The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Jiaming Li

Supervisor:
Qingyao Wu

Student ID：
201530611975

Grade:
Undergraduate

December 15, 2017

Linear Regression, Linear Classification and Gradient Descent

Abstract— using different optimization methods to verify the logic regression and linear classification

I. Introduction

1. Compare and understand the difference and relationship between gradient descent and stochastic gradient descent
2. Compare and understand the difference and relationship between logistic regression and linear classification.
3. Further understand the principle of SVM and practice on larger data.

II. Methods and Theory

1. Initalize logistic regression model parameters, you can consider initalizing zeros, random numbers or normal distribution.

2. Select the loss function and calculate its derivation

3. Calculate gradient toward loss function from partial samples.

4. Update model parameters using different optimized methods(NAG，RMSProp，AdaDelta and Adam).

5. Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative. Predict under validation set and get the different optimized method loss

6. Repeate step 4 to 6 for several times, and drawing graph of loss， and with the number of iterations.

III. Experiment

A. Dataset

Experiment uses a9a of LIBSVM Data, including 32561/16281(testing) samples and each sample has 123/123 (testing) features. Please download the training set and validation set.

B. Implementation

1. RegressionExperiment

1. Initialization

Initalize logistic regression model parameters with zeros

```
# 加载数据
X_train, y_train = get_data(r"E:\machine learning\lab2\a9a")
X_test, y_test = get_data(r"E:\machine learning\lab2\a9a.t")

# 将验证集的122个特征末尾补0
zeros = np.zeros(16281)
X_test = np.c_[X_test, zeros]

# 初始化参数
learning_rate = 0.01
epoch = 300
```

2. Get Loss

Select the appropriate threshold, mark the sample whose predict scores **greater than the threshold as positive, on the contrary as negative**.

```python
def get_loss(X, y, w, b):
    loss = 0
    for i in range(X.shape[0]):
        y_ = sigmoid(np.dot(X[i].data, w) + b)
        if y_ > 0.5:
            y_ = 1
        else:
            y_ = -1
        loss += (y[i] - y_) * (y[i] - y_) * 0.5
    return loss / X.shape[0]
```

3. Compute gradient

```python
def compute_gradient(X, y, w, b, sample_indices):
    # loss_function 对w, b的偏导
    G_w = 0
    G_b = 0
    for i in sample_indices:
        y_ = sigmoid(np.dot(X[i].data, w) + b)
        if y_ > 0.5:
            y_ = 1
        else:
            y_ = -1
        G_w += (y[i] - y_) * X[i].data * (-1)
        G_b += y_ - y[i]
    G_w = G_w / len(sample_indices)
    G_b = G_b / len(sample_indices)
```

4. Update parameters

NAG:

```python
# 更新参数
v_w = Gamma * v_w + learning_rate * G_w
v_b = Gamma * v_b + learning_rate * G_b
w = w - v_w
b = b - v_b
```

RMSProp:

```python
# 更新参数
Gt_w = Gt_w * Gamma + (1 - Gamma) * np.multiply(G_w, G_w)
Gt_b = Gt_b * Gamma + (1 - Gamma) * np.multiply(G_b, G_b)

w = w - np.multiply(learning_rate * pow(Gt_w + epsilon, -1/2), G_w)
b = b - np.multiply(learning_rate * pow(Gt_b + epsilon, -1/2), G_b)
```

AdaDelta:

```python
# 更新参数
Gt_w = Gt_w * Gamma + (1 - Gamma) * np.multiply(G_w, G_w)
Gt_b = Gt_b * Gamma + (1 - Gamma) * np.multiply(G_b, G_b)
step_w = np.multiply((-1) * pow(delta_w + epsilon, 1/2) * pow(Gt_w + epsilon, -1/2), G_w)
step_b = np.multiply((-1) * pow(delta_b + epsilon, 1 / 2) * pow(Gt_b + epsilon, -1 / 2), G_b)
w = w + step_w
b = b + step_b
delta_w = Gamma * delta_w + np.multiply((1 - Gamma) * step_w, step_w)
delta_b = Gamma * delta_b + np.multiply((1 - Gamma) * step_b, step_b)
```

Adam:

```python
# 更新参数
mt_w = mt_w * Beta + (1 - Beta) * G_w
mt_b = mt_b * Beta + (1 - Beta) * G_b
Gt_w = Gt_w * Gamma + (1 - Gamma) * np.multiply(G_w, G_w)
Gt_b = Gt_b * Gamma + (1 - Gamma) * np.multiply(G_b, G_b)
alpha = learning_rate * pow(1 - pow(Gamma, t), 1/2) * pow(1 - pow(Beta, t), -1/2)
w = w - alpha * mt_w * pow(Gt_w + epsilon, -1/2)
b = b - alpha * mt_b * pow(Gt_b + epsilon, -1/2)
```

