Машинное обучение в финансах

Лекция 7: Введение в машинное обучение

Роман В. Литвинов*

*CRO
Финансовая Группа БКС

Высшая школа экономики, Апрель 2021

Содержание

Содержание

Что такое машинное обучение?

Как мы гововорили с вами на первой лекции:

Машинное обучение (machine learning, ML)

Машинное обучение – набор методов/алгоритмов, позволяющих компьютерам обучаться на имеющихся в нашем распоряжении данных для того, чтобы делать (и улучшать сделанные) предсказания/прогнозы (predictions).

Что такое обучение? Когда мы можем сказать, что алгоритм способствует обучению машины?

Tom M. Mitchell (CMU) дал в свое время следующее хорошее определение, которое сейчас довольно распространено (Mitchell, 1997):

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."

Что такое машинное обучение?

Из этого определения можно сделать вывод, что ключевыми компонентами являются:

- некоторая задача/проблема, которую необходимо решить (T)
- мера оценки производительности/ успешности нашего решения (performance measure) (P)
- опыт (E)

Давайте попробуем рассмотреть эти, на первый взгляд абстрактные, понятия через призму нашего ежедневного (бытового) опыта.

Что такое машинное обучение?

Представим, что мы идем в магазин, чтобы преобрести яблоки. Как выглядят тогда эти компоненты?

- задача: приобрести качественные, например, не гнилые яблоки
- мера оценки производительности: после приоретения, мы съедаем яблоко и понимаем ошиблись (0) или нет (1)
- опыт: количество просмотренных/ приобретенных в супемаркете нами яблок

Мы учимся (!), если по мере наших походов в супермаркет и копания в яблоках (рост опыта), мы учимся все лучше и лучше выбирать качественные яблоки (меньше нулей, больше единиц).

Содержание

Основные задачи машинного обучения

Что касательно классов задач, есть смысл выделить следующие основные типы. В литературе вы найдете много примеров других задач, типа машинного перевода или детектирования аномалий, но я бы остановился на следующих двух. По моему мнению это две основные мета-задачи, которые покрывают подавляющее большинство примеров. Плюс, они актуальны для содержания нашего курса:

- регрессия (regression).
- классификация (classification).

В регрессионных задачах вам необходимо предсказать численную оценку заданной переменной.

Решению такой задачи соотвествует поиск функции $f : \mathbb{R}^n \to \mathbb{R}$.

Примеры: предсказание роста (человека), температуры, доходности акций.

Основные задачи машинного обучения

В задачах классификации вам нужно правильно указать к какому из n классов принадлежит заданный объект.

Формально это поиск функции $f : \mathbb{R}^n \to [1,2,...,n]$.

К примерам можно отнести, как выбор яблока (гнилое/не гнилое), так и классификация акций с точки зрения покупки или продажи (long/short signal).

Содержание

Оценка производительности/ успешности (performance measure)

Как мы говорили, для понимания того, прогрессируем мы или нет в решении нашей проблемы нам необходима объективная количественная оценка степени нашей успешности.

Самый простой пример такой метрики - это коэффициент точности (accuracy ratio). Который просто представляет собой отношение успешных предсказаний к общему количеству предсказаний.

Вспомним наш пример с яблоками. Если мы попытались предсказать качество десяти яблок и из десяти нами выбранных, гнилыми оказались восемь, коээфициент точности составит $2/10 = 0.2$ или только 20 процентов.

Если после 'перенастройки' нашей модели коээфициент точности составит 30 процентов. Прогресс будет очевиден (но мы по-прежнему будем ошибаться в семидесяти процентах случаев).

Оценка производительности/ успешности (performance measure)

Интуитивно напрашивающимся примером такой функции для регрессионной задачи будет показатель характеризующий насколько сильно мы отклонились от реального ответа (функция оценки расстояния?).

Например, если реальный рост 182 см, а предсказанный 180 см - это весьма неплохой результат. Ошибка составляет два см. А если прогноз составля 56 см на лицо крупная промашка.

Но здесь тоже возникает много вопросов. Если предсказаний много такие ошибки нужно складывать для общей оценки производительности модели/алгоритма. Что если один ошибки будут с положительным, а другие с отрицательным знаком? Что если алгоритм ошибается часто, но незначительно, но иногда на значительную величину? Как его лучше 'штрафовать' в таких случаях?

