План работы

- Теория: LinReg, SVM, LogReg, ANN + SGD
- Case studies (из документации и Kaggle)
- Теория: Decision Trees, RandomForest, Boosting
- Case studies (из документации и Kaggle)
- Теория: BFGS

Функция потерь

$$Q(w) = \frac{1}{n}\sum_{i=1}^{n} Q_i(w)$$

Стохастический градиентный спуск:

1. Выбрать начальный вектор параметров w и интенсивность обучения η

2. Повторять до сходимости:

   (a) Случайно перемешать выборочные данные

   (b) Для $i = 1, 2, \ldots, n$ выполнить
   $$w := w - \eta \nabla Q_i(w)$$

```
# SGD for simple linear regression
import numpy as np
import random
def NablaQ(beta, x, y):
  return np.array([2*(beta[0] + beta[1]*x - y),
    2*x*(beta[0] + beta[1]*x - y)])
def SGD(x, y, NablaLoss, beta, eta=0.001, iters=2000):
  for i in range(iters):
    for xi, yi in random.sample(zip(x, y), len(x)):
      beta = beta - eta * NablaLoss(beta, xi, yi)
      print beta
x=[10.0,8.0,13.0,9.0,11.0,14.0,6.0,4.0,12.0,7.0,5.0]
y=[8.04,6.95,7.58,8.81,8.33,9.96,7.24,4.26,10.84,4.82,5.68]
SGD(x, y, NablaQ, beta=np.array([1, 1]))
```

Метод опорных векторов
Support vector machines

- Случай линейной разделимости:

$$\min_{w \in R^d} \|w\|^2$$

$$y_i(w^T x_i + b) \geq 1, \quad i = 1 \ldots N$$

- Общий случай

$$\min_{w \in R^d, \xi_i \in R^+} \|w\|^2 + C \sum_{i=1}^{N} \xi_i$$

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1 \ldots N$$

Ограничение

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

равнозначно

$$y_i f(x_i) \geq 1 - \xi_i,$$

что равнозначно (с учетом $\xi_i \geq 0$)

$$\xi_i = \max(0, 1 - y_i f(x_i))$$

Получаем задачу оптимизации без ограничений:

$$\min_{w \in R^d} C \sum_{i=1}^{N} \max(0, 1 - y_i f(x_i)) + \|w\|^2$$

SVM + SGD

$$Q(w) = \frac{1}{N}\sum_{i=1}^{N} Q_i(w); \qquad w := w - \eta \nabla Q_i(w)$$

Запишем функцию потерь в виде суммы:

$$\min_w Q(w) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\lambda}{2}\|w\|^2 + \max(0, 1 - y_i f(x_i))\right),$$

где $\lambda = 2/(NC)$, $f(x) = w^T x + b$.

Вместо градиента возьмем субградиент, получим

$$w := \begin{cases} w - \eta(\lambda w - y_i x_i), & \text{если } y_i f(x_i) < 1, \\ w - \eta\lambda w, & \text{в противном случае.} \end{cases}$$

Support Vector Classifier

```
sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3,
gamma='auto', coef0=0.0, shrinking=True,
probability=False, tol=0.001, cache_size=200,
class_weight=None, verbose=False,
max_iter=-1, decision_function_shape='ovr', random_state=None)
```

```
# SVM: Maximum margin separating hyperplane
# From scikit-learn docs

import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

# we create 40 separable points
np.random.seed(0)
X = np.r_[np.random.randn(20, 2) - [2, 2],
np.random.randn(20, 2) + [2, 2]]
Y = [0] * 20 + [1] * 20

# fit the model
clf = svm.SVC(kernel='linear')
```

```
clf.fit(X, Y)

# get the separating hyperplane
w = clf.coef_[0]
a = -w[0] / w[1]
xx = np.linspace(-5, 5)
yy = a * xx - (clf.intercept_[0]) / w[1]

# plot the parallels to the separating hyperplane
# that pass through the support vectors
b = clf.support_vectors_[0]
yy_down = a * xx + (b[1] - a * b[0])
b = clf.support_vectors_[-1]
yy_up = a * xx + (b[1] - a * b[0])
```

```
# plot the line, the points, and the nearest vectors to the plan
plt.plot(xx, yy, 'k-')
plt.plot(xx, yy_down, 'k--')
plt.plot(xx, yy_up, 'k--')

plt.scatter(clf.support_vectors_[:, 0],
clf.support_vectors_[:, 1], s=80, facecolors='none')
plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.Paired)

plt.axis('tight')
plt.show()
```

Логистическая регрессия
(классификация, а не регрессия!)

$$\frac{1}{2}w^T w + C \sum_{i=1}^{n} \log\left(\exp\left(-y_i(x_i^T w + c) + 1\right)\right) \to \min_w$$

```
sklearn.linear_model.LogisticRegression(penalty='l2',
dual=False, tol=0.0001, C=1.0, fit_intercept=True,
intercept_scaling=1, class_weight=None, random_state=None,
solver='liblinear', max_iter=100, multi_class='ovr',
verbose=0, warm_start=False, n_jobs=1)
```

Логистическая регрессия

- Вопреки названию, является методом классификации, а не регрессии (так же, как и SVM)

- Значения параметра solver: ‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’, sag и saga – варианты SGD.

- Можно найти сходства логистической регрессии как с SVM, так и с линейной регрессией.

- Логистическая регрессия – частный случай перцептрона (простейшей ИНС)

Домашнее задание

1. Составьте программу, реализующую SGD для SVM

2. Напишите формулу шага SGD для логистической регрессии ($w := w + \eta \ldots$)