Transduction

Outline

- Introduction
- History
- Theoretical Details
- Algorithm Walkthrough
- Implementation Results
- Strengths and Weaknesses
- Bells and Whistles
- Current Trajectory
- Conclusion

Introduction --- The Problem Setting

Given:

- Labeled set
- Unlabeled set
- Labels \in {-1, 1}

Find set of labels that best fits unlabeled set

T. Joachims. “Transductive inference for text classification using support vector machines.” ICML, 1999.

Introduction --- The Problem Setting

Given:

- Labeled set
- Unlabeled set
- Labels \in {-1, 1}

Find set of labels that best fits unlabeled set

Note that we aren't extending to unseen examples!

T. Joachims. "Transductive inference for text classification using support vector machines." ICML, 1999.

Introduction --- Why we might care

Given:

- Labeled set
- Unlabeled set
- Labels \in {-1, 1}

Find set of labels that best fits unlabeled set

Note that we aren't extending to unseen examples!

Theoretically:

- For problems with partially labeled data
- When we don't need a general function

Introduction --- Why we might care

Concretely:

- Text classification (Joachims, 1999)
- Sentiment analysis (Talukdar and Crammer, 2009)
- Image classification (Yu et al., 2012)

Theoretically:

- For problems with partially labeled data
- When we don't need a general function

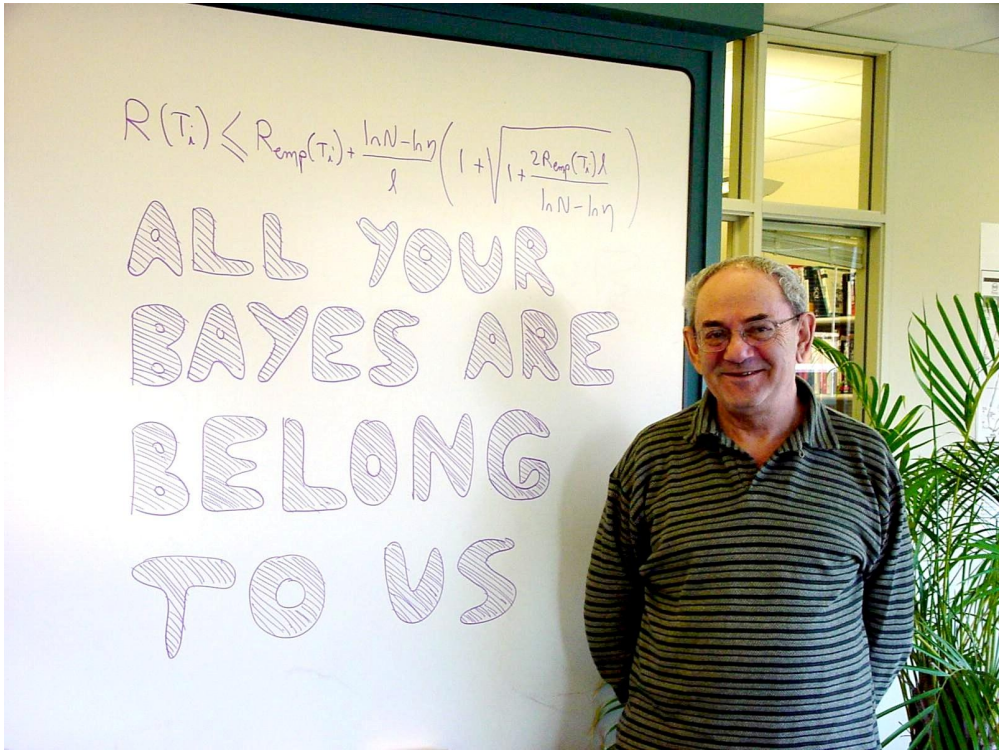
History

1963 --- Maximum-margin hyperplane algorithm by Vapnik

1971 --- Vapnik-Chervonenkis dimension

1992 --- Kernel trick by Boser, Guyon, and Vapnik

1998 --- Transduction by Vapnik



http://www.lecun.org/gallery/libpro/20011121-allyourbayes/dsc01228-02-h.jpg

Theoretical Details

Main Points:

- We want our model to have the least risk possible
- Getting the tightest bound on the risk is best, theoretically
- Transduction theoretically follows the tightest bound

Theoretical Details

Main Points:

- We want our model to have the least risk possible
- Getting the tightest bound on the risk is best, theoretically
- Transduction theoretically follows the tightest bound

Admissions of Ignorance:

- I'm not exactly sure what the function f is
- I don't know why transduction theoretically follows the tightest bound

Useful tidbit:

- VC = Vapnik-Chervonenkis

Theoretical Details

True Risk Functional (accounting for the true probability distribution):

$$R(\alpha) = \int |y - f(x, \alpha)| dP(x, y)$$

Why can't we calculate this very easily?

- We would need to know P(x,y) beforehand; but if we knew P(x,y), then we wouldn't need to do machine learning (marginalize y to get P(x), divide P(x,y) by P(x) to get P(y|x), choose most probably y)

V. Vapnik. "Transductive Inference and Semi-Supervised Learning." In *Semi-supervised Learning*, 1st ed. Cambridge, Mass.: MIT Press, 2006, ch. 24.