Оценка производительности/ успешности (performance measure)

Давайте посмотрим на распространенный пример оценки точности линейной регрессии - среднеквадратичная ошибка (mean square error).

$$MSE = \frac{1}{m}\sum_{i=1}^{n}(y_i - y_i^{pred})^2$$

Понятно, что такая ошибка будет равна нулю, если пресказанное значение равно фактически наблюдаемому $y_i - y_i^{pred}$. И чем меньше ошибка каждого предсказания, тем меньше будет значение MSE в целом.

Содержание

Опыт. Данные

Под опытом логично понимать, в нашем случае, данные. По мере обработки алгоритмом новых объемов данных наш опыт будет расти.

Наши данные будут состоять из набора количественных свойств (features) некоторого объекта или события. Таким образом, мы будем представлять единичное наблюдение (data point) как вектор $x \in \mathbb{R}^n$.

В таком случае общий набор данных (dataset) можно будет представить в виде матрицы m x n. Где m - количество наблюдений, а n - количество свойств/характеристик (features).

В отдельных случаях (и это ситуация, с которой мы будем иметь дело в ходе нашего курса) мы также имеем вектор-столбец размерности m, содержащий метки объекта y (labels) - те самые характеристики/значения, которые мы будем предсказывать с помощью обучающегося алгоритма. Такие данные называются размеченными.

Данные без меток называются, соответственно, неразмеченными.

Содержание

Виды машинного обучения

Выделяют следующие виды машинного обучения:

- обучение на размеченной выборке или обучение с учителем (supervised learning)
- обучение на неразмеченной выборке или обучение без учителя (unsupervised learning)
- обучение с подкреплением (reinforcement learning)

Из обозначенных выше, отдельного комментария требует парадигма reinforcement learning. В рамках такого обучения агент (обучающийся алгоритм) осуществляет свои действия в определенной заданной среде с целью получения максимальной выгоды (reward).

Последние громкие новости в области ML, например, AlphaGo, относятся именно к этому типу обучения.

Содержание

Основные модели

Основные модели supervised learning:

- линейная регрессия (linear regression)
- логистическая регрессия (logistic regression)
- метод опорных векторов (support vector machines - SVM)
- метод к-ближайших соседей (k-nearest neighbors method - KNN)
- деревья решений (decision trees)
- нейронные сети (neural networks)

Основные модели unsupervised learning:

- анализ принципиальных компонент (principal component analysis - PCA)
- метод к-средних (k-means clustering)
- нейронные сети (neural networks)

Содержание

Оптимизация. Градиентный спуск

Как сделать наш алгоритм действительно обучающимся? Нам нужно чтобы базовые параметры модели (например, веса в уравнении линейной регрессии) в процессе обучения и накопления опыта изменялись бы таким образом, что производительность модели увеличивалась, а ошибки минимизировались.

Это представляет собой оптимизационную задачу, связанную с минимизацией функции, отвечающей за оценку ошибки (MSE в примере выше).

В оптимизации, функция, которую мы хотим минимизировать или максимизировать называется целевой функцией (objective function). Функция, которую хотим именно минимизировать - фукция потерь (loss function, cost function).

Как мы с вами помним из второй лекции, задача поиска минимума функции связана с вычислением ее производной.

Оптимизация. Градиентный спуск

Как мы помним, производная функции показывает, насколько изменяется f(x) при небольшом изменении x.

Если мы будем двигаться следующим образом:

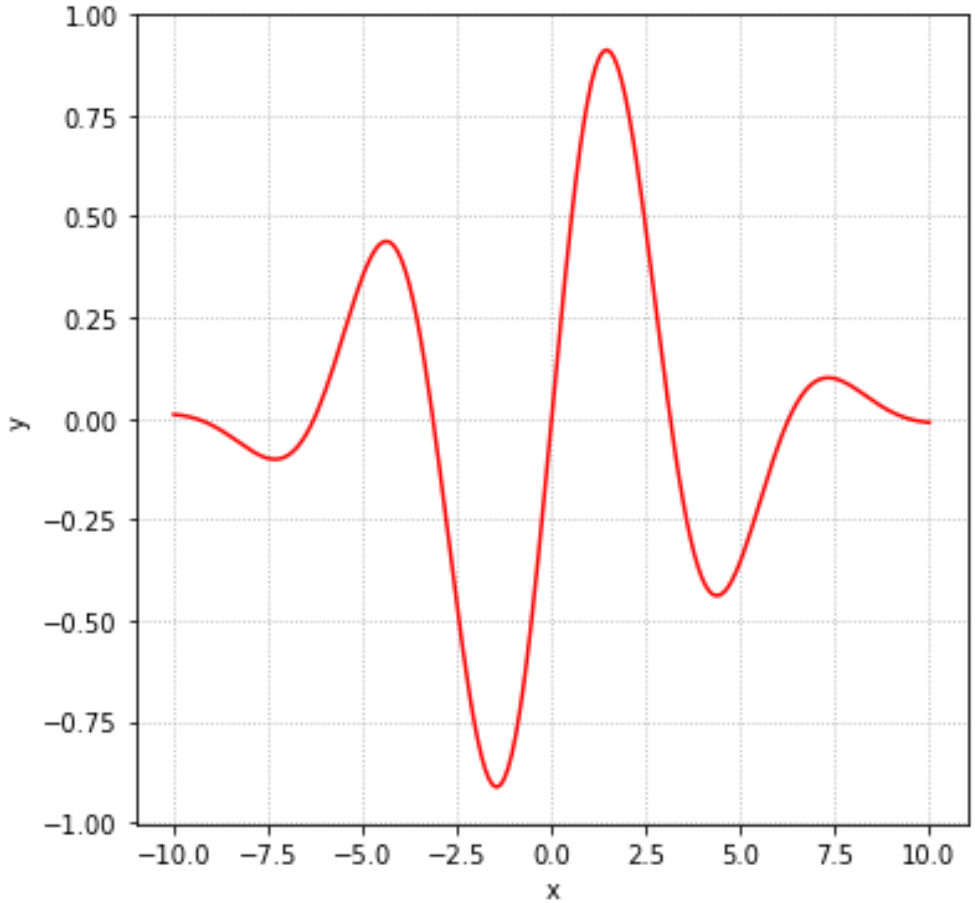
- для $f'(x) < 0$ будем увеличивать значение x (двигаться вперед)
- для $f'(x) > 0$ будем уменьшать значение x (двигаться назад)

То в конце концов найдем (локальный) минимум функции $f'(x) = 0$.

Такая техника называется градиентным спуском (gradient descent) (Коши, 1847).

Ниже график, который мы уже видели на второй лекции. Теперь его можно рассмотреть в контексте градиентного спуска и оптимизации параметров модели.

Оптимизация. Градиентный спуск



Точки роста/убывания функции. Точки максимума/минимума (локального/ глобального)

Оптимизация. Градиентный спуск

Если мы имеем дело с функцией от нескольких переменных, то здесь приходит на помощь понятие частной производной. Как мы говорили ранее, мы изменяем лишь одну переменную, а остальные оставляем без изменений и получаем предел отношения приращения функции по выбранной переменной к приращению этой переменной.

$$\frac{\partial f}{\partial x_1} = \lim_{\Delta x_1 \to 0} \frac{\Delta f(x_1 + \Delta x_1, x_2, ..., x_n)}{\Delta x_1}$$

Градиент (gradient) представляет собой вектор содержащий все частные производные функции. Обозначается $\nabla_x f(x)$.

Критическими точками такой функции будут точки, в которых каждый элемент градиента принимает нулевое значение.

Оптимизация. Градиентный спуск

Градиентный спуск будет работать в таком случае следующим образом. Новое значение вектора содержащего независимые переменные будет равно:

$$x' = x - \lambda \nabla_x f(x)$$

.

Параметр λ здесь представляет собой, так называемый, коэффициент скорости обучения (learning rate). Этот коэффициент задает насколько агрессивно/ быстро мы будем двигаться в направлении противоволожном знаку градиента/производной.

Стохастический градиентный спуск (stochastic gradient descent) представляет собой разновидность алгоритма градиентного спуска. О нем мы поговорим подробнее в соотвествующей части курса.

Содержание

Переобучение (overfitting) / недообучение (underfitting)

Как вы понимаете, нам важно, чтобы алгоритм имел минимальную ошибку не только на тех данных, на которых он обучался, но также и на новых данных, которые он до этого не видел.

Способность успешно решать проблему на основании данных, которые не были доступны в процессе обучения, называется генерализацией или обобщением (generalization).

Процесс обучения алгоритма строится следующим образом. Мы имеем некоторый доступный для обучения объем данных (training set), на которых обучаем алгоритм. Оценка ошибки, которую мы получаем в итоге на тренировочных данных называется тренировочной ошибкой (training error).

После обучения, чтобы понять насколько хорошо алгоритм способен к обобщению, мы тестируем его производительность на данных, которые ему были недоступны в процессе обучения - тестовой выборке (test set). Полученная ошибка называется тестовой (testing error).

