Support Vector Machines

Kazi Aminul Islam
Department of Computer Science
Kennesaw State University

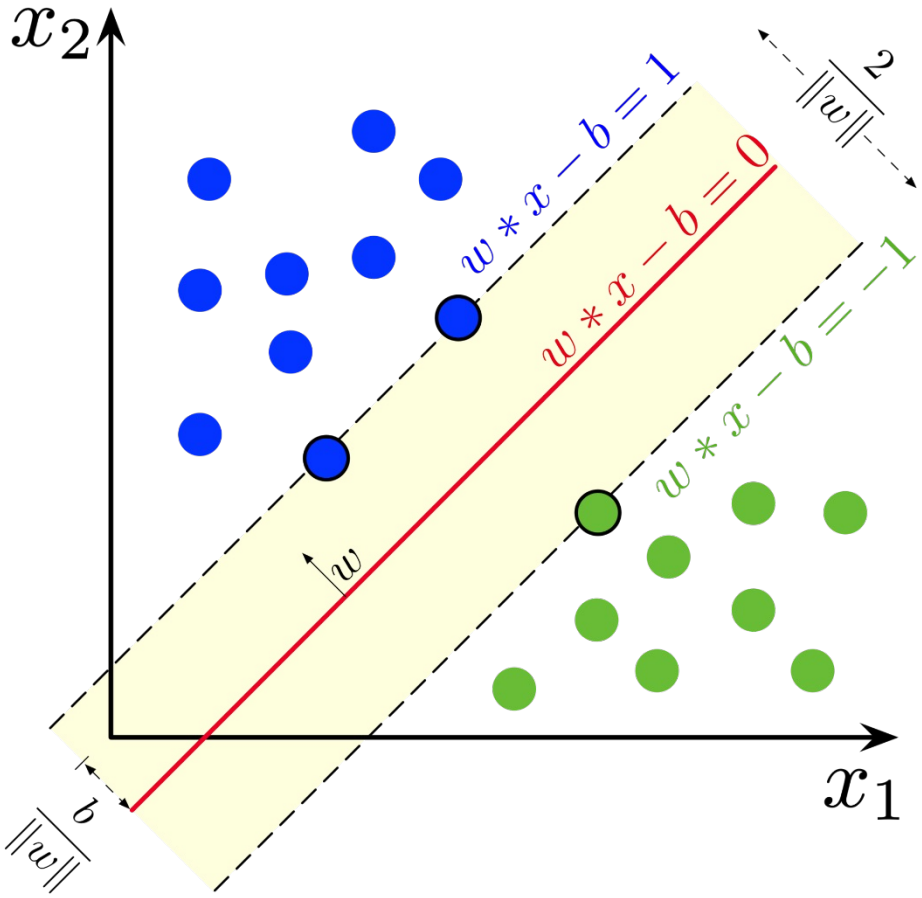
Maximizing the Margin

- We want a classifier with as big a margin as possible.
- The margin is $\frac{2}{||w||}$.
- To maximize the margin, we need to minimize $||w||$ with the condition that there are no data points in the margin.