Theoretical Details

True Risk Functional (accounting for the true probability distribution):

$$R(\alpha) = \int |y - f(x, \alpha)| dP(x, y)$$

Empirical Risk Functional (making do with the data we have):

$$R_{emp}(\alpha) = \sum_{i=1}^{\ell} |y_i - f(x_i, \alpha)|$$

V. Vapnik. “Transductive Inference and Semi-Supervised Learning.” In *Semi-supervised Learning*, 1st ed. Cambridge, Mass.: MIT Press, 2006, ch. 24.

Theoretical Details

VC Entropy

VC Dimension

$$H_P^{\Lambda}(\ell) \leq \ln G^{\Lambda}(\ell) \leq h\left(\ln \frac{\ell}{h} + 1\right)$$

Log of Growth Function

V. Vapnik. “Transductive Inference and Semi-Supervised Learning.” In *Semi-supervised Learning*, 1st ed. Cambridge, Mass.: MIT Press, 2006, ch. 24.

Theoretical Details

VC Entropy Bound

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{H_P^{\Lambda}(2\ell) - \ln \eta}{\ell}}$$

Growth Function Bound

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{\ln G^{\Lambda}(2\ell) - \ln \eta}{\ell}}$$

VC Dimension Bound

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\ln \frac{2\ell}{h} + 1) - \ln \eta}{\ell}}$$

V. Vapnik. “Transductive Inference and Semi-Supervised Learning.” In *Semi-supervised Learning*, 1st ed. Cambridge, Mass.: MIT Press, 2006, ch. 24.

Theoretical Details

Again, I'm not sure how these theoretical details relate to the implementation of transduction.

I have put them in the slides because Vapnik seems to think that they are important.

He does talk about structural risk minimization and applies the bounds to come up with an inequality over test set vectors, which seems like something to do with transduction, but nearly all of that went over my head.

V. Vapnik. "Transductive Inference and Semi-Supervised Learning." In *Semi-supervised Learning*, 1st ed. Cambridge, Mass.: MIT Press, 2006, ch. 24.

Algorithm Walkthrough (Transductive SVM)

OP 2 (Transductive SVM (non-sep. case))

Minimize over $(y_1^*, ..., y_n^*, \vec{w}, b, \xi_1, ..., \xi_n, \xi_1^*, ..., \xi_k^*)$:

$$\frac{1}{2}||\vec{w}||^2 + C\sum_{i=0}^{n}\xi_i + C^*\sum_{j=0}^{k}\xi_j^*$$

subject to:

$$\forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x}_i + b] \geq 1 - \xi_i$$

$$\forall_{j=1}^{k} : y_j^*[\vec{w}\cdot\vec{x}_j^* + b] \geq 1 - \xi_j^*$$

$$\forall_{i=1}^{n} : \xi_i > 0$$

$$\forall_{j=1}^{k} : \xi_j^* > 0$$

T. Joachims. "Transductive inference for text classification using support vector machines." ICML, 1999.

Algorithm Walkthrough (Transductive SVM)

Algorithm TSVM:

Input: – training examples $(\vec{x}_1, y_1), ..., (\vec{x}_n, y_n)$
– test examples $\vec{x}_1^*, ..., \vec{x}_k^*$
Parameters: – C,C^*: parameters from OP(2)
– num_+: number of test examples to be assigned to class +
Output: – predicted labels of the test examples $y_1^*, ..., y_k^*$

$(\vec{w}, b, \vec{\xi}, \_) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [], C, 0, 0)$;

Classify the test examples using $< \vec{w}, b >$. The num_+ test examples with the highest value of $\vec{w} * \vec{x}_j^* + b$ are assigned to the class + $(y_j^* := 1)$; the remaining test examples are assigned to class − $(y_j^* := -1)$.

$C_-^* := 10^{-5}$; // some small number
$C_+^* := 10^{-5} * \frac{num_+}{k - num_+}$;
while$((C_-^* < C^*) \ \| \ (C_+^* < C^*))\{$ // Loop 1
  $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [(\vec{x}_1^*, y_1^*)...(\vec{x}_k^*, y_k^*)], C, C_-^*, C_+^*)$;
  while$(\exists m, l : (y_m^* * y_l^* < 0) \& (\xi_m^* > 0) \& (\xi_l^* > 0) \& (\xi_m^* + \xi_l^* > 2))$ { // Loop 2
    $y_m^* := -y_m^*$; // take a positive and a negative test
    $y_l^* := -y_l^*$; // example, switch their labels, and retrain
    $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [(\vec{x}_1^*, y_1^*)...(\vec{x}_k^*, y_k^*)], C, C_-^*, C_+^*)$;
  }
  $C_-^* := min(C_-^* * 2, C^*)$;
  $C_+^* := min(C_+^* * 2, C^*)$;
}
return$(y_1^*, ..., y_k^*)$;

T. Joachims. “Transductive inference for text classification using support vector machines.” ICML, 1999.

Algorithm Walkthrough (Transductive SVM)

Input: – training examples $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$
– test examples $\vec{x}_1^*, \ldots, \vec{x}_k^*$

Parameters: – C,C^*: parameters from OP(2)
– num_+: number of test examples to be assigned to class +

