Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Компьютерных сетей и систем

Обработка больших объемов информации

РЕФЕРАТ

на тему «Методы борьбы с переобучением моделей»

Студент
гр. 758601
Лимонтов А. С.

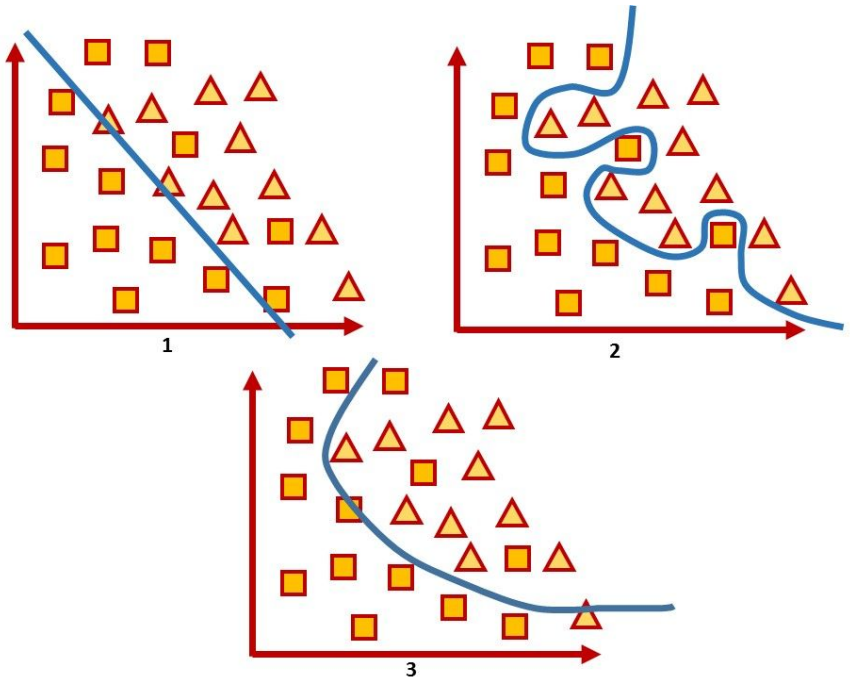
Проверил
Заливако С. С.

Введение

1. Терминология

Переобучение - ситуация, когда алгоритм хорошо подстроился под существующие данные, а на новых значительно ухудшается. Понятие переобучения хорошо показать на примере. Пусть при решении некоторой задачи был построен алгоритм, который ошибается на 10% обучающей выборки. Причем 10% является допустимой ошибкой (то есть изначальные требования предполагали более высокую ошибку). Означает ли это, что алгоритм переобучился? Чтобы ответить на этот вопрос, необходимо пропустить через полученный алгоритм новые данные, которые он еще не видел, и сравнить полученные результаты с настоящими. Если ошибка на них резко возрасла, то алгоритм действительно переобучился. Другими словами, он рассматривает выявленные закономерности на обучающей выборке, которые не характерны для общего набора данных текущей задачи. Говорят, что он показал хорошие результаты без извлечения истинной закономерности. Отсюда же вытекает и смежное понятие - *недообучение* - ситуация, когда алгоритм дает плохие результаты как на обучающей выборки, так и на новых данных.

Давайте рассмотрим следующий пример. На рисунке 3 приведен пример хорошего выделения закономерности размещения объектов на плоскости. Поэтому на рисунке 1 наблюдается недообучение (high bias - алгоритм показывает большую ошибку при небольшом изменении в обучающей выборке) - алгоритм как-то разделил объекты разных типов, но все же его качество низкое (количество ошибок большое). А на рисунке 2 наблюдается переобучение (high variance - алгоритм показывает большую ошибку на новых данных) - в данной задаче явно не подразумевалась такая сложная разделяющая кривая, поэтому, хоть и на обучающей выборке ошибка будет мала (действительно, каждый объект находится в своем сегменте), но для общего решения она не годится.



