Алгоритмы машинного обучения с учителем

Здравствуйте, уважаемые слушатели! Тема нашей лекции – Алгоритмы машинного обучения с учителем.

План лекции:

1. Регрессия
    a. Линейная регрессия
    b. Полиномиальная регрессия
2. Классификация
    a. Логистическая регрессия
    b. k ближайших соседей
    c. Алгоритм опорных векторов
    d. Ансамблевые методы

1. Вступительное слово

Основная тема сегодняшнего занятия является обучение с учителем, которая уже была затронута в прошлых занятиях. Обучение с учителем - это выявление скрытой закономерности по конечному числу примеров. Но следует отметить при схожих постановках задач, они отличаются друг от друга решениями и выдвигающимися требованиями к решению.

2. Регрессия

Регрессия - статистический метод, позволяющая строить математические зависимости значений данных, в результате которого можно прогнозировать будущие значения. Главным отличием от классификации заключается в том, что она предсказывает количества, тогда как классификация предсказывает классы. Данное отличие можно отнести к недостаткам регрессии, так как она работает с количественными данными.

Регрессия различаются двумя типами: парная и множественная. Отличие двух типов заключается в количестве независимых переменных, влияющих на Y, а также видами уравнения: линейные и нелинейные.

Уравнение парной регрессии можно выразить следующим образом:

$$Y= F(x)$$

Уравнение множественной регрессии:

$$Y=F(x_1, x_2, \dots ,x_n)$$

Линейная регрессия

Линейная регрессия относится как парной так и множественной регрессии. Давайте рассмотрим линейную регрессию с множественной переменной.

$$Y=b_0 x_0 + b_1 x_1 + b_2 x_2 + \dots + b_n x_n$$

где x_0 = 1, то уравнение примет следующий вид:

$$Y=b_0 + b_1 x_1 + b_2 x_2 + \dots + b_n x_n$$

Как мы видим из данного уравнения у нас не имеются коэффициенты. Для вычисления используем следующее матричное выражение:

$$B=(X| \quad |TX)^{-1} X^T Y$$

где B – вектор коэффициентов; $(X^T X)^{-1}$ – обратная матрица $X^T X$; X^T – транспонированная матрица X.

Рассмотрим пример. Мы будем использовать готовую реализацию линейной регрессии из библиотеки scikit-learn. Датасет мы также взяли из данной библиотеки.

Подробнее о метриках точности классификации и регрессии будет рассказано в следующем уроке.

Полиномиальная регрессия

Полиномиальная регрессия также может применяться для двух типов регрессии, то есть, для парной и множественной. Но в отличие от линейной регрессии, полиномиальная регрессия позволяет строить гиперплоскости сложной формы. Однако с увеличением числа параметров существенно возрастает вычислительная сложность алгоритма. Кроме этого, существует опасность «переобучения», когда форма кривой или гиперповерхности становится слишком сложной, практически полностью подстроившись под обучающее множество, но дает большую ошибку на тестовом множестве. Это свидетельствует о том, что алгоритм потерял способность к обобщению и предсказанию. Множественная полиномиальная регрессия выражается следующим образом:

$$Y = b_0 + b_1 x_1^1 + b_2 x_2^2 + \ldots + b_n x_n^n$$

Как вы видите формула очень схожая с линейной регрессией, разница только в степени. Из за этого данную регрессию также называют степенной регрессией.

3. Классификация

Метод позволяющая сгруппировать объекты из заранее известных наборов классов. Классы должны быть определены изначально до обработки этих данных. Одним из популярным задач классификации является обнаружение спама.

Логистическая регрессия

Хотя в названии присутствует слово регрессия, на самом деле данный алгоритм не имеет никакого отношение к данному методу. Алгоритм решает задачи бинарной классификации, так как алгоритм применяет сигмоидальную функцию, который расположен между $y \in \{0,1\}$. Отсюда можно сделать вывод, что цель данного алгоритма не восстановление значений или предсказание, а классификация.

В данном методе выполняется условие, где $0 \leq Y \leq 1$, что достигается применением сигмоидальной (логистической) функции:

$$Y = \frac{1}{1+e^{-F(x)}}$$

где $F(x)$ – стандартное уравнение регрессии. При этом нужно учесть что если значение не равно 0 или 1, значение аппроксимируется.