Output: – predicted labels of the test examples $y_1^*, \ldots, y_k^*$

Note that Input and Output follows the definition of a transduction problem

T. Joachims. "Transductive inference for text classification using support vector machines." ICML, 1999.

Algorithm Walkthrough (Transductive SVM)

Input: – training examples $(\vec{x}_1, y_1), ..., (\vec{x}_n, y_n)$
– test examples $\vec{x}_1^*, ..., \vec{x}_k^*$
Parameters: – C,C^*: parameters from OP(2)
– num_+: number of test examples to be assigned to class +
Output: – predicted labels of the test examples $y_1^*, ..., y_k^*$

As for Parameters:

OP 2 (Transductive SVM (non-sep. case))
Minimize over $(y_1^*, ..., y_n^*, \vec{w}, b, \xi_1, ..., \xi_n, \xi_1^*, ..., \xi_k^*)$:

$$\frac{1}{2}||\vec{w}||^2 + C\sum_{i=0}^{n}\xi_i + C^*\sum_{j=0}^{k}\xi_j^*$$

subject to:

$$\forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x}_i + b] \geq 1 - \xi_i$$
$$\forall_{j=1}^{k} : y_j^*[\vec{w}\cdot\vec{x}_j^* + b] \geq 1 - \xi_j^*$$
$$\forall_{i=1}^{n} : \xi_i > 0$$
$$\forall_{j=1}^{k} : \xi_j^* > 0$$

T. Joachims. "Transductive inference for text classification using support vector machines." ICML, 1999.

Algorithm Walkthrough (Transductive SVM)

$(\vec{w}, b, \vec{\xi}, \_) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [], C, 0, 0);$

The function *solve_svm_qp* refers to quadratic programs of the following type.

OP 3 (Inductive SVM (primal))

Minimize over $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*)$*:*

$$\frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{n}\xi_i + C_-^*\sum_{j:y_j^*=-1}\xi_j^* + C_+^*\sum_{j:y_j^*=1}\xi_j^*$$

subject to:

$$\forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x_i} + b] \geq 1 - \xi_i$$

$$\forall_{j=1}^{k} : y_j^*[\vec{w}\cdot\vec{x_j} + b] \geq 1 - \xi_j^*$$

T. Joachims. “Transductive inference for text classification using support vector machines.” ICML, 1999.

Algorithm Walkthrough (Transductive SVM)

$(\vec{w}, b, \vec{\xi}, \_) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [], C, 0, 0);$

The function *solve_svm_qp* refers to quadratic programs of the following type.

OP 3 (Inductive SVM (primal))

Minimize over $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*)$*:*

$$\frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{n}\xi_i + C_-^*\sum_{j:y_j^*=-1}\xi_j^* + C_+^*\sum_{j:y_j^*=1}\xi_j^*$$

subject to:

$$\forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x_i} + b] \geq 1 - \xi_i$$
$$\forall_{j=1}^{k} : y_j^*[\vec{w}\cdot\vec{x_j} + b] \geq 1 - \xi_j^*$$

Why aren't these constraints part of OP 3?

$$\forall_{i=1}^{n} : \xi_i > 0$$
$$\forall_{j=1}^{k} : \xi_j^* > 0$$

T. Joachims. "Transductive inference for text classification using support vector machines." ICML, 1999.

Algorithm Walkthrough (Transductive SVM)

$(\vec{w}, b, \vec{\xi}, \_) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [], C, 0, 0);$

The function *solve_svm_qp* refers to quadratic programs of the following type.

OP 3 (Inductive SVM (primal))

Minimize over $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*)$*:*

$$\frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{n}\xi_i + C_-^*\sum_{j:y_j^*=-1}\xi_j^* + C_+^*\sum_{j:y_j^*=1}\xi_j^*$$

subject to:

$$\forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x_i} + b] \geq 1 - \xi_i$$
$$\forall_{j=1}^{k} : y_j^*[\vec{w}\cdot\vec{x_j} + b] \geq 1 - \xi_j^*$$

Why aren't these constraints part of OP 3?

$$\forall_{i=1}^{n} : \xi_i > 0$$
$$\forall_{j=1}^{k} : \xi_j^* > 0$$

Ideally, they are fulfilled by the other constraints

T. Joachims. "Transductive inference for text classification using support vector machines." ICML, 1999.

Algorithm Walkthrough (Transductive SVM)

$(\vec{w}, b, \vec{\xi}, \_) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [], C, 0, 0);$

The function *solve_svm_qp* refers to quadratic programs of the following type.

OP 3 (Inductive SVM (primal))

Minimize over $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*)$:

$$\frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{n}\xi_i + C_-^*\sum_{j:y_j^*=-1}\xi_j^* + C_+^*\sum_{j:y_j^*=1}\xi_j^*$$

subject to:

$$\forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x_i} + b] \geq 1 - \xi_i$$
$$\forall_{j=1}^{k} : y_j^*[\vec{w}\cdot\vec{x_j} + b] \geq 1 - \xi_j^*$$

Why aren't these constraints part of OP 3?

$$\forall_{i=1}^{n} : \xi_i > 0$$
$$\forall_{j=1}^{k} : \xi_j^* > 0$$

Ideally, they are fulfilled by the other constraints

Then why are they part of OP 2? I don't know...