Таким образом, выявление переобучения без использования дополнительных данных (не участвовавших в процессе обучения), невозможен. Поэтому наличие таких “новых” данных необходимо. Отсюда и вытекает одно из логичных решений этой проблемы - отложить часть имеющихся данных в качестве “новых” данных и не использовать их при обучении. Такая часть называется *отложенной выборкой*, и она используется для проверки алгоритма.

Вообще есть несколько подходов к выявлению переобучения:

- Использование уже описанной отложенной выборки
- Кросс-валидация (более усложненный метод отложенной выборки)
- Использование меры сложности модели

Про них и пойдет речь далее.

2. Методы борьбы с переобучением

2.1. Регуляризация

Регуляризация помогает бороться против переобучения, используя меры сложности модели. В линейных методах мерой сложности модели являются веса при признаках. Например, когда веса сами по себе являются большими числами ($2342342x + 253423x^2 + 132x^3$), это говорит о том, что при небольшой вариации признака его вклад в итоговое значение будет очень большим.

Кроме того, возможна такая ситуация, как *мультиколлинеарность*. Так называется проблема, при которой признаки являются линейно зависимыми. Например, если в ходе обучения алгоритма был найден некий вектор весов *w*,

используемый алгоритмом для предсказания ответа, то линейный сдвиг относительно него по вектору v ($w^* := w + \alpha * v$) также будет решением задачи (будет так же хорошо описывать данные в выборке).

Вернемся к регуляризации. Она решает уже упомянутую проблему больших весов. Для этого помимо исходного функционала ошибки $Q(w, x)$ минимизируется также и дополнительный функционал, именуемый регуляризатором. Например, существует

- квадратичный регуляризатор: $L_2 = ||w||^2$. Он является выпуклым и гладким, что позволяет использовать градиентный спуск.
- абсолютный регуляризатор: $L_1 = ||w||_1$ (LASSO - Least Absolute Shrinkage and Selection Operator). Этот регуляризатор позволяет занулить некоторые признаки, точнее, придать вес равный 0. Такой свойство указывает на то, что с помощью L_1 регуляризатора можно отобрать лишь действительно важные признаки, которые влияют на результат, и отбросить признаки, которые скорее являются шумом.

Теперь на оптимизацию к минимуму подается выражение $Q(w, x) := Q(w, x) + \lambda * ||w||^2 \rightarrow min$ (например). Это можно трактовать как своеобразный штраф за некоторые веса. Например, L_2-регуляризация штрафует за слишком высокие веса.

Множитель lambda перед L_2 выше называется *коэффициентом регуляризации*. Его значение характеризует сложность модели. Соответственно, чем больше lambda, тем проще модель. Почему так? Потому что, например, если слишком завысить коэффициент регуляризации, то оптимальным решением будут веса, по значению близкие к 0. С другой стороны (противоположная крайность), если сделать его слишком маленьким, то это то же самое, что его почти нет, то есть алгоритм переобучится и модель получится очень сложной. Поэтому коэффициент регуляризации должен быть выбран оптимальным образом (как правило, он выбирается на этапе кросс-валидации).

2.2. Отложенная выборка

Простым способом выявления переобучения является отложенная выборка, как уже было показано ранее. Для этого исходная выборка разбивается на две части: одна используется для обучения (*обучающая*), другая для финального сравнения качества алгоритма, подсчета ошибки (*тестовая*). Возникает естественный вопрос: как нужно разбить исходную выборку, чтобы получить наиболее полезную информацию. Крайности здесь следующие: слишком маленькая тестовая выборка:

- позволяет по максимуму использовать вариации/зависимости из обучающей выборки

- делает ненадежной оценку качества алгоритма

С другой стороны, слишком большая тестовая выборка:

- с высокой вероятностью произойдет недообучение, поскольку обучающая выборка мала
- оценка качества становится надежной.

