Guide de travail pour le projet
Soutenances: 22 avril 2020
Introduction à l'apprentissage automatique
Polytech Paris-Sud

1 Choix du problème

Parcourez d'abord rapidement quelques listes de datasets (cf. les liens fournis dans les slides), pour vous faire une idée de l'étendue des possibilités. Attention, avec ce qu'on a vu en cours, il vaut mieux éviter les datasets d'images complexes, de flux audio ou vidéo, qui ne peuvent bien souvent être traités de façon satisfaisante qu'avec du *Deep Learning*. De même, nous n'avons pas fait de *reinforcement learning*, donc ne vous lancez pas dans la création d'une IA de jeux vidéo.

Le genre de tâche que vous devriez essayer de réaliser sont plutôt de l'apprentissage supervisé ou éventuellement non-supervisé (et évidemment semi-supervisé ou auto-supervisé (*self-supervised*) si vous le souhaitez). Vous pouvez vous lancer dans du traitement de langage naturel (texte), si vous trouvez un *embedding* déjà appris adapté à votre dataset.

2 Pipeline: guide

Cette section est un guide, pour vous aider à vous poser les bonnes questions lorsque vous résolvez un problème de ML (lorsque vous travaillez à la construction d'un pipeline de ML pour résoudre automatiquement une *tâche* donnée).

2.1 Exploration des données

La première chose à faire consiste à se donner un aperçu des données. En général les datasets fournis en ligne viennent avec une petite présentation et parfois quelques visualisations, mais il est recommandé de faire vous même quelques fonctions permettant d'avoir un aperçu des données/une donnée, afin d'avoir une idée des problèmes qui vont se poser (et comme outil de débug). Vous pouvez utiliser des outils autres (comme *t-SNE*), sans forcément les maitriser à fond, dans cette phase exploratoire.

2.2 Définition de la tâche, et choix d'une métrique de performance

Selon la complexité du dataset (estimée lors de la visualisation), viser un objectif non ridicule mais pas trop ambitieux non plus (vous avez relativement peu de temps).

Selon le type de tâche ambitionnée (régression ou classification, etc), déterminer une ou des possibilités de "métriques" qui constitueront votre score. Il faut faire attention aux biais dans les données: par exemple si les différentes classes ne sont pas du tout équilibrées, il faut définir une métrique qui donne un poids conséquent y compris aux classes peu fréquentes. Faites un premier choix rapidement, si il s'avère mauvais, vous corrigerez plus tard.

Allez voir les slides "cours4-metriques.pdf" pour rappel.

2.3 Pre-processing (calibration)

Si les données ne sont pas propres, il se peut que vous ayez à les nettoyer. Par exemple, il se peut que certaines valeurs soient manquantes (il faudra alors les remplacer par quelque chose, à choisir, ou modifier l'algorithme d'apprentissage en conséquence). Une autre possibilité est que leur format soit problématique, par exemple des images qui ne sont pas toutes de la même résolution, ou encore pas directement lisibles en python.

Cette étape peut être réalisée une fois pour toute, en général.

C'est aussi le moment de définir le *train set*, *validation set*, et *test set*. Assez souvent, il suffit de mélanger les données (par exemple si MNIST venait avec les chiffres triés dans l'ordre, il faudrait mélanger pour que le *test set* ne soit pas rempli que de 8 et de 9!). Mais attention, parfois on ne peut pas tout mélanger d'un coup,

par exemple si les données sont issues d'une série temporelle et sont corrélées, il faut autant que possible laisser ensemble les points proches dans le temps, et mélanger les *train/validation/test sets* seulement en interne et pas entre eux.

2.4 Optimisations

Cette étape peut être itérée plusieurs fois, selon la satisfaction que vous apportent vos résultats.

2.4.1 Architecture

Il va falloir choisir une architecture, c.a.d. le genre de modèle de ML utilisé pour modéliser les données ("apprendre" quelque chose).

On rappelle ici les méthodes vues en cours:

- Perceptron (à une couche)
- Régression (linéaire ou polynomiale)
- Noyaux (Kernels): polynomial, RBF
- Perceptron multi-classe (avec un softmax): vu, mais rapidement
- K-moyennes
- Modèle Bayésien Naïf (Bernoulli ou Gauss ou autre)
- Algorithme EM (pour le mélange de lois de Gauss par exemple, appellée GMM)
- Estimation de densité: histogrammes, méthode de Parzen, GMM
- K-Nearest Neighbors (si vous l'utilisez, expliquez en le fonctionnement)
- Perceptron Multi Couches (Multi layer Perceptron) (si vous l'utilisez, expliquez en le fonctionnement)
- SVC (SVM, pour la Classification) (si vous l'utilisez, expliquez en le fonctionnement)
- SVR (SVM, pour la Regression) (si vous l'utilisez, expliquez en le fonctionnement)

Vous pouvez choisir librement un modèle de Machine Learning en dehors de ceux qui ont été discutés en cours (ci dessus), tant que vous êtes capbles d'expliquer clairement au moins l'idée générale de la méthode (voir section 2.6).