$$w^T x_i - b \geq 1, \forall y_i = 1$$

$$w^T x_i - b \leq -1, \forall y_i = -1$$

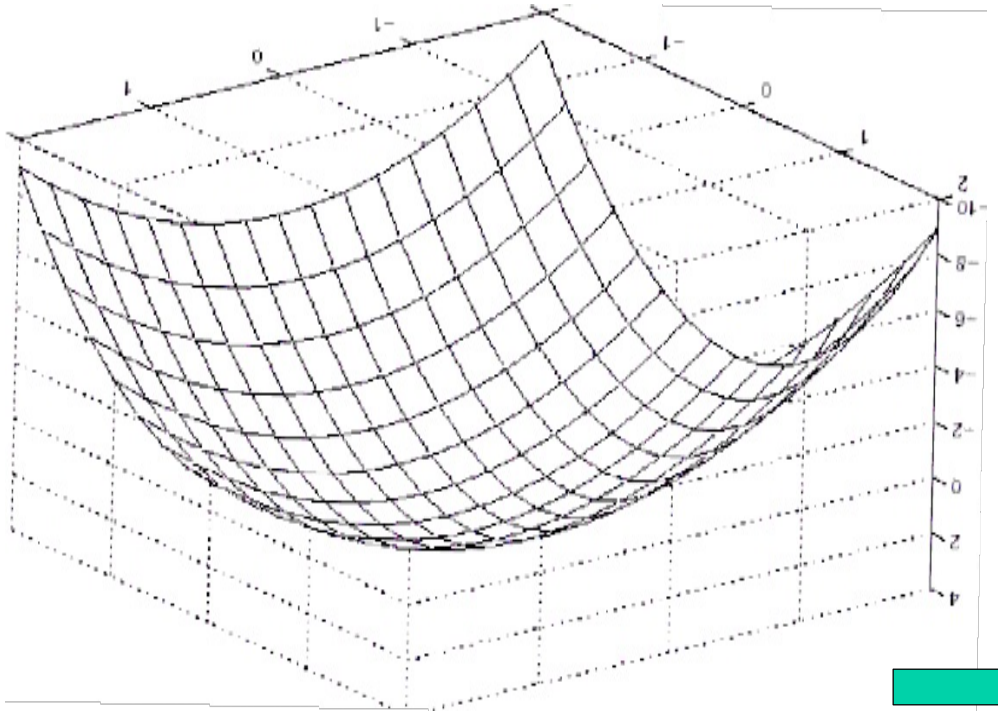
- The condition can be merged to $y_i(w^T x_i - b) \geq 1$
- $y_i(w^T x_i - b) - 1 \geq 0$



Solving a Quadratic Programming Problem

- $\min f(w) = \frac{1}{2}||w||$ subject to $y_i(w^T x_i - b) - 1 = 0, \forall i$
- This is a constrained optimization problem.
- It can be solved by the Lagrangian multiplier method.
- The surface of the quadratic function is a paraboloid with a single global minimum (unlike neural network).

Example: paraboloid $2x^2+2y^2$ s.t. $x+y=1$

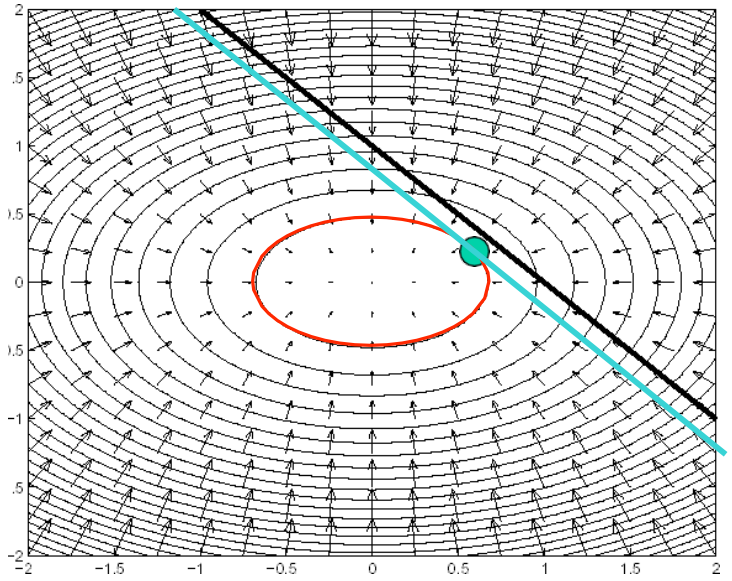


- Intuition: find intersection of two functions *f, g* at a tangent point (intersection = both constraints satisfied; tangent = derivative is 0); this will be a min (or max) for *f* s.t. the constraint *g* is satisfied

