Методы решения задачи многомерной экстраполяции:

1. Random forest decision tree

**Описание**: идея состоит в объединение нескольких деревьев решений, обученных на разных кусках данных. Подаем на вход набор признаков (в нашем случае точку), решение ищется на наборе деревьев, на выход получаем несколько верных ответов. В задачах регрессии усредняем ответы. Сами деревья представляют собой структуру из «листьев» и «веток». На рёбрах («ветках») дерева решения записаны атрибуты, от которых зависит целевая функция, в «листьях» записаны значения целевой функции, а в остальных узлах — атрибуты, по которым различаются случаи. Чтобы классифицировать новый случай, надо спуститься по дереву до листа и выдать соответствующее значение. Есть алгоритмы для построения деревьев.

CART – алгоритм построения дерева(для классификации):

На первой итерации мы строим все возможные (в дискретном смысле) гиперплоскости, которые разбивали бы наше пространство на два. Для каждого такого разбиения пространства считается количество наблюдений в каждом из подпространств разных классов. В результате выбирается такое разбиение, которое максимально выделило в одном из подпространств наблюдения одного из классов. Соответственно, это разбиение будет нашим корнем дерева принятия решений, а листами на данной итерации будет два разбиения.  
На следующих итерациях мы берем один худший (в смысле отношения количества наблюдений разных классов) лист и проводим ту же операцию по разбиению его. В результате этот лист становится узлом с каким-то разбиением, и двумя листами.  
Продолжаем так делать, пока не достигнем ограничения по количеству узлов, либо от одной итерации к другой перестанет улучшаться общая ошибка (количество неправильно классифицированных наблюдений всем деревом).

Сам Random forest

Суть алгоритма состоит в том, что на каждой итерации делается случайная выборка переменных, после чего, на этой новой выборке запускают построение дерева принятия решений. При этом производится “bagging” — выборка случайных двух третей наблюдений для обучения, а оставшаяся треть используется для оценки результата. Такую операцию проделывают сотни или тысячи раз. Результирующая модель будет будет результатом “голосования” набора полученных при моделировании деревьев.

Ссылки: <https://habr.com/ru/post/116385/> , <https://dyakonov.org/2016/11/14/%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D0%B9-%D0%BB%D0%B5%D1%81-random-forest/>

**Достоинства**: высокая точность, можно распараллелить обучение, сам определяет ветвления, то есть при наличии данных для обучения мы сможем получить какую-то новую специфическую зависимость, нечувствителен к пропускам данных

**Недостатки**: это алгоритм скорее классификации, чем регрессии, долгое построение модели для хороших результатов, вес каждого из деревьев большой

1. Бустинг

**Описание**: это просто метод комбинирования алгоритмов. Если, по описаниям, классически random forest формируется обучением на разных случайно-выбранных кусках данных, то здесь обучение происходит иначе. Мы обучаем одно дерево, смотрим, на каких случаях оно отрабатывает хуже всего, следующее дерево обучаем уже на этих «сложных» случаях и так далее. Так как это просто метод, то использовать можно не только деревья, комбинировать так же можно какие-либо виды регрессии.

Для деревьев этот алгоритм можно описать так:

Stochastic Gradient Boosting (Стохастическое градиентное добавление)   
На первой итерации строится ограниченное по количеству узлов дерево принятия решений. После чего считается разность между тем, что предсказало полученное дерево умноженное на learnrate (коэффициент “слабости” каждого дерева) и искомой переменной на этом шаге.  
Yi+1=Yi-Yi\*learnrate  
И уже по этой разнице строится следующая итерация. Так продолжается, пока результат не перестанет улучшаться. Т.е. на каждом шаге мы пытаемся исправить ошибки предыдущего дерева. Однако здесь лучше использовать проверочные данные (не участвовавшие в моделировании), так как на обучающих данных возможно переобучение.

**Достоинства**: этот способ точнее, чем бэггинг(используется для деревьев) и объем деревьев меньше.

**Недостатки**: более медленное обучение, так как для обучения деревьев можно пустить обучение разных деревьев в параллель, потому что они не будут зависеть друг от друга. Этот метод предполагает последовательное обучение каждого дерева.

