Яндекс

Matrixnet

Андрей Гулин
Конспиролог

Москва, 27.02.2010

Линейная регрессия

- Дано: K N-мерных самплов $\{x_i\}$ для каждого известно значение функции $\{f_i\}$
- Найти: вектор a, такой что $a^T x_i = f_i$
- Решение: $a=(X^T X)^{-1} X^T f$

Я

Регуляризация

- Когда данных мало простое решение не работает
- Нужна какая-то дополнительная информация, например, мы можем сказать, что мы хотим “маленький” или “простой” вектор a
- Меры простоты:
  - L0 = feature selection
  - L1 = lasso
  - L2 = ridge = по Тихонову [$a=(X^TX+\lambda I)^{-1}X^Tf$]

Я

L1 регуляризация

- Итеративный алгоритм L1 регуляризации
- У нас есть текущий “остаток” r_i, который в начале равен f_i
- На каждой итерации мы
  - Выбираем самый похожий на r_i фактор и считаем с каким множителем α нам нужно его брать
  - Добавляем $\lambda\alpha$ к коэффициенту при этом факторе ($\lambda < 1$)
  - Считаем новый остаток r_i
- http://www-stat.stanford.edu/~tibs/lasso.html

Нелинейные модели

— Если бы у нас были пропорциональные релевантности независимые факторы, нам бы хватило линейной регрессии

— Это не так и нам понадобятся нелинейные модели

  — Полиномиальные

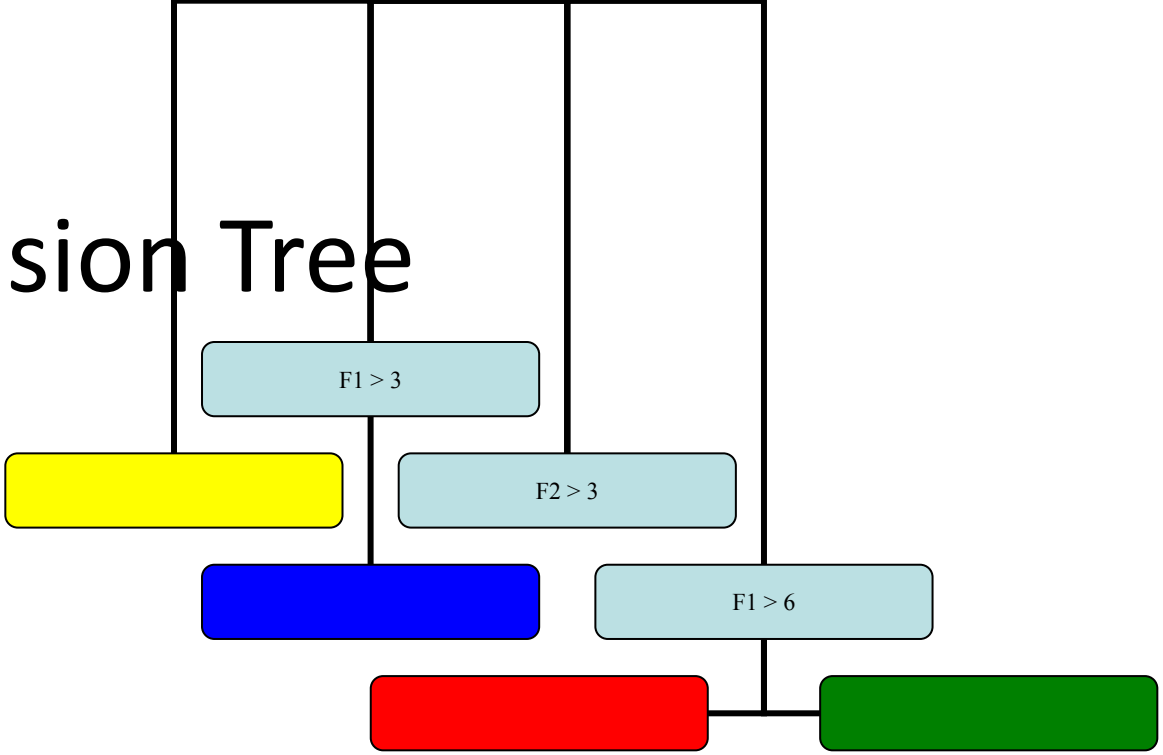
  — “Нейронные сети”

