Отчет по лабораторной работе №2
по курсу «Искусственный интеллект»

Студент группы 8О-407 Зайцев Никита, № по списку 7

Метод — KNN
По определённой метрике считается расстояние от текущего примера до обучающих примеров, из k примеров с наименьшим расстоянием выбирают тот класс, который встречается чаще всего.

Код

```
class KNN(object):

    def __init__(self,*pargs,**kwargs):
        train_data = []

    def _metric(self, a, b):
        sum_quad = 0

        for i in range(len(a)):
            sum_quad += (b[i]-a[i])**2
        return math.sqrt(sum_quad)

    def fit(self,x,y):
        self.train_data = [i for i in zip(x,y)]

    def predict(self,x, k=10):
        result = []
        for test in x:
            res = {}
            testRes = [(self._metric(i[0], test), i[1]) for i in
self.train_data]

            for d in sorted(testRes, key=lambda kk: kk[0])[:k]:
                if d[1] in res:
                    res[d[1]] += 1
                else:
                    res[d[1]] = 1
            result.append(max(res.items(), key=lambda x: x[1])[0])
        return(result)
```

Точность
Численные данные были нормированы с помощью min-max нормализации.
В текстовых данных были выброшены некоторые стопслова, не влияющие на результат и текст преобразован в вектор частоты того или иного слова.
На численных данных точность составила 0.969, а на текстовых 0.5
Точность sklearn составляет 0.987 и 0.5 соответственно. Следовательно показатели моей реализации и sklearn показывают одинаковую точность. На текстовых данных точность не лучше чем при случайном выборе класса.

Сходство и отличие с sklearn
Судя по описанию алгоритма на официальном сайте sklearn алгоритмы работают схожим образом, поэтому выдают почти одинаковые результаты, возможно отличается метрика. В sklearn есть возможность использовать взвешенный способ, но он не использовался. Основным отличием является скорость работы.

Время работы

Сложность алгоритма оценивается как O(K*M*N), где K,M,N соответственно размерность пространства, размер обучающей выборки, размер тестовой выборки. Реализация в 5-6 раз медленнее, чем на sklearn

Полиномиальная регрессия

```
class PolyReg(object):
    def __init__(self,*pargs,**kwargs):
        beta, powers = None, None

    def fit(self, xs, y, deg, model_out=False, powers_out=False):
        y = asarray(y).squeeze()
        rows = y.shape[0]
        xs = asarray(xs)
        num_covariates = xs.shape[1]
        xs = hstack((ones((xs.shape[0], 1), dtype=xs.dtype) , xs))

        generators = [self.basis_vector(num_covariates+1, i)
                                  for i in range(num_covariates+1)]

        powers = map(sum,
itertools.combinations_with_replacement(generators, deg))

        A = hstack(asarray([self.as_tall((xs**p).prod(1)) for p in
powers]))

        beta = linalg.lstsq(A, y)[0]

        self.beta = beta
        self.powers = powers

    def predict(self, args):
        if(self.beta == self.powers == None):
            return -1
        num_covariates = len(self.powers[0]) - 1
        if len(args)!=(num_covariates):
            raise ValueError()
        xs = asarray((1,) + tuple(args) )
        return sum([coeff * (xs**p).prod()
                                 for p, coeff in zip(self.powers,
self.beta)])

    def basis_vector(self, n, i):
        x = zeros(n, dtype=int)
        x[i] = 1
        return x

    def as_tall(self, x):
        return x.reshape(x.shape + (1,))
```

Точность

На численных данных точность составила 0.88 и 0.935 для полинома 4 и 6 степеней соответственно.