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.ml.classification import LogisticRegression
from pyspark.ml.classification import BinaryLogisticRegressionSummary

logr = LogisticRegression().setMaxIter(10).setRegParam(0.3).setElasticNetParam(0.8)

logrModel = logr.fit(trainingData)

print "Weights: %s Intercept: %s" % (logrModel.coefficients, logrModel.intercept)
```

```
Weights: (692,[235,243,244,262,263,272,290,291,300,328,350,351,356,378,379,405,406,407,428,433,434,456,461,462,483,484,489,490,496,511,512,517,539,540,568,604],[-6.56411506086e-05,-3.02912249122e-05,-8.30901849188e-05,-0.000169251909514,-0.000524781223777,-0.000190296425334,-0.000154247405039,-3.92323632965e-05,-0.000205585788632,-3.5389983131e-06,3.89081604026e-05,0.000156272498086,-3.93785311006e-05,0.00072160556785,0.000174349623502,2.58332002002e-05,0.000758596606147,0.000166653600909,-0.000137824099317,0.000387347267355,0.000731960915983,-0.000177992085607,0.000359139681642,0.000377820946662,-9.63539844782e-05,-0.000194382864912,0.000362079145346,0.000161232856735,-0.000112267465886,-0.000224035146181,-0.000370106361044,0.000177813363319,-0.0001999803652,-0.00123042560777,-0.00104268903147,-0.000125273008982]) Intercept: 0.190781165145
```

Example of Logistic Regression (2)

```
trainingSummaryLR = logrModel.summary
objectiveHistoryLR = trainingSummaryLR.objectiveHistory
print objectiveHistoryLR
```

```
[0.6853142072764583, 0.665833889504604, 0.6193324313245665, 0.6100497002983996, 0.6012654439909555, 0.5980155978378509, 0
.593018565828796, 0.5912149048437889, 0.5890264180252197, 0.5839790063991089, 0.5816694167353569]
```

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.ml.regression import LinearRegression

lr = LinearRegression().setMaxIter(10).setRegParam(0.3).setElasticNetParam(0.8)

lrModel = lr.fit(trainingData)

print "Weights: %s Intercept: %s" % (lrModel.coefficients, lrModel.intercept)
```

```
Weights: (692,[235,244,262,263,272,290,291,300,323,328,345,350,351,356,377,378,379,405,406,407,428,433,434,435,456,461,462,483,484,489,490,496,511,512,517,539,540,568,578,604,605],[-3.37294183548e-06,-2.85244277006e-06,-1.7895386243e-05,-3.35730562031e-05,-2.86851170957e-05,-7.82569603452e-06,-3.43031924309e-06,-3.13277870877e-05,3.83855168802e-05,-2.64129159353e-06,-3.87300894998e-06,0.000109143114411,0.000117518437324,-2.97543826502e-06,1.70734396925e-06,0.000164293449113,0.000132971382044,6.24626227608e-05,0.000169937553514,0.000124783682503,-7.08917640918e-06,0.000160720013148,0.000164912164173,2.98893305773e-05,-2.25913691967e-05,0.000149855136869,0.000156764349391,-5.35710471605e-06,-3.21093069039e-05,0.000150850588663,0.000121021992713,-6.13051732871e-06,-3.41317040068e-05,-3.61166277946e-05,0.00013598687393,-2.32499957229e-05,-0.000112233119321,-5.75040024342e-05,-1.35856382097e-06,-4.6523600166e-06,-1.61354575661e-06]) Intercept: 0.343093051246
```

Example of Linear Regression (2)

```
trainingSummaryLLS = lrModel.summary

print "numIterations: %s" % trainingSummaryLLS.totalIterations

print "objectiveHistory: %s" % trainingSummaryLLS.objectiveHistory
```

```
numIterations: 11
objectiveHistory: [0.4921875, 0.4571962454835806, 0.42721256226410376, 0.4197574233133219, 0.4079194334154008, 0.39875369
94705158, 0.3949553498795431, 0.3932725644917028, 0.3915797961054517, 0.38774075744160424, 0.38679591298651783]
```

Example of Linear Regression (2)

```
trainingSummaryLLS.residuals.show(5)
```

```
+--------------------+
|           residuals|
+--------------------+
|-0.23933577912497733|
| -0.2785526569681104|
|-0.24240274742471352|
| 0.17047660785300278|
| 0.21566169880410468|
+--------------------+
only showing top 5 rows
```

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