Flattened paraboloid f: $2x^2+2y^2=0$ *with* superimposed constraint g: $x+y=1$

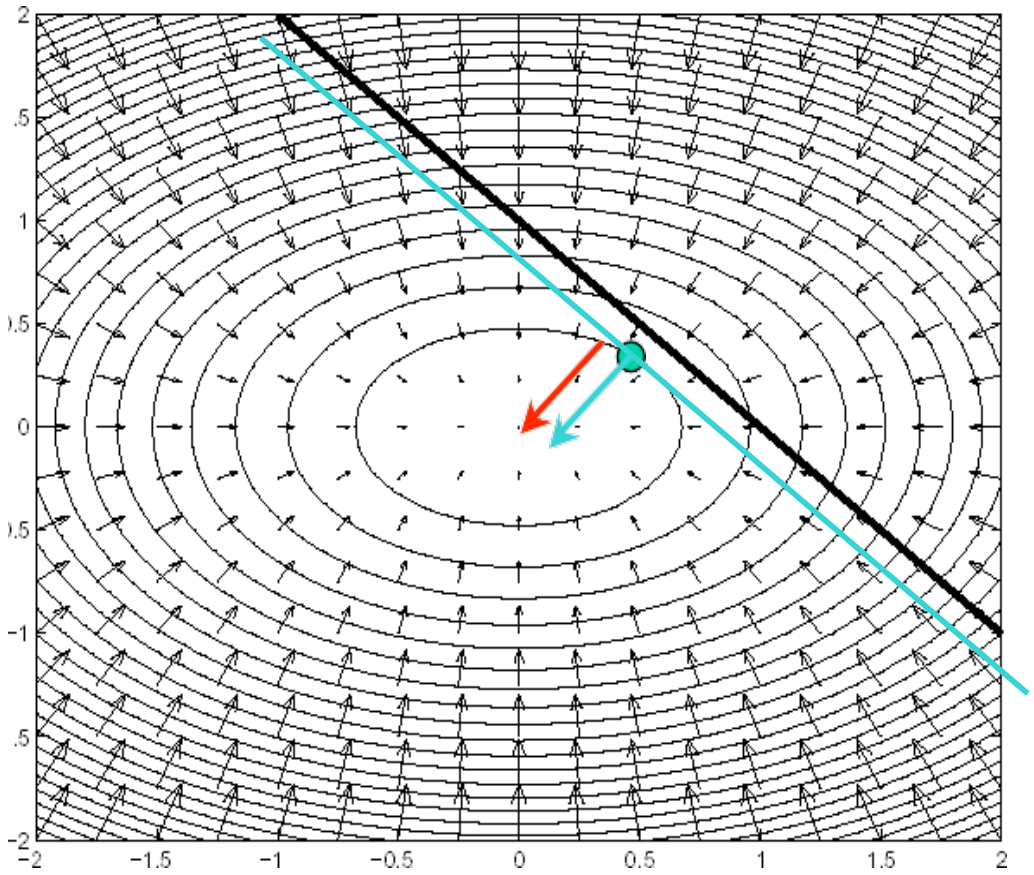
- *Minimize/Maximize* when the constraint line g (shown in cyan)

is tangent to the inner ellipse contour line of f (shown in red) – note direction of gradient arrows.



Parallel Gradient Vectors

- *Minimize/Maximize* when the constraint line g is tangent to the inner ellipse contour line of f
- At tangent solution p, gradient vectors of f, g are parallel.



Lagrangian Form for SVM

- $f(x) = 1/2\left|\left|w\right|\right|^2$ (mathematic convenience)
- $g(x_i) = -y_i(w^T x_i - b) + 1 \leq 0$
- The Lagrangian is
- $\min L(w, b) = 1/2\left|\left|w\right|\right|^2 - \sigma a_i[y_i(w^T x_i - b) - 1]$
- $\Rightarrow \min L(w, b) = 1/2\left|\left|w\right|\right|^2 - \sigma a_i[y_i(w^T x_i - b)] + \sigma a_i$
- Under the KKT dual complementarity condition, we have $\alpha_i^* > 0$ only for data points on the edge (support vectors).
- Here we don't have $\beta_i' s$ Lagrange multipliers because SVM has only inequality constraints.