1. Многомерная регрессия
   1. Линейная регрессия

**Описание**: строим гиперплоскость, наиболее близко расположенную к имеющимся в качестве входных данных точкам, используем линейную функцию

Пусть имеется набор n признаков fj (x), j = 1, . . . , n. Решение системы существенно упрощается, если модель алгоритмов линейна по параметрам α ∈ Rn:

Введём матричные обозначения: матрицу информации F, целевой вектор y, вектор параметров α и диагональную матрицу весов W:

В матричных обозначениях функционал среднего квадрата ошибки принимает вид

Функционал с произвольными весами легко приводится к функционалу с единичными весами путём несложной предварительной обработки данных F ′ = WF, y ′ = W y:

Поэтому в дальнейшем будем рассматривать только задачу с единичными весами.

* 1. Полиномиальная регрессия

**Описание:** строим гиперплоскость, наиболее близко расположенную к имеющимся в качестве входных данных точкам, используем полиномиальную функцию

Если X = R, а признаками являются всевозможные степени fj (x) = x j−1 , то говорят о полиномиальной регрессии:

В этом случае матрица F является матрицей Вандермонда:

* 1. Криволинейная регрессия.

**Описание:** строим гиперплоскость, наиболее близко расположенную к имеющимся в качестве входных данных точкам, используем некоторые преобразования над признаками

Следующим частным случаем является криволинейная регрессия, когда исходные признаки f1, . . . , fn подвергаются некоторым преобразованиям ϕ1, . . . , ϕk, в общем случае нелинейным:

Формально постановка задачи остаётся той же, если функции ϕj f1(x), . . . , fn(x) рассматривать как новые признаки ϕj (x). Матрица Fℓ×k называется обобщённой матрицей Вандермонд

* 1. Логистическая регрессия

**Описание:** строим гиперплоскость, наиболее близко расположенную к имеющимся в качестве входных данных точкам, используем логистическую функцию

**Методы решения**

Метод наименьших квадратов

Запишем необходимое условие минимума в матричном виде:

откуда следует

Эта система линейных уравнений относительно α называется нормальной системой для задачи наименьших квадратов. Матрица FтF имеет размер n × n и совпадает с ковариационной матрицей набора признаков f1, . . . , fn. Если она невырождена, то решением системы является вектор

Подставляя найденное решение в исходный функционал, получаем

Решение имеет простую геометрическую интерпретацию. Произведение PF y есть проекция целевого вектора y на линейную оболочку столбцов матрицы F. Разность (PF y −y) есть проекция целевого вектора y на ортогональное дополнение этой линейной оболочки. Значение функционала

есть длина перпендикуляра, опущенного из y на линейную оболочку. Таким образом, МНК находит кратчайшее расстояние от y до линейной оболочки столбцов

Сингулярное разложение

Если число признаков не превышает число объектов, n <= ℓ, и среди столбцов матрицы F нет линейно зависимых, то F можно представить в виде сингулярного разложения (singular value decomposition, SVD)

F = V DUт

1) ℓ × n матрица V ортогональна, VтV = In, составлена из n собственных векторов матрицы F Fт , соответствующих ненулевым собственным значениям;

2) n × n матрица U ортогональна, U тU = In, и составлена из собственных векторов матрицы F тF;

3) n × n матрица D диагональна, D = diag( . . . , ) , где λ1, . . . , λn — собственные значения матриц F тF и F Fт .

Имея сингулярное разложение, можно выписать решение задачи наименьших квадратов в явном виде, не прибегая к трудоёмкому обращению матриц:

F+ = (UDVт V DUт )−1UDVт = UD−1Vт

Обращение диагональной матрицы тривиально: D−1 = diag √ 1 λ1 , . . . , √ 1 λn .

