Занятие 5

Ансамбли деревьев

Ключевая идея ансамблевых методов

Цель

Построить много разных (!) деревьев и усреднить предсказания → Много слабых алгоритмов должны создать один сильный и помочь избежать переобучения

Методы построения ансамблей

Равное голосование

1. Изменять используемые наборы для обучения
2. Изменять условия построения классификаторов

→ Bagging → Random Forest

Взвешенное голосование

3. Последовательно создавать классификаторы, каждый из которых будет придавать больший вес ошибочно классифицированным данным. → Adaptive Boosting

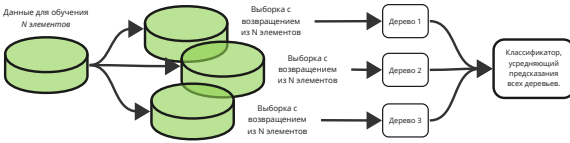
Деревья создают единое предсказание.

4. Последовательно создавать классификаторы. Каждый будет предсказывать ошибку предыдущей версии ансамбля → Gradient Boosting

(Каждое из деревьев по отдельности предсказывать целевую переменную не умеет)

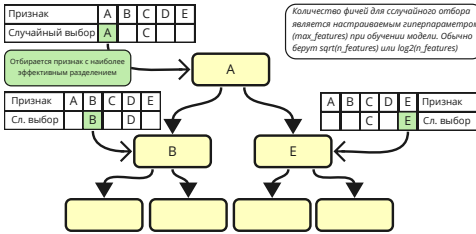
Бэггинг

Бэггинг (Bagging = **B**ootstrap **agg**regat**ing**) заключается в создании выборок с повторениями из исходного датасета и обучении деревьев на них.



Случайный лес: Бэггинг + Случайности при построении деревьев

Чтобы построить отличающиеся друг от друга классификаторы, предлагается при разделении выбирать оптимальный признак не из всех фичей, а из их подвыборки.

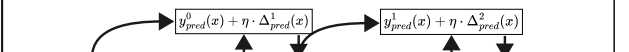


Зачем знать что-то ещё, если есть случайный лес?

К сожалению, случайный лес не гарантирует минимизации функции потерь.

Занятие 5
Ансамбли деревьев

Градиентный бустинг



x	y_{true}	$y^0_{pred}(x)$	$\Delta^1_{true} = y_{true} - y^0_{pred}(x)$	$\Delta^1_{pred}(x)$	$y^1_{pred}(x)$	$\Delta^2_{true} = y_{true} - y^1_{pred}(x)$	$\Delta^2_{pred}(x)$	$y^2_{pred}(x)$	...	$y^k_{pred}(x)$
750	1160	1418	-258	-145.5	1272.5	-258	-258	1180	...	1180
800	1200	1418	-218	-145.5	1272.5	-218	-218	1180	...	1180
850	1280	1418	-138	-145.5	1272.5	-138	-138	1334.2	...	1334.2
900	1450	1418	32	-145.5	1272.5	32	32	1334.2	...	1334.2
950	2000	1418	582	582	2000.5	582	582	2061.7	...	2061.7

Таким образом, конечное предсказание ансамбля можно записать следующим образом:

$$y^k_{pred}(x) = y^0_{pred}(x) + \eta \cdot \Delta^1_{pred}(x) + \ \ldots + \eta \cdot \Delta^k_{pred}(x)$$

где η — learning rate (коэффициент скорости обучения)