Solve SVM Lagrangian Formula

- $\min L(w,b) = 1/2\left|\left|w\right|\right|^2 - \sigma_{i=1}^{l} a_i y_i (w^T x_i - b) + \sigma_{i=1}^{l} a_i \;\; s.t. \forall i, a_i \geq 0$, where l is # of training points.
- Partial derivatives w.r.t w and b:
- $\frac{\partial L}{\partial w} = w - \sigma_{i=1}^{l} a_i y_i \, x_i = 0$, note: w, x_i are vectors.
- $\frac{\partial L}{\partial b} = \sigma_{i=1}^{l} a_i y_i = 0$
- Thus, $w = \sigma_{i=1}^{l} a_i y_i \, x_i$ and $\sigma_{i=1}^{l} a_i y_i = 0$

Remove w and b

- To $\min L(w,b) = 1/2\left|\left|w\right|\right|^2 - \sigma_{i=1}^{l} a_i y_i (w^T x_i - b) + \sigma_{i=1}^{l} a_i$
- Replace $w = \sigma_{i=1}^{l} a_i y_i x_i$, and use the fact $\sigma_{i=1}^{l} a_i y_i = 0$, we have
- $\min L(w,b) = 1/2\left|\left|w\right|\right|^2 - \sigma_{i=1}^{l} a_i y_i (w^T x_i) + b\,\sigma_{i=1}^{l} a_i y_i + \sigma_{i=1}^{l} a_i$
- $\Rightarrow \frac{1}{2}(\sigma_i a_i y_i x_i)(\sigma_j a_j y_j x_j) - \sigma_i a_i y_i (\sigma_j a_j y_j x_j^T x_i) + \sigma_i a_i$
- $\Rightarrow \frac{1}{2}\sigma_i \sigma_j a_i a_j y_i y_j x_i^T x_j - \sigma_i \sigma_j a_i a_j y_i y_j x_i^T x_j + \sigma_i a_i$
- $\Rightarrow \sigma_i a_i - \frac{1}{2}\sigma_i \sigma_j a_i a_j y_i y_j x_i^T x_j$
- The minimum depends on the dot product of samples.
- Most of $a_i's$ are zeros except support vectors.

Solving Dual SVM – Linearly Separable

- The dual problem:
  $\max_{\alpha} \sigma_i \alpha_i - \frac{1}{2} \sigma_i \sigma_j \alpha_i \alpha_j y_i y_j x_i^T x_j$ subject to $\sigma_i \alpha_i y_i = 0$ and $\alpha_i \geq 0$.
- It is quadratic programming problem and can be solved.
- $\frac{\partial}{\partial \alpha_i} = 1 - \frac{1}{2}\left(2 \sigma_{j \neq i} \alpha_j y_i y_j x_i^T x_j + 4\alpha_i y_i^2 \left|\left|x_i\right|\right|^2\right) = 0$
- If we have k support vectors, only those non-zero $\alpha^{'}s$ require to be computed using k equations.
- We find the best solution by left out one equation for $\sigma_i \alpha_i y_i = 0$.
- Thus, $\alpha^{*'}s$ are solved.

Solve w and b

- Once we solve $\alpha^{*'}s$, w can be solve by $w = \sigma_{i=1}^{l} a_i y_i x_i$.
- Again, only few support vectors are involved in the computation.
- How about b? Recall that the set $w^T x_i - b = 1$ for positive samples and $w^T x_i - b = -1$ for negative samples on the gutter, i.e., support vectors.
- So $y_i(w^T x_i - b) = 1, \forall \alpha_i > 0$. So $w^T x_i - b = \frac{1}{y_i} = y_i$
- $b = w^T x_i - \mathrm{y_i}$ for any i such $that\ \alpha_i > 0$.

Transformation to Higher Dimensions

- For data that are not linearly separable, map them to a higher dimensional space.
- Let the map function be $\phi(x)$. To minimize $L(w, b)$, we need to compute $\phi(x_i)^T \phi(x_j)$ and to predict, we need to compute $\phi(x_i)^T \phi(u)$.
- If we can find a function $\mathrm{K}(x_i, x_j) = \phi(x_i)^T \phi(x_j)$, we may skip the transformation and compute the dot product directly. This is so called kernel trick.

