TP n° 1 : Apprentissage de métrique et gradient stochastique

Introduction

Le but de cette séance est la mise en œuvre des techniques d'apprentissage de métrique basées sur des algorithmes de type descente de gradient stochastique (SGD). Un notebook Python vous est fourni (tp_metric_learning.ipynb) pour guider les réponses, avec néanmoins certaines parties manquantes qu'il faut compléter.

Définitions et notations

On rappelle ici le cadre de l'apprentissage supervisé, et l'on présente les notations que l'on utilisera :

- $\mathcal{Y}$ l'ensemble des étiquettes des données (*labels* en anglais) : $\mathcal{Y} = \{1, \dots, C\}$
- $\mathbf{x} = (x_1, \dots, x_d)^\top \in \mathcal{X} \subset \mathbb{R}^d$ est une observation, un exemple, un point (ou un *sample* en anglais). La j-ème coordonnée de $\mathbf{x}$ est la valeur prise par la j-ème variable (*feature* en anglais).
- $\mathcal{D}_n = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i = 1, \dots n\}$, ensemble d'apprentissage contenant n exemples et leurs étiquettes.
- Il existe un modèle probabiliste qui gouverne la génération des observations selon des variables aléatoires X et Y : $\forall i \in \{1, \dots, n\}, (\mathbf{x}_i, y_i) \overset{i.i.d}{\sim} (X, Y)$ et l'on note $\mathbb{E}$ l'espérance associée.
- On cherche à construire à partir de l'ensemble d'apprentissage $\mathcal{D}_n$ une fonction appelée classifieur, $\hat{f} : \mathcal{X} \to \mathcal{Y}$ qui pour un nouveau point $\mathbf{x}_{\text{new}} \in \mathcal{X}$ fournit une étiquette $\hat{f}(\mathbf{x}_{\text{new}})$.
- On mesure la performance d'un classifieur, pour une perte $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, par le risque $\mathbb{E}(\ell(\hat{f}(\mathbf{x}), y))$. En pratique, cette quantité n'est pas calculable, on se sert donc de contrepartie empirique du type $\frac{1}{n} \sum_{i=1}^{n} \ell(\hat{f}(\mathbf{x}_i), y_i)$.

Rappels sur l'apprentissage de métrique

Beaucoup de méthodes en apprentissage statistique s'appuient sur une mesure de distance entre les observations. C'est le cas par exemple de la classification par plus proches voisins ou encore du clustering K-Means. Le choix d'une distance (ou métrique) appropriée est crucial pour la performance de ces méthodes. L'apprentissage de métrique [Kul12, BHS13] consiste à apprendre automatiquement une mesure de distance (ou de similarité) à partir de paires d'observations que l'on sait similaires ou dissimilaires pour la tâche considérée.

On s'intéresse ici à la famille de distance dite "de Mahalanobis" qui est définie de la manière suivante :

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X} \times \mathcal{X}, \quad d^2_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}'), \tag{1}$$

avec $\mathbf{M} \in \mathbb{S}^d_+$, où $\mathbb{S}^d_+$ est l'ensemble des matrices $d \times d$ symétriques semi-définies positives. Ceci permet de garantir que la fonction $d_{\mathbf{M}}$ est une pseudo-distance, c'est-à-dire qu'elle est positive, symétrique et satisfait l'inégalité triangulaire (par contre elle ne satisfait pas toujours la séparation : on peut trouver deux points $\mathbf{z} \neq \mathbf{z}'$ avec $d_{\mathbf{M}}(\mathbf{z}, \mathbf{z}') = 0$). On peut noter que si $\mathbf{M} = \mathbf{I}_d$, $d_{\mathbf{M}}$ correspond à la distance Euclidienne. L'objectif est ici d'optimiser la matrice $\mathbf{M}$ en utilisant l'ensemble d'apprentissage afin d'obtenir une distance adaptée au problème considéré.

Approche basée sur les paires

Dans cette partie, on va considérer une formulation de l'apprentissage de métrique qui utilise des paires d'exemples similaires ou dissimilaires. Introduisons tout d'abord quelques notations. On associe à

toute paire d'exemples $(\mathbf{x}_i, \mathbf{x}_j)$, $1 \leq i, j \leq n$, un label y_{ij} défini par

$$y_{ij} = \begin{cases} 1 & \text{si } y_i = y_j, \\ -1 & \text{sinon.} \end{cases}$$