T. Joachims. "Transductive inference for text classification using support vector machines." ICML, 1999.

Algorithm Walkthrough (Transductive SVM)

Classify the test examples using $< \vec{w}, b >$. The num_+ test examples with the highest value of $\vec{w} * \vec{x}_j^* + b$ are assigned to the class + $(y_j^* := 1)$; the remaining test examples are assigned to class − $(y_j^* := -1)$.

$C_-^* := 10^{-5}$; // some small number

$C_+^* := 10^{-5} * \frac{num_+}{k - num_+}$;

Classification is based on a model trained on only labeled data

T. Joachims. "Transductive inference for text classification using support vector machines." ICML, 1999.

Algorithm Walkthrough (Transductive SVM)

Classify the test examples using $< \vec{w}, b >$. The num_+ test examples with the highest value of $\vec{w} * \vec{x}_j^* + b$ are assigned to the class + $(y_j^* := 1)$; the remaining test examples are assigned to class − $(y_j^* := -1)$.

$C_-^* := 10^{-5}$; // some small number

$C_+^* := 10^{-5} * \frac{num_+}{k - num_+}$;

The small numbers refer back to OP 3

The function *solve_svm_qp* refers to quadratic programs of the following type.

OP 3 (Inductive SVM (primal))

Minimize over $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*)$*:*

$$\frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{n}\xi_i + C_-^*\sum_{j:y_j^*=-1}\xi_j^* + C_+^*\sum_{j:y_j^*=1}\xi_j^*$$

subject to: $\forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x}_i + b] \geq 1 - \xi_i$

$\forall_{j=1}^{k} : y_j^*[\vec{w}\cdot\vec{x}_j + b] \geq 1 - \xi_j^*$

T. Joachims. "Transductive inference for text classification using support vector machines." ICML, 1999.

Algorithm Walkthrough (Transductive SVM)

```
while((C*_- < C*) || (C*_+ < C*)){                                    // Loop 1
    (w, b, ξ, ξ*) := solve_svm_qp([(x_1, y_1)...(x_n, y_n)], [(x*_1, y*_1)...(x*_k, y*_k)], C, C*_-, C*_+);
    while(∃m,l : (y*_m * y*_l < 0)&(ξ*_m > 0)&(ξ*_l > 0)&(ξ*_m + ξ*_l > 2)) {   // Loop 2
        y*_m := -y*_m;                  // take a positive and a negative test
        y*_l := -y*_l;                  // example, switch their labels, and retrain
        (w, b, ξ, ξ*) := solve_svm_qp([(x_1, y_1)...(x_n, y_n)], [(x*_1, y*_1)...(x*_k, y*_k)], C, C*_-, C*_+);
    }
    C*_- := min(C*_- * 2, C*);
    C*_+ := min(C*_+ * 2, C*);
}
```

Loop 1: While the small numbers are small, keep changing them

T. Joachims. "Transductive inference for text classification using support vector machines." ICML, 1999.

Algorithm Walkthrough (Transductive SVM)

```
while((C*_- < C*) || (C*_+ < C*)){                                   // Loop 1
    (w, b, ξ, ξ*) := solve_svm_qp([(x_1, y_1)...(x_n, y_n)], [(x*_1, y*_1)...(x*_k, y*_k)], C, C*_-, C*_+);
    while(∃m,l : (y*_m * y*_l < 0)&(ξ*_m > 0)&(ξ*_l > 0)&(ξ*_m + ξ*_l > 2)) {   // Loop 2
        y*_m := -y*_m;                  // take a positive and a negative test
        y*_l := -y*_l;                  // example, switch their labels, and retrain
        (w, b, ξ, ξ*) := solve_svm_qp([(x_1, y_1)...(x_n, y_n)], [(x*_1, y*_1)...(x*_k, y*_k)], C, C*_-, C*_+);
    }
    C*_- := min(C*_- * 2, C*);
    C*_+ := min(C*_+ * 2, C*);
}
```

Loop 2: Swap test instance labels while corresponding penalties don't match

T. Joachims. "Transductive inference for text classification using support vector machines." ICML, 1999.

Algorithm Walkthrough (Transductive SVM)

Algorithm TSVM:

Input: – training examples $(\vec{x}_1, y_1), ..., (\vec{x}_n, y_n)$
– test examples $\vec{x}_1^*, ..., \vec{x}_k^*$
Parameters: – C,C^*: parameters from OP(2)
– num_+: number of test examples to be assigned to class +
Output: – predicted labels of the test examples $y_1^*, ..., y_k^*$

$(\vec{w}, b, \vec{\xi}, \_) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [], C, 0, 0)$;

Classify the test examples using $< \vec{w}, b >$. The num_+ test examples with the highest value of $\vec{w} * \vec{x}_j^* + b$ are assigned to the class + $(y_j^* := 1)$; the remaining test examples are assigned to class − $(y_j^* := -1)$.