Алгоритм k ближайших соседей (k-Nearest Neighbor – k-NN)

Жадный алгоритм требующий большого количества подсчета объектов обучающей выборки каждого класса в пространстве. После подсчета отбирается пространство из k объектов с минимальным расстоянием в центре которого располагается распознаваемый объект. Классифицируемый объект относят к тому классу, объектов у которого больше всего в данном пространстве. Но прежде чем начать подсчет, нужно выбрать какой алгоритм будет применяться для этого подсчета. Расстояние между классифицируемыми объектами может рассчитываться как расстояние в декартовом пространстве (евклидова метрика), но можно

использовать и другие метрики: манхэттенскую (Manhattan), метрику Чебышева (Chebyshev), Минковского (Minkowski) и др.

В качестве классического алгоритма вычисления расстояния выбирается евклидова метрика, которая выражается следующим образом:

$$d(a, b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

где aиb это точки (объекты), состоящий из координат (признаков).

В целом это один из самых простых, но часто неточных алгоритмов классификации. Алгоритм также отличается высокой вычислительной сложностью. Объем вычислений при использовании евклидовой метрики пропорционален квадрату от числа обучающих примеров.

Важно отметить что алгоритм применяется не только для задач классификации но и для регрессии.

Рассмотрим пример. Используем уже упомянутый ранее набор данных Iris. С помощью данного алгоритма можно решать не только задачи бинарной классификации, но и множественную классификацию.

Процесс обучения классификатора:

```
KNN = neighbors.KNeighborsClassifier(n_neighbors=5, weights='distance')
KNN.fit(x_train, y_train)
y_pred = KNN.predict(x_test)
```

Алгоритм опорных векторов

Алгоритм опорных векторов (Support Vector Machines) [1] относится к группе граничных методов: он определяет классы при помощи границ областей. В основе метода лежит понятие плоскостей решений. Плоскость решения разделяет объекты с разной классовой принадлежностью. В пространствах высоких размерностей вместо прямых необходимо рассматривать гиперплоскости – пространства, размерность которых на единицу меньше, чем размерность исходного пространства. В $\mathbb{R}^3$, например, гиперплоскость – это двумерная плоскость.

Метод опорных векторов отыскивает образцы, находящиеся на границах классов (не меньше двух), т.е. опорные векторы, и решает задачу нахождения разделения множества объектов на классы с помощью линейной решающей функции. Метод опорных векторов строит классифицирующую функцию f(x) в виде:

$$f(x) = sign(\langle w, s \rangle + b),$$

где $\langle w, s \rangle$ – скалярное произведение; *w* – нормальный (перпендикулярный) вектор к разделяющей гиперплоскости; *b* – вспомогательный параметр, который равен по модулю расстоянию от гиперплоскости до начала координат. Если параметр b равен нулю, гиперплоскость проходит через начало координат.

Объекты, для которых f(x) = 1, попадают в один класс, а объекты с f(x) = -1 – в другой.

С точки зрения точности классификации лучше всего выбрать такую прямую, расстояние от которой до каждого класса максимально. Такая прямая (в общем случае – гиперплоскость) называется оптимальной разделяющей гиперплоскостью. Задача состоит в выборе w и b, максимизирующих это расстояние.

1 Support vector machine. – http://en.wikipedia.org/wiki/Support_vector_machine (2012-02-22).

В случае нелинейного разделения существует способ адаптации машины опорных векторов. Нужно вложить пространство признаков R^n в пространство H большей размерности с помощью отображения: $\varphi = R^n \rightarrow H$. Тогда решение задачи сводится к линейно разделимому случаю, т.е. разделяющую классифицирующую функцию вновь ищут в виде: $f(x) = sign(\langle w, \phi(x) \rangle + b)$.

Для разделения он применяет ядра. Наиболее распространенные ядра: линейное, радиальная базисная функция, сигмоидальное, полиномиальное и т.д.

Существенным недостатком классификатора является значительное возрастание времени обучения при увеличении количества примеров. Другими словами, алгоритм обладает высокой вычислительной сложностью.

Данный алгоритм применяется как для задачи классификации так и для регрессии. Рассмотрим пример.

Подключение алгоритма и создание классификатора выполняются командами:

```
from sklearn.svm import SVC
SVC = SVC(kernel = 'rbf')
SVC.fit(x_train, y_train)
y_pred = SVC.predict(x_test)
```

Ансамблевые методы

Методы машинного обучения, объединяют несколько моделей для решения одной задачи. Суть метода показать, что объединение нескольких алгоритмов дает лучший результат, чем каждый метод по отдельности. К популярным видам данного метода относятся:

- Стекинг - объединяет несколько разных методов. В результате их обучения и объединения мы получаем прогноз, основанный на результатах различных методов.
- Бэггинг - один и тот же метод обучается на разных наборах данных, а затем объединяются. Полученные результаты усредняются.
- Бустинг - один и тот же метод исправляет ошибку предыдущего, путем последовательного обучения.