Задание «Композиции алгоритмов»

Срок сдачи – 16 апреля 2014 г. (среда), 23:59

Задание направлено на ознакомление студентов с методами построения композиций алгоритмов Bagging и Gradient Boosting, а также на изучение того, какие базовые алгоритмы лучше подходят для каждого из этих алгоритмов.

Bagging

1. Реализуйте композицию алгоритмов классификации при помощи Bagging. В качестве базовых алгоритмов должны поддерживаться решающие деревья и SVM. На каждой итерации для обучения предлагается выбирать с возвращением выборку такого же размера, как исходная.
2. Проанализируйте качество классификации на обучающей и тестовой выборке по итерациям при использовании деревьев различной сложности, а также SVM с разделяющей поверхностью различной сложности.

Gradient Boosting

3. Реализуйте Gradient Boosting. В качестве базовых алгоритмов должны поддерживаться регрессионные деревья и epsilon-SVR. Должны быть реализованы следующие функции потерь:
   a. logistic loss, $\log\left(1 + \exp\left(-yf(x)\right)\right)$, y – правильный ответ (-1, +1), $f(x)$ – ответ алгоритма (вещественный)
   b. absolute deviation, $|y - f(x)|$, y – правильный ответ (вещественный), $f(x)$ – ответ алгоритма (вещественный)
4. Изучите качество классификации по итерациям. Базовые алгоритмы – деревья различной сложности, epsilon-SVR с разделяющей поверхностью различной сложности. Функция потерь – logistic loss. Сравните полученные результаты с Bagging. Какие базовые алгоритмы хорошо подходят для Bagging? Какие для Gradient Boosting?
5. Зашумите ответы (инвертируйте класс случайных объектов) так, чтобы Gradient Boosting начал переобучаться. Используйте regression stump как базовый алгоритм. Проанализируйте эффект от двух методов регуляризации: уменьшение learning rate и early stopping.
6. Решите задачу регрессии Housing Data Set с функцией потерь L1-loss. Базовый алгоритм – деревья различной сложности. Проанализируйте результаты.

Отчёт и сдача задания

7. Написать отчёт в формате PDF: вывод градиентов функций потерь для Gradient Boosting и описание всех проведённых исследований.
8. Отправить семинаристу реализованные функции, отчёт, и код, воспроизводящий графики из отчёта. Файлы LibSVM присылать не нужно.

Замечания по реализации

Можно использовать реализацию деревьев из MATLAB, классы ClassificationTree и RegressionTree.Для управления сложностью дерева можно использовать параметр MinParent (если он равен числу объектов, получаем decision/regression stump; при его уменьшении будет расти размер дерева, вплоть до полного дерева). Обратите внимание при кросс-валидации, что для получения stump, MinParent должен быть равен числу выбранных обучающих объектов, а не размеру исходной выборки.

Реализацию SVM и epsilon-SVR можно взять из LibSVM. Сложность разделяющей поверхности регулируется в первую очередь параметром ядра gamma.

Использовать чужую реализацию Bagging и Gradient Boosting нельзя.

Для экспериментов предлагается использовать следующие датасеты:

1. Для классификации – Ionosphere Data Set, http://archive.ics.uci.edu/ml/datasets/Ionosphere . В MATLAB можно использовать команду “load ionosphere”.
2. Для регрессии – Housing Data Set, http://archive.ics.uci.edu/ml/datasets/Housing .

Для анализа качества пользуйтесь 5-fold cross-validation.

Спецификация реализуемых функций

```
[ model ] = bagging_train( X, y, num_iterations, base_algorithm, param_name1, param_value1, ... )
```

Построение ансамбля при помощи Bagging. Решается задача классификации. Решение принимается по большинству голосов, в случае равенства числа голосов – выбираем +1.

Вход:
X — матрица объект-признак NxK типа double, где N – число объектов, K – число признаков.
y — вектор ответов Nx1 типа double. Элементы принимают значения -1, +1.
num_iterations — число базовых алгоритмов в ансамбле.
base_algorithm — базовый алгоритм. Строка, либо ‘classification_tree’, либо ‘svm ‘.
(param_name, param_value) – набор параметров:
‘min_parent’ – минимальное число объектов в листе для решающего дерева;
‘gamma’, ‘C’ – параметры SVM.

Выход:
model – обученный ансамбль

```
[ prediction, err ] = bagging_predict( model, X, y )
```

Прогнозирование и оценка качества Bagging ансамбля.

Вход:
model – обученный ансамбль

Выход:
prediction – матрица num_iterations x N, ответы ансамбля на каждой итерации для каждого объекта
err – вектор num_iterations x 1, доля неправильно классифицированных объектов на каждой итерации.

```
bagging_experiment( X, y, num_iterations, base_algorithm, model_complexity )
```

Проведение эксперимента с фиксированными параметрами для Bagging (с 5-fold cross-validation), вывод графика ошибки на обучении и контроле по итерациям.

Параметр model_complexity должен управлять сложностью базового алгоритма, чем он больше, тем сложней алгоритм. Может быть дискретным и подобран под задачу.
Можно добавить свои параметры.

```
[ model ] = gradient_boosting_train( X, y, num_iterations, base_algorithm, loss, param_name1, param_value1, ... )
```

Построение ансамбля при помощи Gradient Boosting. Решается задача классификации или регрессии, в зависимости от параметра loss. Используется адаптивный шаг, то есть после настройки алгоритма решается задача одномерной минимизации.

Вход:
X — матрица объект-признак NxK типа double, где N – число объектов, K – число признаков.

y — вектор ответов Nx1 типа double. Для задачи классификации элементы принимают значения -1, +1. Для задачи регрессии – любые вещественные числа.
num_iterations — число базовых алгоритмов в ансамбле.
base_algorithm — базовый алгоритм. Строка, либо ‘regression_tree’, либо ‘epsilon_svr‘.
loss — функция потерь. Строка, возможные значения: ‘logistic’ – logistic loss, задача классификации; ‘absolute’ – absolute deviation, задача регрессии.
(param_name, param_value) – набор параметров:
‘learning_rate’ – коэффициент, на который домножается оптимальный шаг. По умолчанию 1.0
‘min_parent’ – минимальное число объектов в листе/узле дерева для решающего дерева;
‘epsilon’, ‘gamma’, ‘C’ – параметры epsilon-SVR.

Выход:
model – обученный ансамбль

```
[ prediction, err ] = gradient_boosting_predict( model, X, y )
```

Прогнозирование и оценка качества Gradient Boosting ансамбля.

Вход:
model – обученный ансамбль

Выход:
prediction – матрица num_iterations x N, ответы ансамбля на каждой итерации для каждого объекта
err – вектор num_iterations x 1. Если функция потерь – ‘logistic’, то доля неправильно классифицированных объектов на каждой итерации. Если функция потерь – ‘absolute’, то средний модуль разности истинного ответа и предложенного.

```
gradient_boosting_experiment( X, y, num_iterations, base_algorithm, loss,
model_complexity, learning_rate )
```

Проведение эксперимента с фиксированными параметрами для Gradient Boosting (с 5-fold cross-validation), вывод графика ошибки на обучении и контроле по итерациям.

Параметр model_complexity должен управлять сложностью базового алгоритма, чем он больше, тем сложней алгоритм. Может быть дискретным и подобран под задачу.
Можно добавить свои параметры.