$C_-^* := 10^{-5}$; // some small number
$C_+^* := 10^{-5} * \frac{num_+}{k - num_+}$;
while$((C_-^* < C^*) \ \| \ (C_+^* < C^*))\{$ // Loop 1
  $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [(\vec{x}_1^*, y_1^*)...(\vec{x}_k^*, y_k^*)], C, C_-^*, C_+^*)$;
  while$(\exists m, l : (y_m^* * y_l^* < 0) \& (\xi_m^* > 0) \& (\xi_l^* > 0) \& (\xi_m^* + \xi_l^* > 2))$ { // Loop 2
    $y_m^* := -y_m^*$; // take a positive and a negative test
    $y_l^* := -y_l^*$; // example, switch their labels, and retrain
    $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [(\vec{x}_1^*, y_1^*)...(\vec{x}_k^*, y_k^*)], C, C_-^*, C_+^*)$;
  }
  $C_-^* := min(C_-^* * 2, C^*)$;
  $C_+^* := min(C_+^* * 2, C^*)$;
}
return$(y_1^*, ..., y_k^*)$;

T. Joachims. “Transductive inference for text classification using support vector machines.” ICML, 1999.

Algorithm Walkthrough (Transductive SVM)

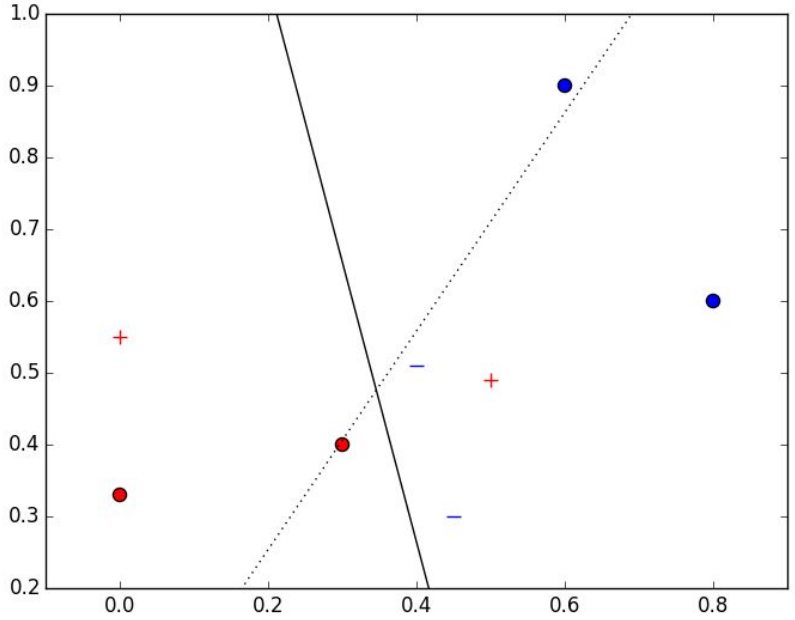
Red is 1, Blue is -1

\+ is given 1, - is given -1

Circles are test data

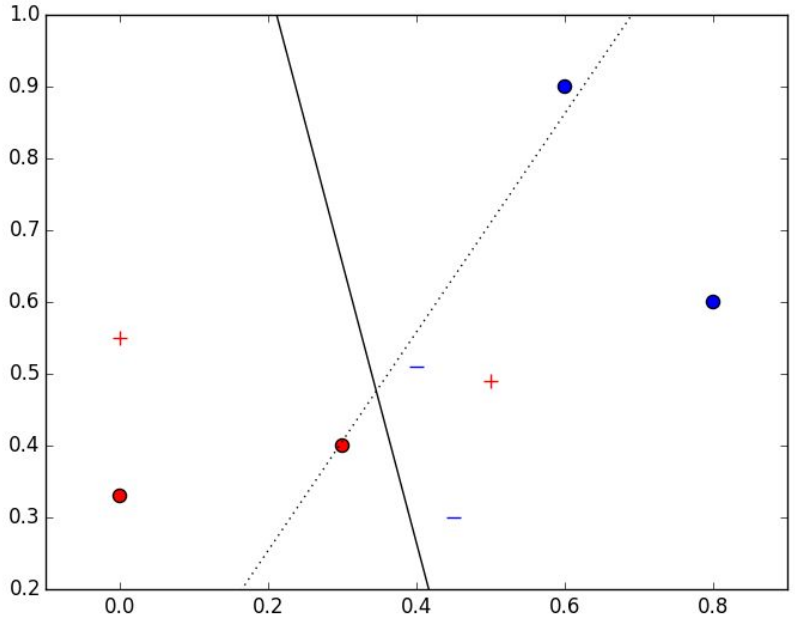
Dashed line is initial hyperplane

Solid line is final hyperplane



Algorithm Walkthrough (Transductive SVM)

0	0.55	1
0.5	0.49	1
0	0.33	?
0.3	0.4	?
0.6	0.9	?
0.8	0.6	?
0.4	0.51	-1
0.45	0.3	-1



Algorithm Walkthrough (Transductive SVM)

```
Initial
########################################
Initial
predictions:    [1,-1,1,-1]
weights:    [-0.3499999989415537,0.2299999993580931]
bias:   0.0116206854790473
training penalties:
[0.8618793073756819,1.0506793068090083,0.9889206780774836,0.9231206782648367]
testing penalties:  Any[]
#--------------------------------------#
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    C_star_plus:    9.999999999999999e-6
    C_star_minus:   9.999999999999999e-6
C**************************************C
```

$$(\vec{w}, b, \vec{\xi}, \_) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [], C, 0, 0);$$

Algorithm Walkthrough (Transductive SVM)

```
Initial
########################################
Initial
predictions:    [1,-1,1,-1]
weights:    [-0.3499999989415537,0.2299999993580931]
bias:   0.0116206854790473
training penalties:
[0.8618793073756819,1.0506793068090083,0.9889206780774836,0.9231206782648367]
testing penalties:  Any[]
#--------------------------------------#
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    C_star_plus:    9.999999999999999e-6
    C_star_minus:   9.999999999999999e-6
C**************************************C
```

Classify the test examples using $< \vec{w}, b >$. The num_+ test examples with the highest value of $\vec{w} * \vec{x}_j^* + b$ are assigned to the class + $(y_j^* := 1)$; the remaining test examples are assigned to class − $(y_j^* := -1)$.