On va formuler l'apprentissage de métrique sous la forme du problème d'optimisation suivant :

$$\min_{\mathbf{M} \in \mathbb{S}_+^d} \quad \frac{1}{n^2} \sum_{i,j=1}^{n} \ell(d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j), y_{ij}) = \frac{1}{n^2} \sum_{i,j=1}^{n} \ell_{i,j}(\mathbf{M}),$$

où $\ell : \mathbb{R}_+ \times \{-1, 1\} \to \mathbb{R}$ est une fonction de perte sur les paires. L'algorithme ci-dessous est une descente de gradient stochastique appliquée à ce problème. Notons qu'une étape de projection est nécessaire pour maintenir la matrice itérée dans l'ensemble admissible $\mathbb{S}_+^d$. Cette projection est définie par

$$\begin{aligned} \Pi_{\mathbb{S}_+^d} : \mathbb{S}^d &\to \mathbb{S}_+^d, \\ \mathbf{M} &\mapsto \Pi_{\mathbb{S}_+^d}(\mathbf{M}) = \operatorname*{arg\,min}_{S \in \mathbb{S}_+^d} \|\mathbf{S} - \mathbf{M}\|_F, \end{aligned}$$

où $\mathbb{S}^d$ est l'ensemble des matrices symétriques de taille $d \times d$ et $\|M\|_F = \sqrt{\sum_{ij} M_{ij}^2}$ est la norme de Frobenius. La projection $\Pi_{\mathbb{S}_+^d}$ est décrite dans la fonction psd_proj du notebook. [1]

Algorithme 1 : Algorithme du gradient stochastique pour l'apprentissage de métrique

Data : les observations et leurs étiquettes $\mathcal{D}_n = \{(\mathbf{x}_i, y_i) : 1 \leq i \leq n\}$
nombre maximal d'itérations : T
suite de pas de gradient : $(\gamma_t)_{t=1,\dots,T}$
Result : $\mathbf{M}_T$
initialiser (aléatoirement) $\mathbf{M}_0 \in \mathbb{S}_+^d$; initialiser $t = 0$
while $t \leq T$ **do**
  tirer aléatoirement (i, j) dans $\{1, \dots, n\}^2$
  $\mathbf{M}_{t+1} \leftarrow \Pi_{\mathbb{S}_+^d}\left[\mathbf{M}_t - \gamma_t \nabla_{\mathbf{M}} \ell_{i,j}(\mathbf{M})\right]$
  $t \leftarrow t + 1$
optionnellement : $\bar{\mathbf{M}}_T \leftarrow \frac{1}{T} \sum_{t=1}^{T} \mathbf{M}_t$

1) Une fonction de perte très utilisée en apprentissage de métrique est basée sur la perte hinge :

$$\ell(d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j), y_{ij})) = \max(0, 1 + y_{ij}(d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - 2)). \tag{2}$$

   Quelle est l'interprétation de cette fonction de perte ?

2) Coder cet algorithme pour la fonction de perte hinge donnée par (2) en complétant la fonction sgd_metric_learning_pairs. Vous n'avez qu'à ajouter le calcul du sous-gradient de la perte. Un sous-gradient est calculé de la manière suivante :

$$\nabla_{\mathbf{M}} \ell_{i,j}(\mathbf{M}) = \begin{cases} 0 & \text{si } 1 + y_{ij}(d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - 2)) \leq 0, \\ y_{ij}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T & \text{sinon.} \end{cases}$$

   Astuce : pour deux vecteurs $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$, on peut calculer $\mathbf{a}\mathbf{b}^T \in \mathbb{R}^{d \times d}$ avec la function np.outer(a, b).

3) Tester l'algorithme sur les jeux de données Iris et Digits (on peut passer de l'un à l'autre en changeant le 1 en 0 en ligne 14 du fichier) et adapter le pas et le nombre d'itérations afin de le faire converger. Notez que pour le jeu de données Digits, vous pouvez modifier le nombre de classes considérées afin de changer la difficulté du problème.

1. Une preuve de ce résultat est donnée en annexe du sujet pour les curieux.

4) La factorisation de Cholesky de la matrice $\mathbf{M}$ donne une matrice $\mathbf{L}$ telle que $\mathbf{M} = \mathbf{L}\mathbf{L}^T$. On peut ainsi interpréter la distance apprise comme une distance Euclidienne après transformation linéaire des données par la matrice $\mathbf{L}$:

$$d^2_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T\mathbf{M}(\mathbf{x} - \mathbf{x}') = (\mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{x}')^T(\mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{x}') = \|\mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{x}'\|_2^2.$$

Utiliser le code fourni pour transformer les données et comparer visuellement les deux représentations des données en dimension 2 (en utilisant une ACP).