Kernel Functions

- Polynomial $\mathrm{K}(\mathrm{u}, \mathrm{v}) = (uv + r)^n$
- Gaussian/Radial Kernels $\mathrm{K}(\mathrm{u}, \mathrm{v}) = e^{-\frac{||u-v||^2}{2\sigma^2}}$
- Sigmoid $\mathrm{K}(\mathrm{u}, \mathrm{v}) = \tanh(\eta uv + \tau)$

Polynomial Kernel

- In the quadratic transformation example, $\mathrm{x} \rightarrow \mathrm{x}^2$.
- Given two data points a, and b, the dot product after mapping is $(a, a^2)(b, b^2) = ab + a^2b^2 \Rightarrow (ab + 1/2)^2$ by choosing n=2 and r=1/2 in the polynomial kernel form.
- $K(a, b) = (ab + 1/2)^2 = \mathrm{a}^2\mathrm{b}^2 + \mathrm{ab} + \frac{1}{4} = \left(\mathrm{a}, \mathrm{a}^2, \frac{1}{2}\right)\left(\mathrm{a}, \mathrm{b}^2, \frac{1}{2}\right) = \phi(a)\phi(b)$
- By computing the kernel (with ¼ constant, that does not affect optimization calculations), the mapping raises one dimensional data to 3 dimensional space.

Linear Support Vector Machine (SVM)

- Linear SVM is a standard method for large-scale classification tasks with the following hinge loss:
- $L(w; x, y) = \max\{0, 1 - yw^T x\}$
- By default, linear SVMs are trained with an L2 regularization.
- The linear SVMs algorithm outputs an SVM model. Given a new data point, the model makes a prediction based on the value of $w^T x$. If $w^T x \geq 0$ then the outcome is positive, and negative otherwise.

Scala Code

```
import org.apache.spark.mllib.classification.{SVMModel,
SVMWithSGD}

import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics

import org.apache.spark.mllib.util.MLUtils

// Load training data in LIBSVM format.

val data = MLUtils.loadLibSVMFile(sc,
"data/mllib/sample_libsvm_data.txt")

// Split data into training (60%) and test (40%).

val splits = data.randomSplit(Array(0.6, 0.4), seed = 11L)

val training = splits(0).cache()

val test = splits(1)

// Run training algorithm to build the model

val numIterations = 100

val model = SVMWithSGD.train(training, numIterations)

// Clear the default threshold.

model.clearThreshold()

// Compute raw scores on the test set.
val scoreAndLabels = test.map { point =>
val score = model.predict(point.features)
(score, point.label)
}

// Get evaluation metrics.
val metrics = new BinaryClassificationMetrics(scoreAndLabels)
val auROC = metrics.areaUnderROC()

println(s"Area under ROC = $auROC")

// Save and load model
model.save(sc, "target/tmp/scalaSVMWithSGDModel")
val sameModel = SVMModel.load(sc,
"target/tmp/scalaSVMWithSGDModel")
```