Algorithm Walkthrough (Transductive SVM)

```
Initial
########################################
Initial
predictions:    [1,-1,1,-1]
weights:    [-0.3499999989415537,0.2299999993580931]
bias:   0.0116206854790473
training penalties:
[0.8618793073756819,1.0506793068090083,0.9889206780774836,0.9231206782648367]
testing penalties:  Any[]
#--------------------------------------#
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    C_star_plus:    9.999999999999999e-6
    C_star_minus:   9.999999999999999e-6
C**************************************C
```

$C_-^* := 10^{-5};$ // some small number

$C_+^* := 10^{-5} * \frac{num_+}{k - num_+};$

Algorithm Walkthrough (Transductive SVM)

```
Iteration: 1
########################################
Iteration: 1
predictions:    [1,-1,1,-1]
weights:    [-0.35000499744602587,0.23000229886942783]
bias:   0.04813871789811412
training penalties:
[0.8253600102296041,1.0141626468847826,1.0254378838490155,0.9596371512141343]
testing penalties:
[0.8762113078710513,1.0353887820662886,0.955112927968292,0.9063867484489777]
#---------------------------------------#
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    C_star_plus:    1.9999999999999998e-5
    C_star_minus:   1.9999999999999998e-5
C*****************************************C
```

$$(\vec{w}, b, \vec{\xi}, \vec{\xi}^*) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [(\vec{x}_1^*, y_1^*)...(\vec{x}_k^*, y_k^*)], C, C_-^*, C_+^*);$$

Algorithm Walkthrough (Transductive SVM)

```
Iteration: 1
########################################
Iteration: 1
predictions:    [1,-1,1,-1]
weights:    [-0.35000499744602587,0.23000229886942783]
bias:   0.04813871789811412
training penalties:
[0.8253600102296041,1.0141626468847826,1.0254378838490155,0.9596371512141343]
testing penalties:
[0.8762113078710513,1.0353887820662886,0.955112927968292,0.9063867484489777]
#--------------------------------------#
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    C_star_plus:    1.9999999999999998e-5
    C_star_minus:   1.9999999999999998e-5
C**************************************C
```

$$C_{-}^{*} := min(C_{-}^{*} * 2, C^{*});$$
$$C_{+}^{*} := min(C_{+}^{*} * 2, C^{*});$$

Algorithm Walkthrough (Transductive SVM)

```
Iteration: 16
########################################
Iteration: 16
predictions:    [1,-1,1,-1]
weights:    [-0.5138399971676509,0.3053663988809765]
bias:   0.14047075283260615
training penalties:
[0.6915777202844386,0.9668197028022565,1.0906716098973297,1.0008526662735882]
testing penalties:
[0.7587583331474819,1.1084653108507676,0.8930034840908901,0.9126185920347551]
#--------------------------------------#
```

while $(\exists m, l : (y_m^* * y_l^* < 0) \& (\xi_m^* > 0) \& (\xi_l^* > 0) \& (\xi_m^* + \xi_l^* > 2))$ { // Loop 2

Algorithm Walkthrough (Transductive SVM)

```
Iteration: 16
########################################
Iteration: 16
predictions:    [1,-1,1,-1]
weights:    [-0.5138399971676509,0.3053663988809765]
bias:   0.14047075283260615
training penalties:
[0.6915777202844386,0.9668197028022565,1.0906716098973297,1.0008526662735882]
testing penalties:
[0.7587583331474819,1.1084653108507676,0.8930034840908901,0.9126185920347551]
#--------------------------------------#
```

$$y_m^* := -y_m^*; \qquad \text{// take a positive and a negative test}$$
$$y_l^* := -y_l^*; \qquad \text{// example, switch their labels, and retrain}$$
$$(\vec{w}, b, \vec{\xi}, \vec{\xi}^*) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [(\vec{x}_1^*, y_1^*)...(\vec{x}_k^*, y_k^*)], C, C_-^*, C_+^*);$$

Algorithm Walkthrough (Transductive SVM)

```
Iteration: 16
########################################
Iteration: 16
predictions:    [1,-1,1,-1]
weights:    [-0.5138399971676509,0.3053663988809765]
bias:   0.14047075283260615
training penalties:
[0.6915777202844386,0.9668197028022565,1.0906716098973297,1.0008526662735882]
testing penalties:
[0.7587583331474819,1.1084653108507676,0.8930034840908901,0.9126185920347551]
#-------------------------------------#
#### Sum: 2.001468794941658
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
    index1: 2   1.1084653108507676
    index2: 3   0.8930034840908901
I*************************************I
```