5) La distance apprise peut être utilisée pour prédire si deux exemples appartiennent ou non à la même classe. C'est une approche très populaire dans le domaine de la vérification faciale par exemple. Étant donné un seuil $\tau \geq 0$, on prédit $y_i = y_j$ si $d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \leq \tau$ et $y_i \neq y_j$ sinon. La valeur de τ a un impact sur le nombre de faux positifs et de vrais positifs. Pour un échantillon aléatoire de paires d'apprentissage, calculer les valeurs de distance et utilisez Scikit-Learn pour comparer la courbe ROC de la distance apprise avec celle de la distance Euclidienne. Calculer également l'aire sous la courbe ROC (AUC).

http://scikit-learn.org/stable/modules/model_evaluation.html#receiver-operating-characteristic-roc
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html

D'autre part, appliquer un knn sur les données transformées obtenues en question 4) et comparer (sur une partie test) la performance de prédiction par rapport à la distance euclidienne.

6) Modifiez votre code afin de pouvoir utiliser plusieurs paires à chaque itération pour calculer le gradient (mini-batch SGD). Quel est l'effet de la taille du mini-batch sur la convergence de l'algorithme ? On utilisera l'approche qui consiste à tirer b paires aléatoirement (Bonus : comparer avec l'autre approche qui consiste à tirer m observations aléatoirement, puis à former les $m(m-1)/2$ paires associées).

7) On a vu en cours qu'utiliser une régularisation basée sur la norme trace de $\mathbf{M}$ permet de favoriser des solutions de rang faible et ainsi la réduction de la dimension des données. Cette norme est définie par $\|\mathbf{M}\|_* = \sum_{i=1}^p \lambda_i(\mathbf{M})$, où $\lambda_i(\mathbf{M})$ est la i-ème valeur propre de $\mathbf{M}$. Quand $\mathbf{M}$ est symétrique semi-définie positive, on a une expression plus simple et facilement dérivable : $\|\mathbf{M}\|_* = \mathrm{tr}(\mathbf{M}) = \sum_{i=1}^p \mathbf{M}_{i,i}$. On veut donc résoudre la version régularisée du problème précédent :

$$\min_{\mathbf{M} \in \mathbb{S}_+^d} \quad \frac{1}{n^2} \sum_{i,j=1}^n \ell_{i,j}(\mathbf{M}) + \alpha\, \mathrm{tr}(\mathbf{M}),$$

où $\alpha \geq 0$ est un paramètre de régularisation. Modifiez votre code afin de prendre en compte la régularisation. Comment varie le nombre de valeurs propres nulles en fonction de α ? Peut-on trouver un bon compromis entre performance et réduction de dimension sur les datasets Iris et Digits ?

Approche basée sur les triplets

Dans cette partie, on va expérimenter avec une formulation d'apprentissage de métrique qui utilise cette fois des triplets d'exemples de la forme $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ avec $y_i = y_j \neq y_k$. On note $\mathcal{R}$ l'ensemble de ces triplets et on considère le problème d'optimisation suivant :

$$\min_{\mathbf{M} \in \mathbb{S}_+^d} \quad \sum_{(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in \mathcal{R}} \ell(\mathbf{M}, \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k),$$

où $\ell : \mathbb{S}_+^d \times \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ est une fonction de perte sur les triplets.

8) Une fonction de perte très utilisée pour les triplets est encore une fois basée sur la perte hinge :

$$\ell(\mathbf{M}, \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) = \max(0, 1 - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k) + d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)).$$

Quelle est l'interprétation de cette perte ?

9) Après avoir trouvé comment calculer un sous-gradient de cette perte, implémentez une version triplet de l'algorithme de descente de gradient stochastique. Vous n'avez qu'à ajouter le calcul du sous-gradient de la perte dans la fonction sgd_metric_learning_pairs. Comparez la performance en AUC de cette approche avec celle basée sur les paires.

Question bonus : OASIS

L'algorithme OASIS [CSSB09] est un algorithme en ligne pour apprendre une similarité bilinéaire $S_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{M} \mathbf{x}'$, où $\mathbf{M} \in \mathbb{R}^{d \times d}$ n'est pas nécessairement semi-définie positive (ni même forcément symétrique !). A chaque itération t, OASIS reçoit un triplet $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ et résout le sous-problème suivant :

$$\begin{aligned} \mathbf{M}^t = \quad & \underset{\mathbf{M}, \xi}{\arg\min} \quad \frac{1}{2}\|\mathbf{M} - \mathbf{M}^{t-1}\|_F^2 + C\xi \\ & \text{s.t.} \quad 1 - S_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + S_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k) \leq \xi \\ & \qquad \xi \geq 0 \end{aligned}$$

où $C \geq 0$ est un paramètre dit "d'agressivité".

