

MiniProject 2

- Details are now available:

ULaval	IFT&GLO
Mini-Projet #2 (sur 20%) à rendre le 22 juillet 2020 à 23h55mn	
Professeur Benjamin Chab-draa	GLO-7050 : Apprentissage machine en pratique

Les projets du cours sont les projets des versions du cours COMP 551, cours donné par les collègues de McGill, un grand merci à eux pour nous avoir donné l'autorisation de les utiliser.

Preamble

- Ce mini-projet constitue un travail individuel. Toutefois, ceci nous vous empêche pas de discuter avec les autres étudiants qui suivent le cours. En aucun cas cependant vous ne devez reprendre le code et l'écrit d'autrui; il vous est demandé d'élaborer les vôtres.
- Vous allez soumettre votre travail sur MonPortail, et également à une compétition Kaggle. Vous devez vous inscrire pour la compétition Kaggle en utilisant le courriel auquel vous êtes associé sur MonPortail (c.-à-d. ulaval.ca). Vous pouvez vous inscrire à la compétition à l'adresse suivante : <https://www.kaggle.com/t/aba68fed40124aa9a379a59f55a11213>. Pour le MiniProject 2, vous devez enregistrer votre Equipe individuelle (un étudiant par équipe) sur MonPortail. Vous devez par la suite former une équipe (d'un étudiant) sur Kaggle portant le même nom d'équipe que MonPortail (vous devez utiliser le nom de votre équipe MonPortail comme nom d'équipe sur Kaggle). Toutes les soumissions Kaggle doivent être associées à une équipe valide enregistrée sur MonPortail.
- Si vous "empruntez" des idées, méthodes, démarches ou autres, merci d'indiquer vos sources dans le rapport.
- Il vous est fortement suggéré d'argumenter et/ou justifier vos réponses.
- Après la date de remise, vous avez jusqu'à une semaine pour remettre votre travail avec une pénalité de 20%. Au-delà le travail vaut 0.
- Vous êtes libres d'utiliser les bibliothèques telles que Numpy pour Python. Toutefois à moins d'être explicitement autorisées, vous ne devez pas utiliser des implementations pré-existantes des algorithmes demandés, vous devez les implémenter par vous-même.
- si vous avez des questions concernant le travail, merci de passer par le Forum, en posant clairement vos questions.

Background

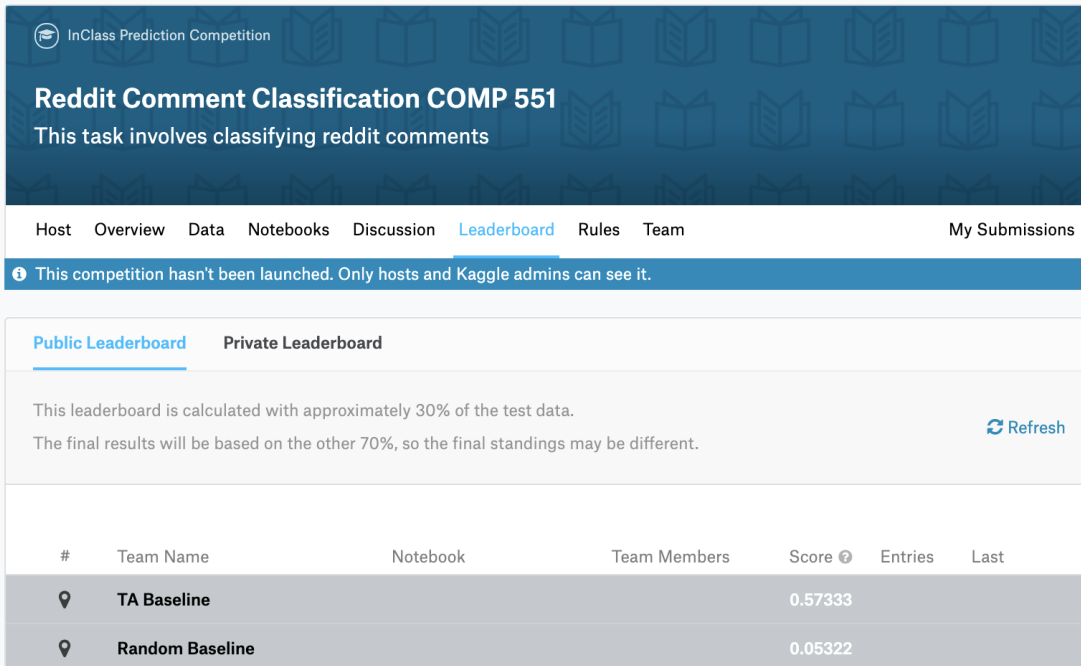
In this mini-project you will develop models to analyze text from the website Reddit (<https://www.reddit.com/>), a popular social media forum where users post and comment on content in different themed communities, or subreddits. The goal of this project is to develop a supervised classification model that can predict what community a comment came from. You will be competing with other groups to achieve the best accuracy in a competition for this prediction task. However, your

1

- Due 22 juillet à 23h55 .