Algorithm Walkthrough (Transductive SVM)

```
Iteration: 17
Swapped 2 and 3
########################################
Iteration: 17
Swapped 2 and 3
predictions:    [1,1,-1,-1]
weights:    [-0.7104479950209311,-0.022313598219624 53]
bias:   0.25134504027522153
training penalties:
[0.7609274312467963,1.1148126128656513,0.9557858996772918,0.9249493555522572]
testing penalties:
[0.7560184447407811,0.9707147951327828,0.8049940024740828,0.6695984829281191]
#--------------------------------------#
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    C_star_plus:    0.6553599999999999
    C_star_minus:   0.6553599999999999
C**************************************C
```

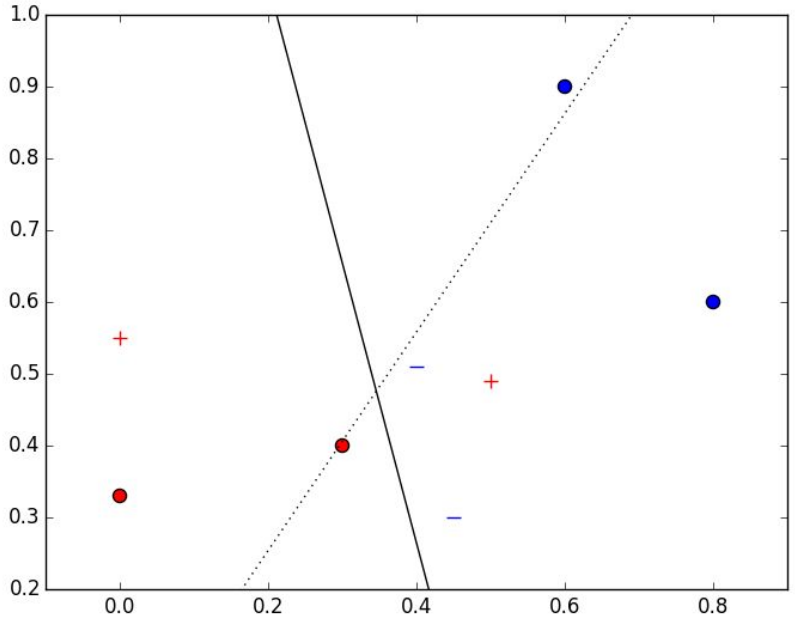
Algorithm Walkthrough (Transductive SVM)

```
Iteration: 18
########################################
Iteration: 18
predictions:    [1,1,-1,-1]
weights:    [-1.070895993602519,-0.274627197579686]
bias:   0.5010333531255504
training penalties:
[0.6500115980434442,1.1689819629926717,0.9326150774213058,0.936741892329326]
testing penalties:
[0.5895936158842809,0.9300863178018592,0.611331272953632,0.4795402335037401]
#--------------------------------------#
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    C_star_plus:    1.0
    C_star_minus:   1.0
C**************************************C
```

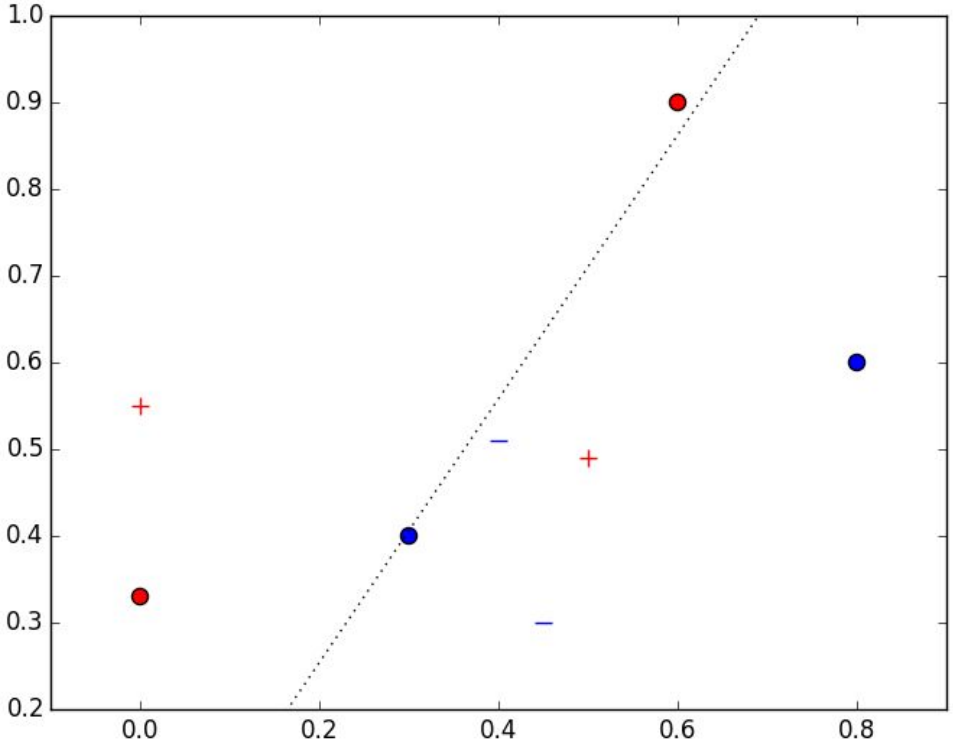
while($(C_-^* < C^*)$ || $(C_+^* < C^*)$){ // Loop 1

