} ||y - x||^2$$

for some $\mu > 0$ and all $x, y \in U$.

Note that the strong convexity does not require the differentiability of the function, and the gradient is replaced by the *subgradient* if the function is non-smooth.

6. Other Methods.

- Local Search.
- Branch and Bound.
- Simulated Annealing. : Use sampling methods (*e.g.* Metropolis-Hastings algorithm) to sample from

$$p_t(x) \propto \exp\left(\frac{1}{t}\sum_{c\in\mathcal{C}}\theta_c\left(x_c\right)\right).$$

The parameter t is called the temperature. The idea of *simulated annealing* is to run a sampling algorithm starting with a high t, and gradually decrease it, as the algorithm is being run.

References.

1. "MAP Inference", Volodymyr Kuleshov and Stefano Ermon, Lecture Materials on the course CS228, Stanford University. : https://ermongroup.github.io/cs228-notes/inference/map/
2. "Introduction to Dual Decomposition for Inference", David Sontag, Amir Globerson and Tommi Jaakkola.
3. "Convex Optimization", Stephen Boyd and Lieven Vandenberghe.