2.4.2 Choix d'hyper-paramètres

Pour chaque méthode, il y a en général un certain nombre d'hyper-paramètre associés, qu'il vous est nécessaire de choisir. Il y a parfois aussi des paramètres qui servent seulement à imposer la méthode de résolution numérique du modèle. On ne vous demande pas d'optimiser ces paramètres là.

Remarque/rappel: le choix de l'architecture est en soi un hyper-paramètre. La taille du *training set* en est aussi un.

Lorsque vous chercherez à optimiser des hyper-paramètres, n'oubliez pas :

- la cross validation
- ne vous lancez pas tête baissée dans une recherche systématique de plus de 2 paramètres à la fois
- si les calculs sont lents, réduisez la taille du *train set*, mais seulement si ça coute peu en termes de performance.
- si les calculs sont lents, envisagez de comprimer les features d'entrée (PCA), même si ça coute un peu en termes de performance.

Allez voir les slides "cours3-overfitting+validation.pdf" pour rappel.

2.4.3 Pre-processing (plus avancé, style *feature selection*)

On rappelle les quelques pre-processings vus en cours:

- Standardisation
- PCA
- suppression de features inutilies/nuisibles à la main
- Équilibrage des classes (sous-échantillonage), dans le cas de classes mal équilibrées, et lorsque l'algorithme ne permet pas de corriger le biais automatiquement.

Vous pouvez utiliser d'autres pre-processings (ICA par exemple), à condition d'être capables d'en expliquer l'idée générale de fonctionnement (voir section 2.6).

2.4.4 Test

Pensez à regarder les différentes mesures de scores envisagées au départ sur l'ensemble de validation (une fois vos hyper-paramètres fixés). Vérifiez qu'il n'y a pas un gros problème que vous n'auriez pas vu. Si les résultats sont raisonablement satisfaisants, allez y avec le test set ! (Bref, ne trichez pas lors du test !)

2.5 Conclusion

Lorsque vous présentez vos résultats finaux, vous pouvez faire apparaître seulement les performances du *test set*, éventuellement accompagnés de ceux sur le *valdiation set*, et éventuellement encore accompagnés de ceux pour le *train set*. En effet, certaines visualisations ne permettent pas de présenter simultanément plusieurs résultats (par exemple si ce sont des images reconstruites, etc). Dans ce cas, privilégiez toujours le *test set*.

Ne vous focalisez pas seulement sur la performance quantitative (la valeur du score), essayez d'analyser la façon dont les résultats dépendent de vos choix (même si le niveau total n'est "pas très bon").

2.6 Cadre du projet noté

Dans le cadre du projet, si vous souhaitez travailler plutôt sur l'aspect algorithmique que sur l'aspect data/tâche, vous pouvez utiliser un dataset très classique (Fashion-MNIST ou même MNIST, ou des data sets synthétiques, du genre des *blobs* gaussiens) plutôt que de passer du temps à explorer les datasets et les données elles-même ensuite. Il sera alors attendu que vous codiez vous même l'algorithme, après en avoir bien compris le fonctionnement. Des algorithmes envisageables sont le *Multi-Layer Perceptron*, les *Decision Trees*. Dans ce cas là, la recherche d'optimisation des hyper-paramètres pourra rester très sommaire (il faudra plutôt prouver que votre implémentation tourne correctement).

Si vous choisissez d'utiliser un algo hors du cours, mais sans le re-coder, on attendra de vous que vous ayez bien compris l'algorithme, qu'il soit expliqué avec **vos propres mots** dans le rapport et à l'oral, et il faudra alors tout de même choisir un dataset légèrement "non trivial". Vous pourrez en conséquence alléger la partie "optimisation des hyper-paramètres".