MiniProject 2

Kaggle leaderboard



InClass Prediction Competition

Reddit Comment Classification COMP 551

This task involves classifying reddit comments

Host Overview Data Notebooks Discussion **Leaderboard** Rules Team My Submissions

This competition hasn't been launched. Only hosts and Kaggle admins can see it.

Public Leaderboard Private Leaderboard

This leaderboard is calculated with approximately 30% of the test data.
The final results will be based on the other 70%, so the final standings may be different. [Refresh](#)

#	Team Name	Notebook	Team Members	Score ?	Entries	Last
📍	TA Baseline			0.57333		
📍	Random Baseline			0.05322		

$$\text{points} = 50 * \begin{cases} 0 & \text{if } X < R \\ \frac{X-R}{T-R} * 0.75 & \text{if } X > R \text{ and } X \leq T \\ \frac{X-T}{B-T} & \text{if } X > T \text{ and } X \leq B \\ 1 & \text{if } X > B \end{cases}$$

Your accuracy

Random Baseline

TA Baseline

Accuracy of 3rd best group

Steps to solving a supervised learning problem

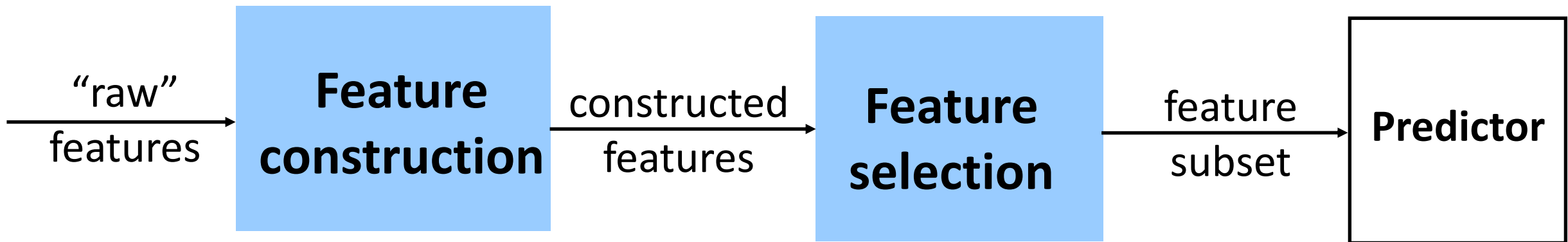
1. Decide what the input-output pairs are.
2. Decide how to encode inputs and outputs.
 - This defines the input space X and output space Y .
3. Choose a class of hypotheses / representations H .
 - E.g. linear functions.
4. Choose an error function (cost function) to define best hypothesis.
 - E.g. Least-mean squares.
5. Choose an algorithm for searching through space of hypotheses.

With a focus on feature extraction
for language problems!

Today:
deciding on
what the
inputs are

So far:
we have been
focusing on
this

Feature extraction steps



A few strategies

- Use domain knowledge to construct “ad hoc” features.
- Normalization across different features, e.g. centering and scaling with $x_i = (x'_i - \mu_i) / \sigma_i$.
- Normalization across different data instances, e.g. counts/histogram of pixel colors.
- Non-linear expansions when first order interactions are not enough for good results, e.g. products x_1x_2 , x_1x_3 , etc.
- Other functions of features (e.g. sin, cos, log, exponential etc.)
- Regularization (lasso, ridge).

Feature construction

Why do we do feature construction?

- Increase predictor performance.
- Reduce time / memory requirements.
- Improve interpretability.

But: Don't lose important information!

Problem: we may end up with lots of possibly irrelevant, noisy, redundant features. (Here, “noisy” is in the sense that it can lead the predictor astray.)

