Linear Methods

Lesson Objectives

- After completing this lesson, you should be able to:
  - –Understand the Pipelines API for Logistic Regression and Linear Least Squares
  - –Perform classification with Logistic Regression
  - –Perform regression with Linear Least Squares
  - –Use regularization with Logistic Regression and Linear Least Squares

Linear Methods

- Logistic Regression
- Linear Least Squares

Logistic Regression

- Widely used to predict binary responses (classification)
- Can be generalized into multinomial logistic regression (multiclass)
  - for K possible outcomes, choose one outcome as "pivot"
  - the other K-1 outcomes can be separately regressed against the pivot outcome

Logistic regression advantages

- Has no tuning parameters
- Its prediction equation is simple and easy to implement

MLlib's implementation

- MLlib chooses first class (0) as the "pivot" class
- For multiclass classification problem, outputs a multinomial logistic regression model, containing K-1 binary logistic regression models regressed against the "pivot" class
- For a new data point, K-1 models are run and the class with largest probability is chosen as the predicted class
- Supported algorithms:
  - mini-batch gradient descent
  - L-BFGS (recommended for faster convergence)

Regularization

–L2 regularization: Ridge Regression (penalizes beta parameters by the square of their magnitude)
–L1 regularization: Lasso (penalizes beta parameters by their absolute value)
–Elastic net regularization combines L1 and L2, with a weight for each

- equivalent to ridge regression (L2) if alfa set to 0
- equivalent to Lasso (L1) if alfa set to 1

Elastic net regularization: parameters

- elasticNetParam corresponds to alfa
- regParam corresponds to lambda

	regularizer $R(\mathbf{w})$	gradient or sub-gradient
zero (unregularized)	0	$\mathbf{0}$
L2	$\frac{1}{2}\|\|\mathbf{w}\|\|_2^2$	$\mathbf{w}$
L1	$\|\|\mathbf{w}\|\|_1$	$\mathrm{sign}(\mathbf{w})$
elastic net	$\alpha\|\|\mathbf{w}\|\|_1 + (1-\alpha)\frac{1}{2}\|\|\mathbf{w}\|\|_2^2$	$\alpha\mathrm{sign}(\mathbf{w}) + (1-\alpha)\mathbf{w}$

Example of Logistic Regression (1)

```
from pyspark.mllib.classification import LogisticRegressionWithLBFGS, LogisticRegressionModel
from pyspark.mllib.regression import LabeledPoint

!wget https://raw.githubusercontent.com/apache/spark/master/data/mllib/sample_svm_data.txt
```

```
--2016-09-26 08:55:31--  https://raw.githubusercontent.com/apache/spark/master/data/mllib/sample_svm_data.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.12.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.12.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 39474 (39K) [text/plain]
Saving to: ‘sample_svm_data.txt.1’

100%[======================================>] 39.474      --.-K/s   in 0,06s

2016-09-26 08:55:31 (670 KB/s) - ‘sample_svm_data.txt.1’ saved [39474/39474]
```

Example of Logistic Regression (2)

```
def parsePoint(line):
    values = [float(x) for x in line.split(' ')]
    return LabeledPoint(values[0], values[1:])

data = sc.textFile("sample_svm_data.txt")

parsedData = data.map(parsePoint)
```

```
parsedData.take(1)
```

```
[LabeledPoint(1.0, [0.0,2.52078447202,0.0,0.0,0.0,2.00468443649,2.00034729927,0.0,2.22838704274,2.22838704274,0.0,0.0,0.0
,0.0,0.0,0.0])]
```

Example of Logistic Regression (3)

```
model = LogisticRegressionWithLBFGS.train(parsedData)

labelsAndPreds = parsedData.map(lambda p: (p.label, model.predict(p.features)))

trainErr = labelsAndPreds.filter(lambda (v, p): v != p).count() / float(parsedData.count())
print("Training Error = " + str(trainErr))
```

```
Training Error = 0.366459627329
```

Linear Least Squares

- Most common formulation for regression problems
- As in logistic regression, different types of regularization are possible:
  - no regularization: Ordinary Least Squares
  - L2 regularization: Ridge Regression
  - L1 regularization: Lasso
  - Elastic net
- Average loss = Mean Squared Error

Example of Linear Regression (1)

```
from pyspark.mllib.regression import LinearRegressionWithSGD, LinearRegressionModel

!wget https://raw.githubusercontent.com/apache/spark/master/data/mllib/ridge-data/lpsa.data
```

```
--2016-09-26 08:58:53--  https://raw.githubusercontent.com/apache/spark/master/data/mllib/ridge-data/lpsa.data
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.12.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.12.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10395 (10K) [text/plain]
Saving to: ‘lpsa.data’

100%[======================================>] 10.395      --.-K/s   in 0,001s

2016-09-26 08:58:54 (11,1 MB/s) - ‘lpsa.data’ saved [10395/10395]
```

Example of Linear Regression (2)

```
def parsePoint(line):
    values = [float(x) for x in line.replace(',', ' ').split(' ')]
    return LabeledPoint(values[0], values[1:])

data = sc.textFile("lpsa.data")
parsedData = data.map(parsePoint)
```

```
parsedData.take(1)
```

```
[LabeledPoint(-0.4307829, [-1.63735562648,-2.00621178481,-1.86242597251,-1.02470580167,-0.522940888712,-0.863171185426,-1
.04215728919,-0.864466507337])]
```

Example of Linear Regression (3)

```
model = LinearRegressionWithSGD.train(parsedData, iterations=100, step=0.00000001)

valuesAndPreds = parsedData.map(lambda p: (p.label, model.predict(p.features)))

MSE = valuesAndPreds.map(lambda (v, p): (v - p)**2).reduce(lambda x, y: x + y) / valuesAndPreds.count()

print("Mean Squared Error = " + str(MSE))
```

```
Mean Squared Error = 7.4510328101
```

Lesson Summary

- Having completed this lesson, you should be able to:
  - Understand the Pipelines API for Logistic Regression and Linear Least Squares
  - Perform classification with Logistic Regression
  - Perform regression with Linear Least Squares
  - Use regularization with Logistic Regression and Linear Least Squares