La solution de ce sous problème s'écrit $\mathbf{M}^t = \mathbf{M}^{t-1} + \beta \mathbf{V}$, où $\mathbf{V} = \mathbf{x}_i(\mathbf{x}_j - \mathbf{x}_k)^T$ et

$$\beta = \min\left(C, \frac{\max(0, 1 - S_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + S_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k))}{\|\mathbf{V}\|_F^2}\right).$$

Implémentez OASIS et testez sa performance sur les datasets utilisés précédemment.

Références

[Ber99] D. P. Bertsekas. *Nonlinear programming.* Athena Scientific, 1999. 5

[BHS13] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A Survey on Metric Learning for Feature Vectors and Structured Data. Technical report, arXiv :1306.6709, June 2013. 1

[CSSB09] Gal Chechik, Uri Shalit, Varun Sharma, and Samy Bengio. An Online Algorithm for Large Scale Image Similarity Learning. In *Advances in Neural Information Processing Systems 22*, pages 306–314, 2009. 4

[Kul12] Brian Kulis. Metric Learning : A Survey. *Foundations and Trends in Machine Learning*, 5(4) :287–364, 2012. 1

Projection sur les matrices symétriques semi-définie positives

Soit $\mathbf{M} \in \mathbb{S}_+^d$. Pour tout $t \in \mathbb{R}$, on note $t_+ = \max(0, t)$ (resp. $t_- = \min(0, t)$), ainsi on a la décomposition en partie positive et négative suivante :

$$t = t_+ + t_-. \tag{3}$$

Définissons sa décomposition spectrale par $\mathbf{M} = \mathbf{U}^\top \mathbf{D}\mathbf{U}$ avec $\mathbf{U} \in \mathbb{R}^{d\times d}$ vérifiant $\mathbf{U}^\top\mathbf{U} = \mathbf{I}_d$ et $\mathbf{D} \in \mathbb{R}^{d\times d}$ est diagonale. On note $\mathbf{D} = \operatorname{diag}(d_1, \dots, d_p) = \mathbf{D}_+ + \mathbf{D}_-$, avec $\mathbf{D}_+ = \operatorname{diag}((d_1)_+, \dots, (d_p)_+) \in \mathbb{R}^{d\times d}$ (resp. $\mathbf{D}_- = \operatorname{diag}((d_1)_-, \dots, (d_p)_-) \in \mathbb{R}^{d\times d}$) la matrice diagonale contenant les termes positifs (resp. négatifs) de $\mathbf{D}$. On notera au passage que $\mathbf{D}_-\mathbf{D}_+ = 0$).

On rappelle maintenant que $\Pi_{\mathbb{S}_+^d} : \mathbb{S}^d \to \mathbb{S}_+^d$, $\Pi_{\mathbb{S}_+^d}(\mathbf{M}) = \arg\min_{\mathbf{S}\in\mathbb{S}_+^d} \|\mathbf{S} - \mathbf{M}\|_F$, où $\mathbb{S}^d$ est l'ensemble des matrices symétriques et $\|\cdot\|_F$ est la norme de Frobenius. On va montrer que $\Pi_{\mathbb{S}_+^d}(\mathbf{M}) = \mathbf{U}^\top\mathbf{D}_+\mathbf{U}$.

Pour cela on rappelle que pour le produit scalaire associé à la norme $\|\cdot\|_F$, et défini sur $\mathbb{S}_+^d$ par $\langle \mathbf{M}, \mathbf{N}\rangle = \operatorname{tr}(\mathbf{M}^\top\mathbf{N})$, $\Pi_{\mathbb{S}_+^d}(\mathbf{M})$ (projection sur le convexe fermé $\mathbb{S}_+^d$) est la seule matrice vérifiant :

$$\forall \mathbf{N} \in \mathbb{S}_+^d, \quad \langle \mathbf{M} - \Pi_{\mathbb{S}_+^d}(\mathbf{M}), \mathbf{N} - \Pi_{\mathbb{S}_+^d}(\mathbf{M})\rangle \le 0. \tag{4}$$

Nous allons donc vérifier que $\mathbf{M}_+ = \mathbf{U}^\top\mathbf{D}_+\mathbf{U}$ vérifie cette relation (*cf.* Figure 1 et [Ber99, Prop. 2.1.3] pour une preuve de cette propriété).