Теперь легко выписать вектор МНК-решения:

α∗ = F+ y = UD−1Vт y

и МНК-аппроксимацию целевого вектора y:

F α∗ = PF y = (V DUт )UD−1Vт y = V Vт y =

Эта формула означает, что МНК-аппроксимация Fα∗ есть проекция целевого вектора y на линейную оболочку собственных векторов матрицы F Fт

**Ссылки**: <http://www.ccas.ru/voron/download/Regression.pdf>

**Достоинства**: простая модель

**Недостатки**: данные могут не иметь линейной/полиномиальной и т д зависимости, если данные коллинеарные, то решить СЛАУ будет затруднительно

1. Лассо

**Описание:** строим гиперплоскость, наиболее близко расположенную к имеющимся в качестве входных данных точкам, добавляем ограничение-неравенство, запрещающее слишком большие абсолютные значения коэффициентов

Функционал качества теперь выглядит вот так:

h — параметр регуляризации. При больших значениях h ограничение обращается в строгое неравенство, и решение совпадает с методом наименьших квадратов. Чем меньше h, тем больше коэффициентов αj принимают нулевое значение. Образно говоря, параметр h зажимает вектор коэффициентов, заставляя его отказываться от лишних степеней свободы. Отсюда и название метода — лассо (LASSO, least absolute shrinkage and selection operator).

**Ссылки**: <http://www.ccas.ru/voron/download/Regression.pdf>

**Достоинства**: решает проблемы мультиколлинеарности, переобучения, и уменьшают разброс коэффициентов

**Недостатки:** как и у любой регрессии, необходимо правильно выбрать ее тип

1. Гребневая регрессия

**Описание:** строим гиперплоскость, наиболее близко расположенную к имеющимся в качестве входных данных точкам, к функции потерь добавляем слагаемое, штрафующее большие значения нормы вектора весов

Для решения проблемы мультиколлинеарности припишем к функционалу Q дополнительное слагаемое, штрафующее большие значения нормы вектора весов ||α||:

Qτ (α) = ||F α – y||2 + τ||α||2 , где τ — неотрицательный параметр.

В случае мультиколлинеарности имеется бесконечно много векторов α, доставляющих функционалу Q значения, близкие к минимальному. Штрафное слагаемое выполняет роль регуляризатора, благодаря которому среди них выбирается решение с минимальной нормой. Приравнивая нулю производную Qτ(α) по параметру α, находим:

α∗ τ = (FтF + τ In)−1Fт y.

Таким образом, перед обращением матрицы к ней добавляется «гребень» — диагональная матрица τ In. Отсюда и название метода — гребневая регрессия (ridge regression). Добавление гребня к матрице FтF увеличивает все её собственные значения на τ , но не изменяет её собственных векторов. В результате матрица становится хорошо обусловленной, оставаясь в то же время «похожей» на исходную.

**Достоинства**: решает проблемы мультиколлинеарности, переобучения, и уменьшают разброс коэффициентов

**Недостатки:** как и у любой регрессии, необходимо правильно выбрать ее тип

1. Метод опорных векторов

**Описание**: строим разделяющую гиперплоскость , есть параметр C, который мы можем регулировать и следовательно регулировать кривизну гиперплоскости

Однако в некоторых случаях более естественно использовать кусочно-линейную функцию ε-чувствительности, показанную на Рис 3: |z|ε = max{0, |z| − ε}, которая не считает за ошибки отклонения a(xi) от yi , меньшие ε. Предполагается, что значение параметра ε задаёт эксперт, исходя из априорных соображений. С этой функцией потерь функционал принимает вид

|z|ε = max{0, |z| − ε}

где τ — параметр регуляризации. Выбор именно квадратичной функции потерь обусловлен удобством решения задачи наименьших квадратов

Положим C = 1/2τ . Введём дополнительные переменные ξ+i и ξ−i , значения которых равны потере при завышенном и заниженном ответе a(xi) соответственно

ξ+i = (a(xi) − yi − ε)+ , ξ−i = (−a(xi) + yi − ε)+ , i = 1,... ,ℓ

Тогда задача минимизации может быть переписана в эквивалентной форме как задача квадратичного программирования с линейными ограничениями-неравенствами относительно переменных wi , w0, ξ+i и ξ−i :

Решается как задача квадратичного программирования. Строим действенную, заменяем скалярное произведение на ядро и выражаем уравнение регрессии как:

**Ссылки**: <http://www.williamspublishing.com/PDF/978-5-9500296-2-2/part.pdf><http://www.ccas.ru/voron/download/SVM.pdf>

**Достоинства**: задача квадратичного программирования может быть решена эффективно

**Недостатки**: необходимы знания о предметной области, чтобы грамотно построить ядро, неустойчив по отношению к шуму