😱 **ШОК! Оказалось, что для квадратичной функции потерь обучение дерева предсказывать разность между предсказанием и таргетом идентично обучению дерева предсказывать антиградиент функции потерь, посчитанный по текущим предсказаниям ансамбля:**

$$Loss(y_{true\ i}, y_{pred\ i}) = \frac{1}{2}(y_{true\ i} - y_{pred\ i})^2$$

$$\frac{\partial Loss(y_{true\ i}, y_{pred\ i})}{\partial y_{pred\ i}} = -(y_{true\ i} - y_{pred\ i})$$

$$\underbrace{y_{true\ i} - y_{pred\ i}}_{\Delta_i} = -\frac{\partial Loss(y_{true\ i}, y_{pred\ i})}{\partial y_{pred\ i}}$$

i — номер обучающего примера

Поэтому градиентный бустинг и называется градиентным. 😎

Поскольку для задачи классификации разность между предиктом и таргетом не является целевой метрикой, как в регрессии, подход с построением деревьев, предсказывающих антиградиент функции потерь, мы можем использовать точно так же!

Гиперпараметры ансамблевых методов, которые необходимо настраивать.

Random Forest:
Главные гиперпараметры (стоит подобрать, т.к. значение по умолчанию редко бывает оптимальным):
🐝 **n_estimators (default=100)** — количество построенных деревьев. Чем больше, тем лучше. После критического количества деревьев результаты перестанут улучшаться и строить дальше нет смысла (т.к. увеличивается вычислительная сложность)
🐝 **max_features (default='sqrt')** — случайное количество признаков, которое выбирается для каждого разделения. Если поставить None, то модель станет Бэггингом. Стоит уменьшить, если наблюдается переобучение.

Критерии **max_depth** стоит уменьшить, если наблюдается переобучение. Критерии **min_samples_leaf** и **min_samples_split** стоит увеличить, если наблюдается переобучение. Но обучно эти параметры вообще не перебирают.

GradientBossting
🐝 **n_estimators (default=100)**. Большое количество деревьев приводит к переобучению!
Стоит уменьшить, если наблюдается переобучение.
🐝 **max_features(default='sqrt')** Стоит уменьшить, если наблюдается переобучение.
🐝 **learning_rate (default =).** Чем больше learning_rate, тем меньше нужно n_estimators. Рекомендуется выбрать небольшой learning_rate <= 0.1 и оставить его в покое.
🕊️ Рекомендуется строить обучающую кривую для уменьшения переобучения

Feature Importance

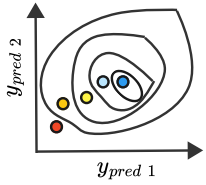
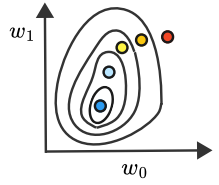
Для каждого построенного дерева мы мы можем присвоить каждому признаку ранг, который соответствует порядковому номеру разделения. Далее мы можем усреднить предсказанные ранги по всем деревьям. Получим некую характеристику важности признака. Чем больше Feature Importance, тем в среднем выше данный признак в каждом решающем дереве. Есть ещё permutation_importance, которая часто даёт хорошие результаты

Занятие 5
Приложение

Разница между бустингом и нейронными сетями

Разница между нейронными сетями (линейными моделями) и градиентным бустингом состоит в том, что в нейронных сетях функция потерь минимизируется по пространству параметров (модели), а в бустинге лосс минимизируется по пространству предсказаний (никаких обучаемых параметров нет).

То есть, фактически, нейронные сети и линейные методы, градиентный бустинг, решающие деревья делают одно и то же только разными способами! 😉

Градиентный бустинг	Линейная регрессия
$Loss(y_{pred\ 1}, \dots, y_{pred\ N}) =$ $= \frac{1}{N}\sum_{i=1}^{N}\left(y_{true\ i} - y_{pred\ i}\right)^2 \to min$	$Loss(w_0, w_1, .., w_N) =$ $= \frac{1}{N}\sum_{i=1}^{N}\left(y_{true\ i} - \underbrace{(w_0 + \dots + w_M \cdot x_{iM})}_{y_{pred\ i}}\right)^2 \to min$
	

Нейронные сети называются параметрическими моделями, а модели на решающих деревьях — непараметрическими (никакие параметры не обучаются).