Scala Code

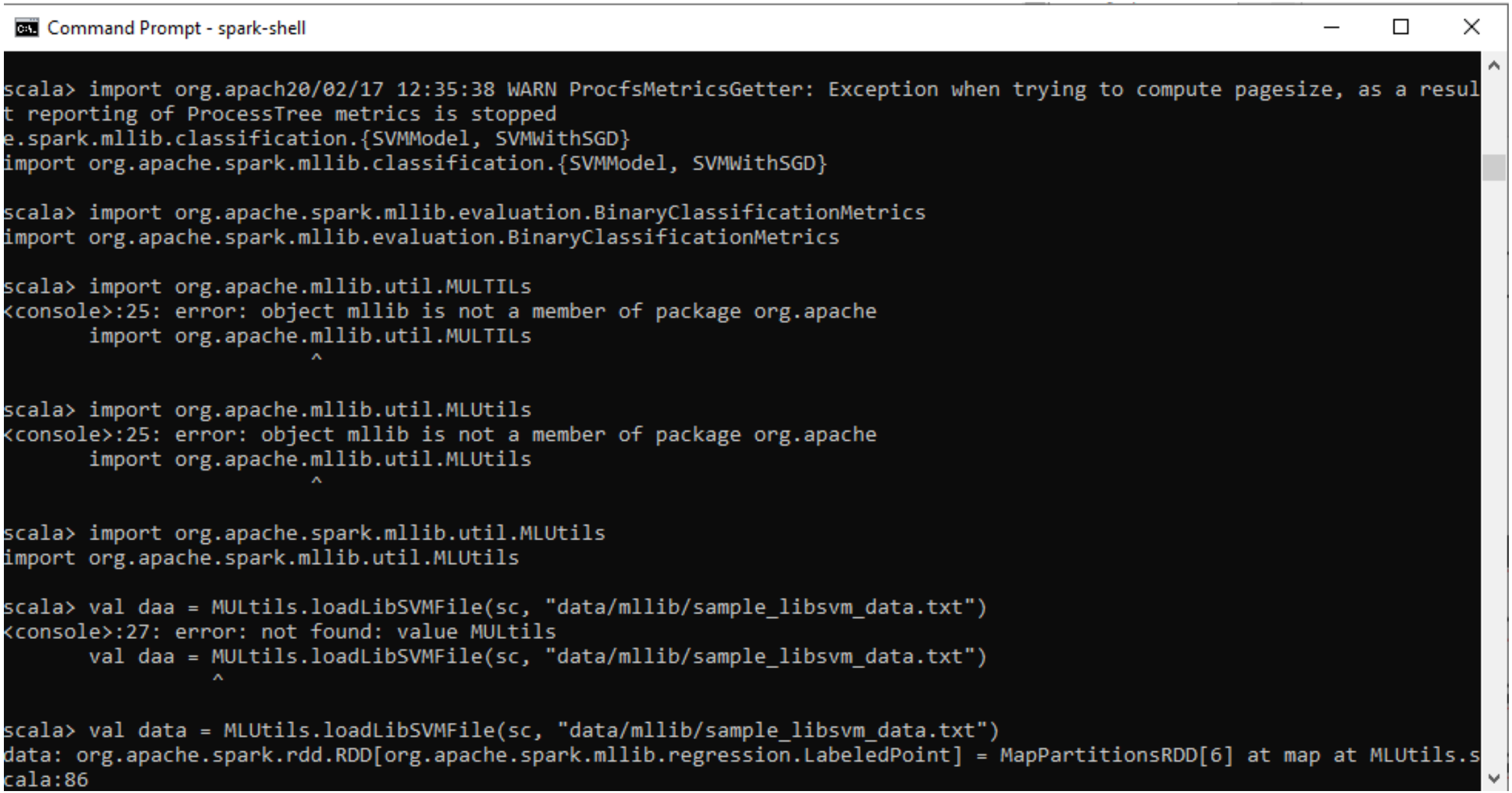
https://spark.apache.org/docs/latest/mllib-linear-methods.html#linear-support-vector-machines-svms

Example

- Find full example code at "examples/src/main/scala/org/apache/spark/examples/mllib/SVMWithSGDExample.scala" in the c:/spark directory.
- The SVMWithSGD.train() method by default performs L2 regularization with the regularization parameter set to 1.0.
- If we want to configure this algorithm, we can customize SVMWithSGD further by creating a new object directly and calling setter methods.
- All other spark.mllib algorithms support customization in this way as well. For example, the following code produces an L1 regularized variant of SVMs with regularization parameter set to 0.1, and runs the training algorithm for 200 iterations.

L1 Regularization

```
import org.apache.spark.mllib.optimization.L1Updater

val svmAlg = new SVMWithSGD()
svmAlg.optimizer
  .setNumIterations(200)
  .setRegParam(0.1)
  .setUpdater(new L1Updater)
val modelL1 = svmAlg.run(training)
```


Command Prompt - spark-shell
scala> import org.apach20/02/17 12:35:38 WARN ProcfsMetricsGetter: Exception when trying to compute pagesize, as a resul
t reporting of ProcessTree metrics is stopped
e.spark.mllib.classification.{SVMModel, SVMWithSGD}
import org.apache.spark.mllib.classification.{SVMModel, SVMWithSGD}
scala> import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
scala> import org.apache.mllib.util.MULTILs
<console>:25: error: object mllib is not a member of package org.apache
import org.apache.mllib.util.MULTILs
^
scala> import org.apache.mllib.util.MLUtils
<console>:25: error: object mllib is not a member of package org.apache
import org.apache.mllib.util.MLUtils
^
scala> import org.apache.spark.mllib.util.MLUtils
import org.apache.spark.mllib.util.MLUtils
scala> val daa = MULtils.loadLibSVMFile(sc, "data/mllib/sample_libsvm_data.txt")
<console>:27: error: not found: value MULtils
val daa = MULtils.loadLibSVMFile(sc, "data/mllib/sample_libsvm_data.txt")
^
scala> val data = MLUtils.loadLibSVMFile(sc, "data/mllib/sample_libsvm_data.txt")
data: org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint] = MapPartitionsRDD[6] at map at MLUtils.s
cala:86