Переобучение (overfitting) / недообучение (underfitting)

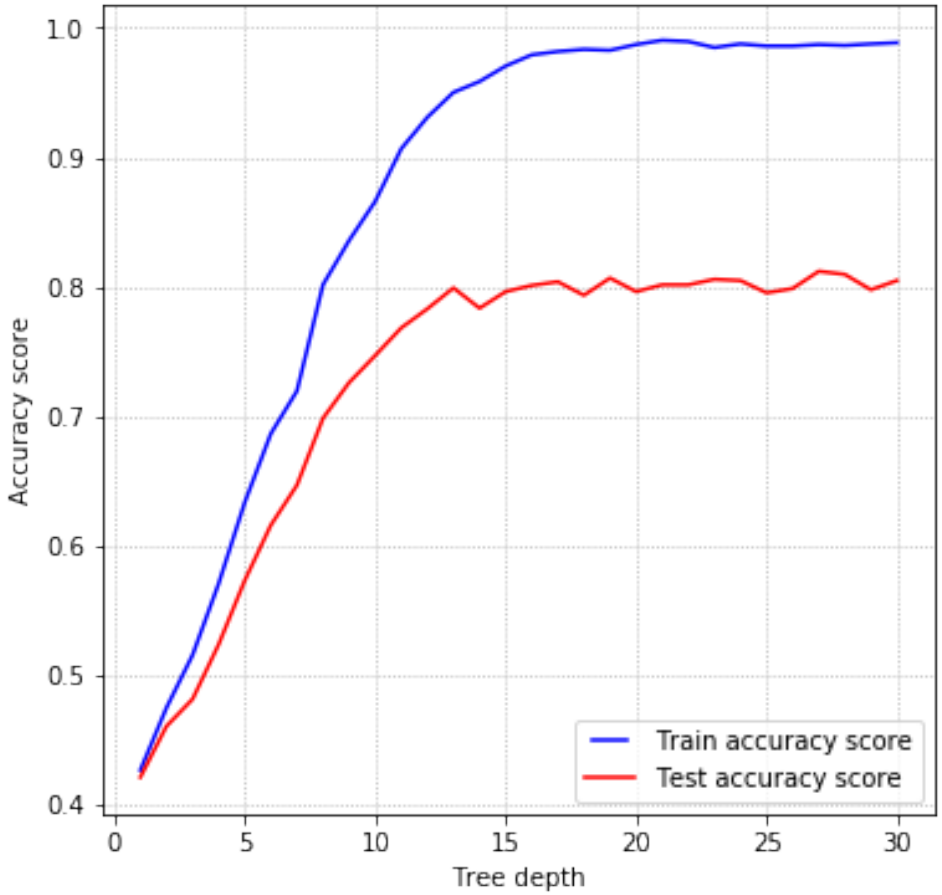
После такого тестирования, мы будем считать, что алгоритм успешно справляется с задачей, если:

- тренировочная ошибка (training error) мала
- раница между тренировочной и тестовой ошибкой (testing error) незначительна

Когда не соблюдается первый пункт, понятно, что модель неспособна успешно решать задачу даже на тренировочных данных. Такая модель называется недообученной (underfitted).

Если первый пункт соблюдается, а второй - нет. То такая модель успешно работает на тренировочных данных, но сбоит на тестовых данных, данных которые она не видела в процессе обучения. Модель не способна к обобщению и работе в 'боевых' условиях. Такая модель называется переобученной (overfitted).

Переобучение (overfitting) / недообучение (underfitting)



С такого рода примером мы будем сталкиваться постоянно. Давайте разберемся с тем, что здесь происходит.

Содержание

Фундаментальные блоки ML

Я бы подытожил то, что мы сегодня обсуждали, следующим образом. Машинное обучение (в тч глубокие нейронные сети) невозможно без следующих фундаментальных блоков:

- данных, на которых алгоритм будет обучаться/тестироваться
- целевой функции - loss/cost function (квадрат отклонений, напр. – squared error loss)
- оптимизационной процедуры/метода (optimization routine), которая с использованием тренировочных данных найдет необходимое решение для выбранного нами критерия
- непосредственно модели (регрессионное уравнение, спецификация нейронной сети и тп)

Литература

Дополнительная литература к сегодняшней лекции:

- Deisenroth, M. Fisal, A., Ong, C.S. (2020) Mathematics for machine learning. Cambridge.
- Goodfellow, I. Bengio, Y., Courville, A. (2016). Deep learning. MIT. (есть на русском)
- Hastie, T., Tibshirani, R., Friedman, J. (2017). The elements of statistical learning. Springer. (есть на русском)
- James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). An introduction to statistical learning. Springer. (есть на русском)
- Kuhn, M., Johnson, K., (2016). Applied predictive modeling. Springer. (есть на русском)
- Mitchell, T. (1997). Machine Learning. McGraw Hill.