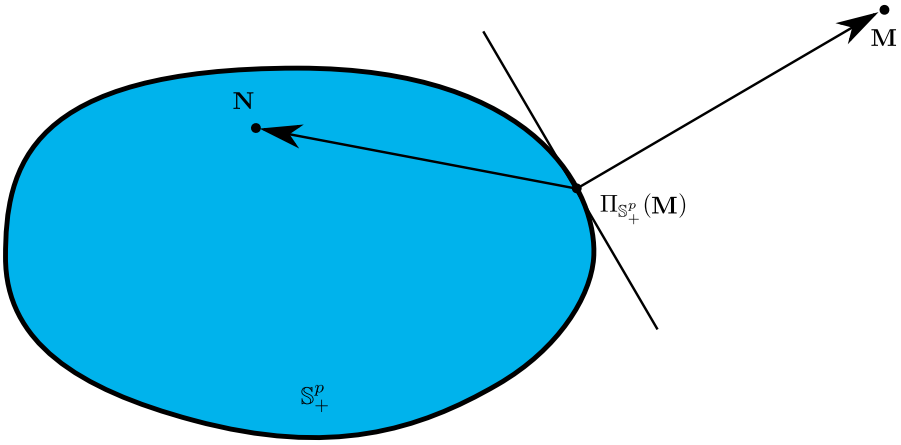

FIGURE 1 – Rappel visuel sur le théorème de projection sur les convexes fermés.

Prenons $\mathbf{N} \in \mathbb{S}_+^d$ quelconque

$$\begin{aligned}
\langle \mathbf{M} - \mathbf{M}_+, \mathbf{N} - \mathbf{M}_+\rangle &= \langle \mathbf{M} - \mathbf{U}^\top\mathbf{D}_+\mathbf{U}, \mathbf{N} - \mathbf{U}^\top\mathbf{D}_+\mathbf{U}\rangle,\\
&= \langle \mathbf{U}^\top\mathbf{D}_-\mathbf{U}, \mathbf{N} - \mathbf{U}^\top\mathbf{D}_+\mathbf{U}\rangle,\\
&= \langle \mathbf{U}^\top\mathbf{D}_-\mathbf{U}, \mathbf{N}\rangle - \langle \mathbf{U}^\top\mathbf{D}_-\mathbf{U}, \mathbf{U}^\top\mathbf{D}_+\mathbf{U}\rangle,\\
&= \langle \mathbf{U}^\top\mathbf{D}_-\mathbf{U}, \mathbf{N}\rangle - \langle \mathbf{U}, \mathbf{D}_-\mathbf{D}_+\mathbf{U}\rangle,\\
&= \langle \mathbf{U}^\top\mathbf{D}_-\mathbf{U}, \mathbf{N}\rangle,\\
&= \operatorname{tr}(\mathbf{U}^\top\mathbf{D}_-\mathbf{U}\mathbf{N}),\\
&= \operatorname{tr}(\mathbf{D}_-\mathbf{U}\mathbf{N}\mathbf{U}^\top),
\end{aligned}$$

Notons que $\mathbf{U}\mathbf{N}\mathbf{U}^\top$ est symétrique semi-définie positive (car $\mathbf{N}$ l'est), donc sa décomposition spectrale donne : $\mathbf{U}\mathbf{N}\mathbf{U}^\top = \sum_{j=1}^p \lambda_j v_j v_j^\top$ avec $\lambda_j \ge 0$ pour $j = 1, \dots, p$. Ainsi, en notant $e_k = (0, \dots, 0, \underbrace{1}_{k^{\text{e}}}, 0, \dots, 0)^\top$ et en utilisant la linéarité de la trace :

$$
\begin{aligned}
\langle \mathbf{M} - \mathbf{M}_+, \mathbf{N} - \mathbf{M}_+ \rangle &= \operatorname{tr}(\mathbf{D}_- \sum_{j=1}^{p} \lambda_j v_j v_j^\top), \\
&= \operatorname{tr}\left(\sum_{k=1}^{p} (d_k)_- e_k e_k^\top \sum_{j=1}^{p} \lambda_j v_j v_j^\top \right), \\
&= \operatorname{tr}\left(\sum_{k=1}^{p} \sum_{j=1}^{p} \lambda_j (d_k)_- e_k^\top v_j v_j^\top e_k \right), \\
&= \sum_{k=1}^{p} \sum_{j=1}^{p} \underbrace{\lambda_j (d_k)_-}_{\leq 0} \operatorname{tr} \underbrace{\left((v_j^\top e_k)^\top v_j^\top e_k \right)}_{\geq 0}, \\
&\leq 0.
\end{aligned}
$$

On a donc montré que $\mathbf{M}_+ = \mathbf{U}^\top \mathbf{D}_+ \mathbf{U} = \Pi_{\mathbb{S}_+^d}(\mathbf{M})$.