Data Point

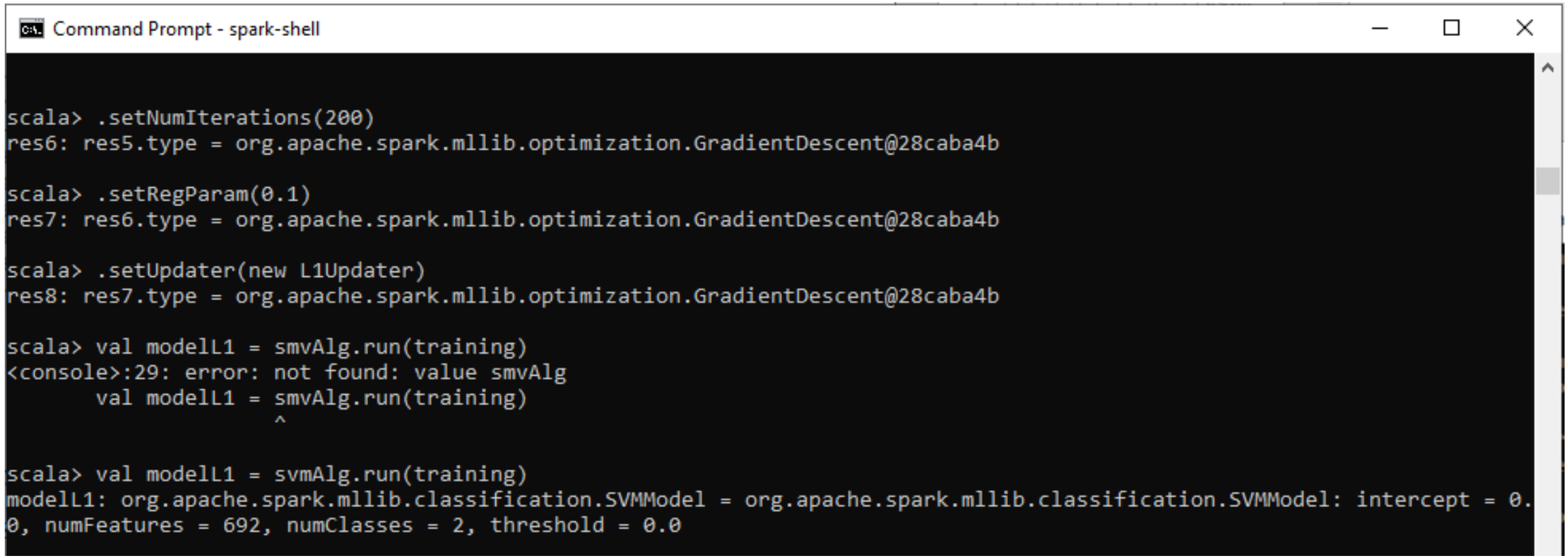
```
scala> data.first
res9: org.apache.spark.mllib.regression.LabeledPoint = (0.0,(692,[127,128,129,130,131,154,155,156,157,158,159,181,182,18
3,184,185,186,187,188,189,207,208,209,210,211,212,213,214,215,216,217,235,236,237,238,239,240,241,242,243,244,245,262,26
3,264,265,266,267,268,269,270,271,272,273,289,290,291,292,293,294,295,296,297,300,301,302,316,317,318,319,320,321,328,32
9,330,343,344,345,346,347,348,349,356,357,358,371,372,373,374,384,385,386,399,400,401,412,413,414,426,427,428,429,440,44
1,442,454,455,456,457,466,467,468,469,470,482,483,484,493,494,495,496,497,510,511,512,520,521,522,523,538,539,540,547,54
8,549,550,566,567,568,569,570,571,572,573,574,575,576,577,578,594,595,596,597,598,599,600,601,602,603,604,622,623,624,62
5,626,627,628,629,630,651,652,653,654,655,656,657],[51.0,159.0,253.0,159.0,50...

scala>
```