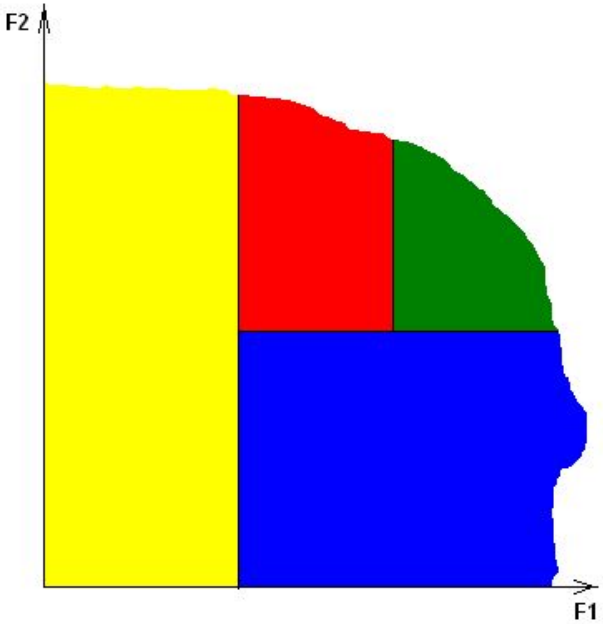
  — Decision Trees

  — …

Я

Decision Tree





Boosting

- Построение strong learner как комбинации “weak learners”
- Связь с L1 регуляризацией
  - weak learner = единственный фактор с коэффициентом
  - strong learner = линейная регрессия с L1 регуляризацией
- Для более сложного weak learner boosting дает сложно формализуемую sort of L1 регуляризацию

Я

Bagging

- На каждой итерации будем брать не все самплы, а их случайное подмножество
- Магическим образом более устойчиво
- Defeats boosting impossibility argument (http://velblod.videolectures.net/2008/pascal2/icml08_helsinki/long_rcn/icml08_long_rcn_01.ppt)

Я

Limit on decision tree leafs

- Дисперсия ошибки значения в листе пропорциональна 1/N, где N – количество самплов в листе
- Введем ограничение – в кошерном дереве должно быть не меньше 10 самплов обучающей выборки на лист
- Наш лимит ограничивает ошибку аппроксимации “выравнивая” ее по выборке

Я

TreeNet

- TreeNet товарища Friedman-а это Boosted Decision Tree с Bagging и ограничением на минимальное количество самплов в листе