5. Validation

Predict under validation set and get the different optimized method loss

NAG:

```python
# 初始化参数
w = np.random.normal(size=123)
b = np.random.normal(size=1)
v = np.zeros(124)

# 记录验证集的loss随迭代次数的值
val_losses = []

for num in range(epoch):
    # 通过SGD更新参数，随机选择10个样本
    sample_indices = random.sample(range(X_test.shape[0]), 100)

    # 获得loss
    val_loss = get_loss(X_test, y_test, w, b)

    w, b, v = NAG(X_test, y_test, w, b, v, learning_rate, sample_indices)

    # 将loss加入列表
    val_losses.append(val_loss)
```

RMSProp:

```python
for num in range(epoch):
    # 通过SGD更新参数，随机选择10个样本
    sample_indices = random.sample(range(X_test.shape[0]), 100)

    # 获得loss
    val_loss = get_loss(X_test, y_test, w, b)

    w, b, Gt = RMSProp(X_test, y_test, w, b, Gt, learning_rate, sample_indices)

    # 将loss加入列表
    val_losses.append(val_loss)
```

AdaDelta:

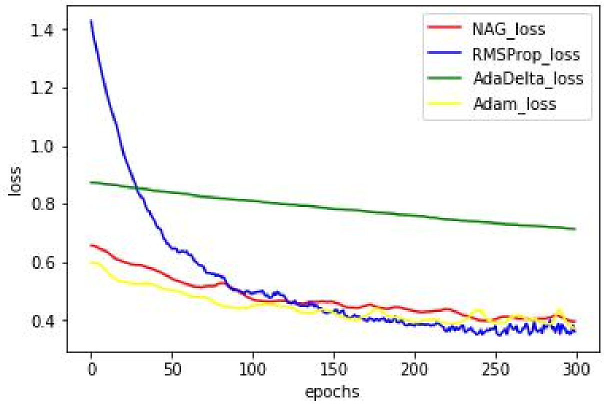
```python
for num in range(epoch):
    # 通过SGD更新参数，随机选择10个样本
    sample_indices = random.sample(range(X_test.shape[0]), 100)

    # 获得loss
    val_loss = get_loss(X_test, y_test, w, b)

    w, b, Gt, delta = AdaDelta(X_test, y_test, w, b, Gt, delta, sample_indices)

    # 将loss加入列表
    val_losses.append(val_loss)
```

Adam:

```python
for num in range(epoch):

    # 通过SGD更新参数，随机选择10个样本
    sample_indices = random.sample(range(X_test.shape[0]), 100)
    # 获得loss
    val_loss = get_loss(X_test, y_test, w, b)

    w, b, Gt, mt = Adam(X_test, y_test, w, b, Gt, mt, sample_indices, learning_rate, num + 1)

    # 将loss加入列表
    val_losses.append(val_loss)
```

6. Draw the graph

Epoch = 300   learning_rate = 0.01



2. ClassificationExperiment

1. Initialization

Initalize logistic regression model parameters with zeros

```
# 加载数据
X_train, y_train = get_data(r"E:\machine learning\lab2\a9a")
X_test, y_test = get_data(r"E:\machine learning\lab2\a9a.t")

# 将验证集的122个特征末尾补0
zeros = np.zeros(16281)
X_test = np.c_[X_test, zeros]

# 初始化参数
learning_rate = 0.01
epoch = 300
```

2. Get Loss

Select the appropriate threshold, mark the sample whose predict scores **greater than the threshold as positive, on the contrary as negative**.

```
def get_loss(X, y, w, b):
    loss = 0
    for i in range(X.shape[0]):
        y_ = sigmoid(np.dot(X[i].data, w) + b)
        if y_ > 0.5:
            y_ = 1
        else:
            y_ = -1
        loss += max(0, 1 - y[i] * y_)
    return loss / X.shape[0] + np.sum(w ** 2) / 2
```

3. Compute gradient

```
def compute_gradient(X, y, w, b, sample_indices):
    # loss_function 对w, b的偏导
    G_w = 0
    G_b = 0
    for i in sample_indices:
        y_ = sigmoid(np.dot(X[i].data, w) + b)
        if y_ > 0.5:
            y_ = 1
        else:
            y_ = -1
        if 1 - y[i] * y_ >= 0:
            G_w += y[i] * X[i].data * (-1)
            G_b += y[i] + (-1)
        else:
            G_w += 0
            G_b += 0
    G_w = G_w / len(sample_indices) + w
    G_b = G_b / len(sample_indices)
```

4. Update parameters

NAG:

```
# 更新参数
v_w = Gamma * v_w + learning_rate * G_w
v_b = Gamma * v_b + learning_rate * G_b
w = w - v_w
b = b - v_b
```

RMSProp:

```
# 更新参数
Gt_w = Gt_w * Gamma + (1 - Gamma) * np.multiply(G_w, G_w)
Gt_b = Gt_b * Gamma + (1 - Gamma) * np.multiply(G_b, G_b)

w = w - np.multiply(learning_rate * pow(Gt_w + epsilon, -1/2), G_w)
b = b - np.multiply(learning_rate * pow(Gt_b + epsilon, -1/2), G_b)
```