3 Indications pour l'oral

Ces conseils sont en fait assez génériques (valides en dehors de ce cours)

3.1 Objectifs de l'oral

- Faire découvrir un problème intéressant/amusant/intriguant à vos camarades (et profs).
- Montrer que vous avez sû mettre en oeuvre des méthodes de Machine Learning, les avez comprises, etc (comme pour le rapport écrit)
- Montrer que vous savez transmettre des connaissances récemment acquises à un public non expert
- Faire comprendre à vos camarades (certaines) des méthodes que vous avez utilisées (en particulier des points nouveaux/peu ou pas mentionnés en cours)
- Partager votre expérience sur les problèmes spécifiques à votre tâche/dataset/algorithme que vous avez rencontrés (et résolu, ou pas). Vous êtes encouragés à faire preuve d'esprit critique !
- Accessoirement, si tout marche bien, exhiber un résultat inattendu (une corrélation par exemple) sur des données réelles (si le sujet s'y prête).
- Être **intéressant** ! Lorsque vous préparez, demandez vous: si vous étiez dans le public, seriez vous intéressés par cette présentation ? Auriez vous envie d'écouter?

3.2 Structure typique d'une présentation

(pour l'oral de ML)

1. Intro générale au problème (**contexte** de votre dataset, **d'où vient-il, pourquoi est-il là**, etc).
2. Aperçu du dataset: 1 ou 2 visualisations.
3. Définition du problème: **quel type de tâche est-ce**, quel est votre **critère de qualité** (métrique de score) ?

optionnel Présenter votre travail de **nettoyage des données** (si c'est intéressant/non trivial)

4. Votre choix d'algos (pre-proc, architecture) finalement choisis. Présentation de votre **pipeline**. **Justifications à l'aide de figures**.

optionnel *Si* vous avez choisi de ré-implémenter un algo depuis *numpy*, ou bien choisi d'utiliser un algo pas étudié en cours, *alors* en expliquer le fonctionnement sera le coeur de votre présentation.

optionnel Présenter l'**optimisation d'un hyper-paramètre** (au moins un, sauf si ça n'a pas du tout été votre problème). (Ça peut aussi être la comparaison entre deux modèles, deux architectures différentes).

5. **Présentation des résultats** (performance/représentation construite) sur l'ensemble de test.
6. **Conclusion**: vous êtes encouragés à faire preuve d'**esprit critique** ! (par exemple, il se peut que le dataset soit simplement trop petit pour conclure quoi que ce soit avec les méthodes disponibles, et on peut parfois le montrer: vous avez le droit de critiquer les données!)

3.3 Astuces pour faire une bonne présentation (assez génériques)

- Vous ne pourrez pas parler de tout ce que vous avez fait en 10 minutes. Il faudra choisir. Il vaut mieux parler de peu de choses mais bien, clairement, posément, que de plein de choses, mal.
- Visez d'avoir seulement un (ou deux) "message à retenir" dans votre exposé. L'objectif de votre exposé sera alors de transmettre ce message à l'auditoire, et donc d'introduire juste ce qui est nécessaire pour que le message soit compréhensible.
- Une fois le message principal établi, lorsque vous vous demandez "est ce que je parle de ceci, de cela", posez vous la question : est ce indispensable pour transmettre mon message ? Si non, alors laissez tombez cette mini-explication, c'est une perte de temps.
- Un travers typique d'étudiant débutant en exposé est de vouloir 'montrer qu'on est fort'/'montrer que c'est compliqué'. Évitez de vouloir épater la galerie avec plein d'équations compliquées et pas bien expliquées, ou des gros schémas pas bien expliqués. Ce qui est impressionnant, c'est de voir une présentation très claire. Ce que vous faites est déjà assez complexe, ce n'est pas la peine d'en rajouter.
- Il n'est pas honteux de nous ré-expliquer des notions vues en cours, avec vos propres mots (c'est en partie le but de l'exercice de l'oral).
- Essayer d'avoir un seul sujet par slide (et pas 5)
- Viser environ 1 min par slide. Répéter au moins 2 fois pour bien minuter.
- Slides pas trop remplis: pas plus de 3-4 lignes de texte par slide, typiquement.
- Les figures en disent souvent plus que le texte (y compris les schémas).
- Les figures (graphiques) doivent être lisibles (axe des x et y lisible, etc). N'hésitez pas à les mettre en plein écran, qu'on voie bien (si on voit mal c'est de votre faute, pas de celle du public).
- Mettez vous à la place du public, pour vérifier que vous avez bien exposé le contexte (l'idéal est de présenter à un autre groupe)
- Répétez votre présentation ! Équilibrez la quantité de texte entre les membres du groupe.
- Mieux vaut faire un peu moins de 10 min (8 par exemple) que plus.
- Numéroter les slides (ça aide pour poser les questions), c'est une bonne habitude.
- Vous pouvez prévoir de faire une démo à un moment. Si c'est le cas, prévoyez le temps adéquat.