- http://www.salford-systems.com/doc/GreedyFuncApproxSS.pdf

- http://www.salford-systems.com/doc/StochasticBoostingSS.pdf

Я

MatrixNet

http://seodemotivators.ru/

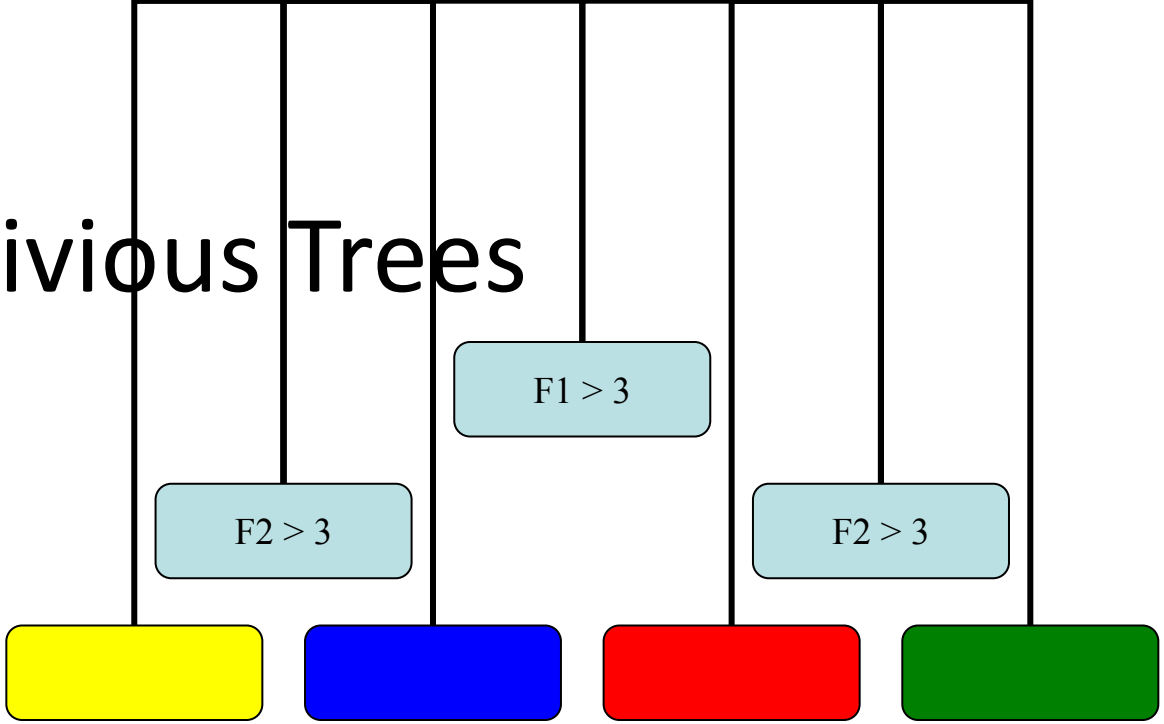
MatrixNet

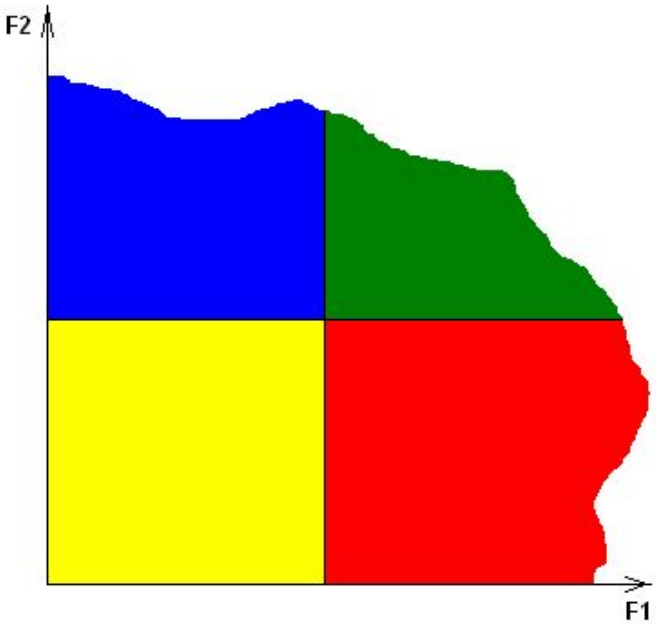
— MatrixNet отличается в 3-х моментах

- — Использование Oblivious Trees
- — Регуляризация значений в листах вместо ограничения на количество самплов в листе
- — Зависимость сложности модели от итерации (начинаем с простых моделей, заканчиваем сложными)

Я

Oblivious Trees





Регуляризация в листьях

- Вместо ограничения на количество самплов в листьях будем “регуляризовать” значение в листе
- Например, если домножить значение в листе на sqrt(N/(N+100)), где N – число самплов в листе, то результаты улучшатся.
- Оптимальный способ регуляризации, видимо, зависит от выборки

Я

Другие целевые функции

- А что, если вместо квадратичной ошибки мы хотим оптимизировать что-нибудь другое? Например, для задач классификации больше подходит средний log(p), где p – вероятность, назначенная моделью правильному ответу
- Получаем обычную задачу максимизации функции, которую можно решать
  - Градиентным спуском = gradient boosting = greedy function approximation
  - Методом Ньютона = logit boost для классификации

Я

Gradient boosting

- На каждом шаге boosting-а вместо невязки r_i мы аппроксимируем производную целевой функции в текущей точке

- Размер шага зависит от величины производной, т.е. от гладкости функции

- Вместо шага по производной в текущей точке мы можем посчитать куда приведет нас производная и шагать в направлении финальной точки траектории

Я

Ranking

- А что же делать, если мы хотим научиться ранжировать?
- Целевая функция для ranking (NDCG/pFound/whatever) задана на порядках и разрывна (описание pFound http://romip.ru/romip2009/15_yandex.pdf)
- Нужна какая-то непрерывная замена. Замены делятся на классы
  - Pointwise (rmse, классификация, …)
  - Pairwise (RankNet, …)
  - Listwise (SoftRank, …)

Я

Luce-Plackett model

— Luce-Plackett model позволяет нам назначить вероятности всем перестановкам, если у нас есть веса документов $\{w_i\}$

— Вероятность перестановки вычисляется рекурсивно. Вероятность поставить документ k на первое место равна $w_k / (w_1 + w_2 + \ldots + w_n)$, далее аналогично считаем вероятность выбрать второй документ из оставшихся и т.д. Произведение этих вероятностей равно вероятности перестановки.

Я

Expected pFound

- Для каждой перестановки мы можем посчитать ее pFound(perm). Также мы знаем вероятность этой перестановки P_{LP}(perm)
- Просуммировав pFound(perm) * P_{LP}(perm) по всем перестановкам получим expected pFound
- Expected pFound непрерывен и мы можем максимизировать его с помощью gradient boosting
- Вместо pFound мы можем подставить любую нужную нам меру

Я

Вопросы?

Akasaka Mitsuki
Everyone possesses many different faces.
For living...
For working...
And for falling in love...
I wonder how many faces I have?
And which one is the real me?
ANIMEWALLPAPERS.COM

Я