AdaDelta:

```
# 更新参数
Gt_w = Gt_w * Gamma + (1 - Gamma) * np.multiply(G_w, G_w)
Gt_b = Gt_b * Gamma + (1 - Gamma) * np.multiply(G_b, G_b)
step_w = np.multiply((-1) * pow(delta_w + epsilon, 1/2) * pow(Gt_w + epsilon, -1/2), G_w)
step_b = np.multiply((-1) * pow(delta_b + epsilon, 1 / 2) * pow(Gt_b + epsilon, -1 / 2), G_b)
w = w + step_w
b = b + step_b
delta_w = Gamma * delta_w + np.multiply((1 - Gamma) * step_w, step_w)
delta_b = Gamma * delta_b + np.multiply((1 - Gamma) * step_b, step_b)
```

Adam:

```
# 更新参数
mt_w = mt_w * Beta + (1 - Beta) * G_w
mt_b = mt_b * Beta + (1 - Beta) * G_b
Gt_w = Gt_w * Gamma + (1 - Gamma) * np.multiply(G_w, G_w)
Gt_b = Gt_b * Gamma + (1 - Gamma) * np.multiply(G_b, G_b)
alpha = learning_rate * pow(1 - pow(Gamma, t), 1/2) * pow(1 - pow(Beta, t), -1/2)
w = w - alpha * mt_w * pow(Gt_w + epsilon, -1/2)
b = b - alpha * mt_b * pow(Gt_b + epsilon, -1/2)
```

5. Validation

Predict under validation set and get the different optimized method loss

NAG:

```
# 初始化参数
w = np.random.normal(size=123)
b = np.random.normal(size=1)
v = np.zeros(124)

# 记录验证集的loss随迭代次数的值
val_losses = []

for num in range(epoch):
    # 通过SGD更新参数，随机选择10个样本
    sample_indices = random.sample(range(X_test.shape[0]), 100)

    # 获得loss
    val_loss = get_loss(X_test, y_test, w, b)

    w, b, v = NAG(X_test, y_test, w, b, v, learning_rate, sample_indices)

    # 将loss加入列表
    val_losses.append(val_loss)
```

RMSProp:

```
for num in range(epoch):
    # 通过SGD更新参数，随机选择10个样本
    sample_indices = random.sample(range(X_test.shape[0]), 100)

    # 获得loss
    val_loss = get_loss(X_test, y_test, w, b)

    w, b, Gt = RMSProp(X_test, y_test, w, b, Gt, learning_rate, sample_indic

    # 将loss加入列表
    val_losses.append(val_loss)
```

AdaDelta:

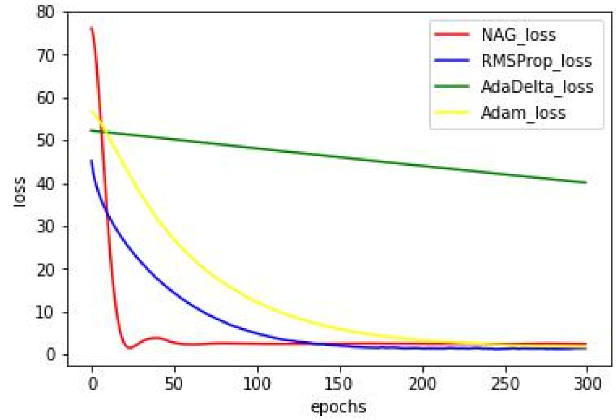
```
for num in range(epoch):
    # 通过SGD更新参数，随机选择10个样本
    sample_indices = random.sample(range(X_test.shape[0]), 100)

    # 获得loss
    val_loss = get_loss(X_test, y_test, w, b)

    w, b, Gt, delta = AdaDelta(X_test, y_test, w, b, Gt, delta, sample_indice

    # 将loss加入列表
    val_losses.append(val_loss)
```

Adam:

```
for num in range(epoch):

    # 通过SGD更新参数，随机选择10个样本
    sample_indices = random.sample(range(X_test.shape[0]), 100)
    # 获得loss
    val_loss = get_loss(X_test, y_test, w, b)

    w, b, Gt, mt = Adam(X_test, y_test, w, b, Gt, mt, sample_indices, learning_rate, num +

    # 将loss加入列表
    val_losses.append(val_loss)
```

6. Draw the graph

Epoch = 300 learning_rate = 0.01



IV. CONCLUSION

1. Better understand the difference between gradient descent and stochastic gradient descent.
2. Understand the principles of SVM and be able to practice on larger dataset.
3. Understand the influence of difference verification methods on the convergence of loss function and the difference and relationships are preliminarily recognized.
4. ,Can refer to different optimization methods and choose the appropriate method according to the specific situation in future projects