4 Indications pour le rendu écrit (rapport+code)

Le rapport est un complément à la présentation orale, qui constitue le coeur de votre rendu. Le rapport sera donc consulté, et servira a amender la note d'oral, mais ne sera pas forcément noté explicitement. Une excellente présentation ne sera pas sanctionnée si le rapport+code est un peu faible (sauf si il y a pipeautage). En revanche, une présentation faible pourra être remontée par un rapport+code de bonne qualité.

Cependant, il est recommandé de travailler le fond du rapport, dans la mesure où c'est un guide pour produire une bonne présentation.

4.1 Objectifs du rapport

- Montrer que vous avez sû mettre en oeuvre des méthodes de Machine Learning ; vous les avez comprises ; savez analyser les résultats
- Convaincre qu'en cas de situation similaire dans le futur, vous seriez à même de mener ce genre d'investigation scientifique (exemple: convaincre votre patron que telle méthode est la plus performante, et que sa performance est sous contrôle.)
- Compléter l'oral, en mettant par écrit les choses un peu fastidieuses à présenter, mais qui sont tout de même importantes. Si un résultat nécessite une analyse un peu fine pour être démontré rigoureusement, c'est ici qu'il faut le faire (et à l'oral, donner l'idée générale).
- Les résultats "négatifs" (faible performance d'une méthode) sont également intéressants, vous pouvez les présenter et les commenter !

4.2 Contenu (conseils)

Vous pouvez suivre la trame fournie plus haut. Vous êtes également fortement encouragés à mentionner également toutes sortes d'observations qui vous sembleraient pertinentes !
Évitez d'écrire des tartines ! "Analyse" ne signifie pas "bavarder".
La concision est votre amie. **Longueur: 6 pages max.**
Préférer la comparaison de peu de modèles/hyper-paramètres, bien faite, plutôt que la comparaison de tous les cas possibles, mal faite.
Il n'y a pas un seul modèle de bon rapport attendu. Étudiez ce qui vous semble intéressant, et commentez vos observations.
Une observation non expliquée n'a que peu d'intérêt. Justifiez vos affirmations, mais ne vous embourbez pas dans des discussions vagues (quand on ne sait pas, il est parfois bon de se taire).

4.3 Les figures (!)

Penser la figure comme un ensemble cohérent, qui doit véhiculer un message : avant de mettre le point final sur une figure, demandez vous "le fait scientifique que je souhaite exhiber/démontrer par cette figure en ressort-il clairement ?"
Penser à tracer plusieurs courbes (mais pas trop nombreuses) sur un même graphique, pour éviter la multiplication des figures.
Adopter un code couleur cohérent (éventuellement aussi des types de ligne avec l'option “linestyle” de plot())
Utiliser des légendes (plt.plot(...., label="ma legende pour la courbe 1", et plt.legend(loc="best"))
Astuce: pour des figures de bonne qualité (en termes de pixels), les enregistrer en .pdf ou .svg plutôt qu'en .png
En général, on ne se souvient que des figures... prenez en soin !

4.4 Qualité rédactionnelle du rapport

On rappelle quelques évidences. Seront prises en comptes:

Présentation, lisibilité :
Apparence générale
Qualité des images et des schémas
Légendes
Harmonie, esthétique

Rédaction :
Respect de l'orthographe
Proportion texte/nombre de figures
Clarté de l'expression et cohérence de la syntaxe

Construction des paragraphes et rédaction de transitions

Mise en page :
Numérotation des pages
Mise en page soulignant clairement les différentes parties
Présence de titres, sous titres, paragraphes...
Soin de la syntaxe et respect de l'orthographe

4.5 Code

Si vous êtes particulièrement fièr(e)s d'un morceau de code que vous avez écrit, vous pouvez le commenter (succinctement) dans le rapport.
Un bon code est un code commenté
Prendre soin du choix des noms de variables/fonctions (astucieux, cohérent)
Pas plus de 3 fichiers de code (et de préférence 1 seul, pour notre confort de lecture-correction)

5 Concrètement

Attendu dans un .zip (ou autre .tar.gz) : (NOM1_NOM2.tar.gz)

1. Un dossier ayant pour nom vos noms de famille: NOM1_NOM2 , et contenant lui-même:
2. rapport.pdf (rappelez vos noms en première page)
3. (votre code python) (de sorte que je puisse exécuter votre code facilement)
4. les données, si elles sont très petites (archive de moins de 10 Mo), ou un lien vers les données, très clair/facile à utiliser (à mettre dans le rapport)

Contents