Command Prompt - spark-shell

```
scala> val splits = data.randomSplit(Array(0.6, 0.4), seed = 11L)
splits: Array[org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint]] = Array(MapPartitionsRDD[7] at r
andomSplit at <console>:28, MapPartitionsRDD[8] at randomSplit at <console>:28)

scala> val training = splits(0).cache()
training: org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint] = MapPartitionsRDD[7] at randomSplit
at <console>:28

scala> val test = splits(1)
test: org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint] = MapPartitionsRDD[8] at randomSplit at <
console>:28

scala> val numIterations = 100
numIterations: Int = 100

scala> val model = SVMWithSGD.train(training, numIterations)
model: org.apache.spark.mllib.classification.SVMModel = org.apache.spark.mllib.classification.SVMModel: intercept = 0.0,
 numFeatures = 692, numClasses = 2, threshold = 0.0

scala> model.clearThreshold()
res2: model.type = org.apache.spark.mllib.classification.SVMModel: intercept = 0.0, numFeatures = 692, numClasses = 2, t
hreshold = None

scala> val scoreAndLabels = test.map { point =>
     | val score = model.predict(point.features)
     | (score, point.label)
     | }
scoreAndLabels: org.apache.spark.rdd.RDD[(Double, Double)] = MapPartitionsRDD[212] at map at <console>:30
```

Command Prompt - spark-shell

```
scala> val metrics = new BinaryClassificationMetrics(scoreAndLabels)
metrics: org.apache.spark.mllib.evaluation.BinaryClassificationMetrics = org.apache.spark.mllib.evaluation.BinaryClassif
icationMetrics@4a3306c2

scala> val auROC = metrics.areaUnderROC()
auROC: Double = 1.0

scala> println(s"Area under ROC = $auROC")
Area under ROC = 1.0

scala> model.save(sc, "target/tmp/scalaSVMWithSGDModel")

scala> val sameModel = SVMModel.load(sc, "target/tmp/scalaSVMWithSGDModel")
sameModel: org.apache.spark.mllib.classification.SVMModel = org.apache.spark.mllib.classification.SVMModel: intercept =
0.0, numFeatures = 692, numClasses = 2, threshold = None

scala> import org.apache.spark.mllib.optimization.L1Updater
import org.apache.spark.mllib.optimization.L1Updater

scala> val svmAlg = new SVMWithSGD()
svmAlg: org.apache.spark.mllib.classification.SVMWithSGD = org.apache.spark.mllib.classification.SVMWithSGD@3d4de7ec

scala> svmAlg.optimizer
res5: org.apache.spark.mllib.optimization.GradientDescent = org.apache.spark.mllib.optimization.GradientDescent@28caba4b


scala> .setNumIterations(200)
res6: res5.type = org.apache.spark.mllib.optimization.GradientDescent@28caba4b

scala> .setRegParam(0.1)
```


Command Prompt - spark-shell
scala> .setNumIterations(200)
res6: res5.type = org.apache.spark.mllib.optimization.GradientDescent@28caba4b
scala> .setRegParam(0.1)
res7: res6.type = org.apache.spark.mllib.optimization.GradientDescent@28caba4b
scala> .setUpdater(new L1Updater)
res8: res7.type = org.apache.spark.mllib.optimization.GradientDescent@28caba4b
scala> val modelL1 = smvAlg.run(training)
<console>:29: error: not found: value smvAlg
val modelL1 = smvAlg.run(training)
^
scala> val modelL1 = svmAlg.run(training)
modelL1: org.apache.spark.mllib.classification.SVMModel = org.apache.spark.mllib.classification.SVMModel: intercept = 0.
0, numFeatures = 692, numClasses = 2, threshold = 0.0

References

- Support Vector Machine Lagrange Multipliers and Simplex Volume Decompositions, http://www-math.mit.edu/~edelman/publications/support_vector.pdf
- Spark 3.0.1 Documents at https://spark.apache.org/docs/latest/mllib-linear-methods.html#regression