Scikit-learn et exemple

- <http://scikit-learn.org/stable/modules/classes.html>
- [implementing-a-naive-bayes-classifier-for-text-categorization](#)
- Help/Questions : Amar Ali-Bey amar.ali-bey.1@ulaval.ca

Applications with lots of features

- Any kind of task involving **images** or **videos** - object recognition, face recognition. Lots of pixels!
- Classifying from gene expression data. Lots of different genes!
 - Number of data examples: 100
 - Number of variables: 6000 to 60,000
- Natural language processing tasks. Lots of possible **words**!

Features for modelling natural language

- Words
- TF-IDF
- N-grams
- Syntactic features
- Word embeddings (not in this lecture...)
- Useful Python package for implementing these:
<http://www.nltk.org/>

Words

- Binary (present or absent)
- Absolute frequency
 - i.e., raw count
- Relative frequency
 - i.e., proportion
 - document length

Document 1

The quick brown
fox jumped over
the lazy dog's
back.

Document 2

Now is the time
for all good men
to come to the
aid of their party.

Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Stopword List

for
is
of
the
to

Multinomial vs Bernoulli Naïve Bayes

- Note that using binary vs. count features can have implications for your model!
- In Lecture 5 we discussed the **Bernoulli (i.e., binary) Naïve Bayes**:

$$P(\mathbf{x}|y = k) = \prod_{j=1}^m \theta_{j,k}^{x_j} (1 - \theta_{j,k})^{(1-x_j)} \quad \leftarrow \text{Bernoulli distribution}$$

- Assumes the features are binary counts.
- But we can also have a **Multinomial Naïve Bayes**:

$$P(\mathbf{x}|y = k) = \frac{(\sum_{j=1}^m x_j)!}{\prod_{j=1}^m x_j!} \prod_{j=1}^m \theta_{j,k}^{x_j} \quad \leftarrow \text{Multinomial distribution}$$

- Assumes the features are **integer** counts.

Homework: what is the maximum likelihood estimate for the multinomial Naïve Bayes?

More options for words

- Stopwords

- Common words like “the”, “of”, “about” are unlikely to be informative about the contents of a document.
- Standard practice to remove them, though they can be useful (e.g., for authorship identification)

- Lemmatization

- Inflectional morphology: changes to a word required by the grammar of a language
 - e.g., “perplexing” “perplexed” “perplexes”
 - (Much worse in languages other than English, Chinese, Vietnamese)
- **Lemmatize** to recover the canonical form; e.g., “perplex”

TF*IDF (Salton, 1988)

- **Term Frequency Times Inverse Document Frequency**
- A term is important/indicative of a document if it:
 1. Appears many times in the document
 2. Is a relative rare word overall
- TF is usually just the count of the word
- IDF is a little more complicated:
 - $IDF(t, Corpus) = \log \frac{\#(\text{Docs in Corpus})}{\#(\text{Docs with term } t) + 1}$
 - Can use a separate large training corpus for this
- Originally designed for document retrieval

TF-IDF (Wiki)

- <https://fr.wikipedia.org/wiki/TF-IDF>

N-grams

- Use sequences of words, instead of individual words
- e.g., ... *quick brown fox jumped* ...
 - Unigrams (i.e. words)
 - quick, brown, fox, jumped
 - Bigrams
 - quick_brown, brown_fox, fox_jumped
 - Trigrams
 - quick_brown_fox, brown_fox_jumped
- Usually stop at $N \leq 3$, unless you have lots and lots of data

Rich linguistic features

- **Syntactic**

- Extract features from a parse tree of a sentence
- [SUBJ The chicken] [VERB crossed] [OBJ the road].

- **Semantic**

- e.g., Extract the semantic roles in a sentence
- [AGENT The chicken] [VERB crossed] [LOC the road].
- e.g., Features are synonym clusters (“chicken” and “fowl” are the same feature) → WordNet
- **Trade-off:** Rich, descriptive features might be more discriminative, but are hard (expensive, noisy) to get!

Autres liens

- <https://medium.com/@RareLoot/basics-of-data-extraction-of-reddit-threads-using-python-c96854c41344>
- <https://praw.readthedocs.io/en/latest/tutorials/comments.html>
- <https://towardsdatascience.com/implementing-a-naive-bayes-classifier-for-text-categorization-in-five-steps-f9192cdd54c3>
- <https://www.learndatasci.com/tutorials/sentiment-analysis-reddit-headlines-pythons-nltk/>