Эксперты рекомендуют разбивать в отношении обучающая/тестовая = 70/30 (или чтобы тестовая занимала $\frac{1}{e}$ часть от общей выборки).

Из определения отложенной выборки вытекает, что разбиение и обучение стоит производить один раз. Но в этом случае появляется зависимость от разбиения, например, если некоторые признаки обладают высокой информативностью для небольшой подвыборки, а она попала в тестовую часть выборки, то алгоритм не распознает их вклад при обучении и получит достаточно большую ошибку. Чтобы снизить риск возникновения такой ситуации, применяется *композиция* (*ensembling*). Для этого делается n разбиений *признаков* (не самой выборки), причем можно проводить эти операции как возвращением, так и без, на примерно одинаковые по мощности множества. Затем для каждого подмножества признаков над всей выборкой происходит обучение алгоритма, а в качестве итогового ответа берется среднее взвешенное значение из всех предсказанных ответов.

2.3. Кросс-валидация

Кросс-валидация (*k-fold*) зависит от единственного параметра *k*. Ее алгоритм следующий:

- Выборка разделяется на k блоков одинакового размера
- Проводится k обучений: в каждом из них блок *i* используется в качестве тестовой выборки, все остальное - в качестве обучающей. Ответ берется как среднее из значений на каждом из k прогонов.

Здесь возникает вопрос, как и в отложенной выборке, как необходимо разбивать выборку, или какое k будет приемлемым. Опять же, крайности таковы, что: при малом k тестовые выборки будут большими, а значит надежность будет высока, но обучение может пропустить важные зависимости в признаках (a.k.a. high bias - low variance), при большом же k надежность уменьшается, но алгоритм учитывает больше зависимостей (a.k.a. low bias - high variance). Эксперты рекомендуют выбирать k ~ 5.

Стоит помнить, что в кросс-валидации, как и при отложенной выборке, в большинстве случаев стоит перемешивать исходные данные.

3. Сравнение алгоритмов

При выборе оптимального алгоритма очень часто приходится подбирать гиперпараметры. Одним из возможных методов является *перебор по сетке*, когда заранее задаются несколько значений каждого перебираемого гиперпараметра, затем на каждой комбинации проверяется эффективность обученной модели. В *sklearn* есть специальный класс *sklearn.model_selection.GridSearchCV*, который позволяет легко подбирать необходимые значения.

Кроме подбора гиперпараметров, стоит пробовать воздействие различных регуляризаторов, выбирать алгоритмы различных категорий, изменять параметры. При сравнении алгоритмов можно пользоваться как отложенной выборкой, так и кросс-валидацией. Но стоит помнить, что если проверка проводится на одной и той же тестовой выборке, то фактически при сравнении алгоритмов легко “переобучиться” на ней и выбрать неправильную модель, поскольку в этом случае тестовая выборка будет использовать как часть обучения. В связи с этим рекомендуется разбивать исходную выборку на 3 части: *обучающую*, *валидационную* и *контрольную* в случае отложенной выборки и *обучающую* и *контрольную* в случае кросс-валидации. Обучающая выборка используется для обучения каждого алгоритма, на валидационной каждый алгоритм проверяет себя (считает variance), из этих алгоритмов выбираются лучшие, и затем в качестве “проверки на адекватность” они запускаются на контрольной выборке.

Литература

1. For k-fold cross-validation, what k should be selected? https://www.quora.com/For-K-fold-cross-validation-what-k-should-be-selected
2. Machine Learning. Курс от Яндекса для тех, кто хочет провести новогодние каникулы с пользой https://habr.com/company/yandex/blog/208034/
3. Overfitting in Machine Learning: What It Is and How to Prevent It

   https://elitedatascience.com/overfitting-in-machine-learning
4. The problem of overfitting https://www.coursera.org/lecture/machine-learning/the-problem-of-overfitting-ACpTQ
5. Underfitting and overfitting in machine learning https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/