Algorithm Walkthrough (Transductive SVM)

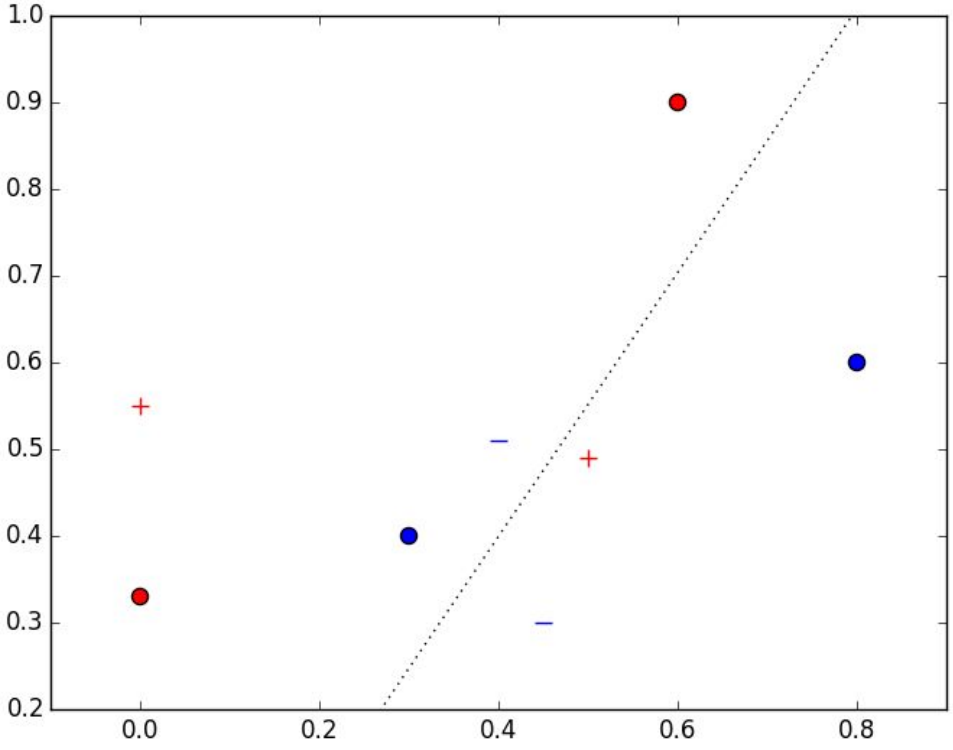
0	0.55	1
0.5	0.49	1
0	0.33	1
0.3	0.4	1
0.6	0.9	-1
0.8	0.6	-1
0.4	0.51	-1
0.45	0.3	-1



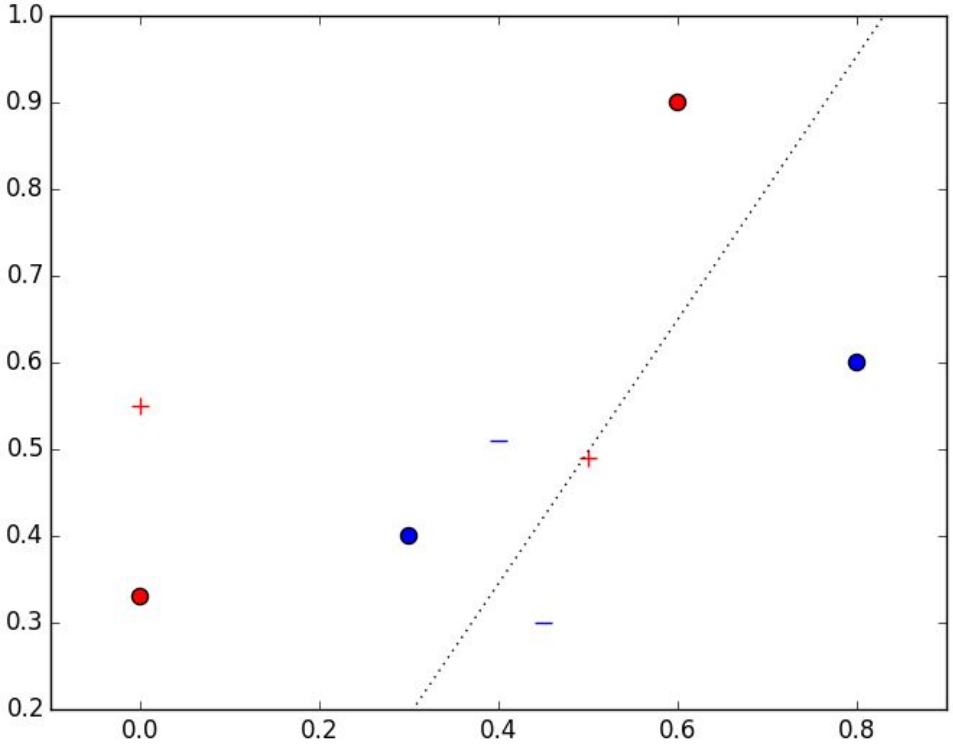
Algorithm Walkthrough (Transductive SVM)



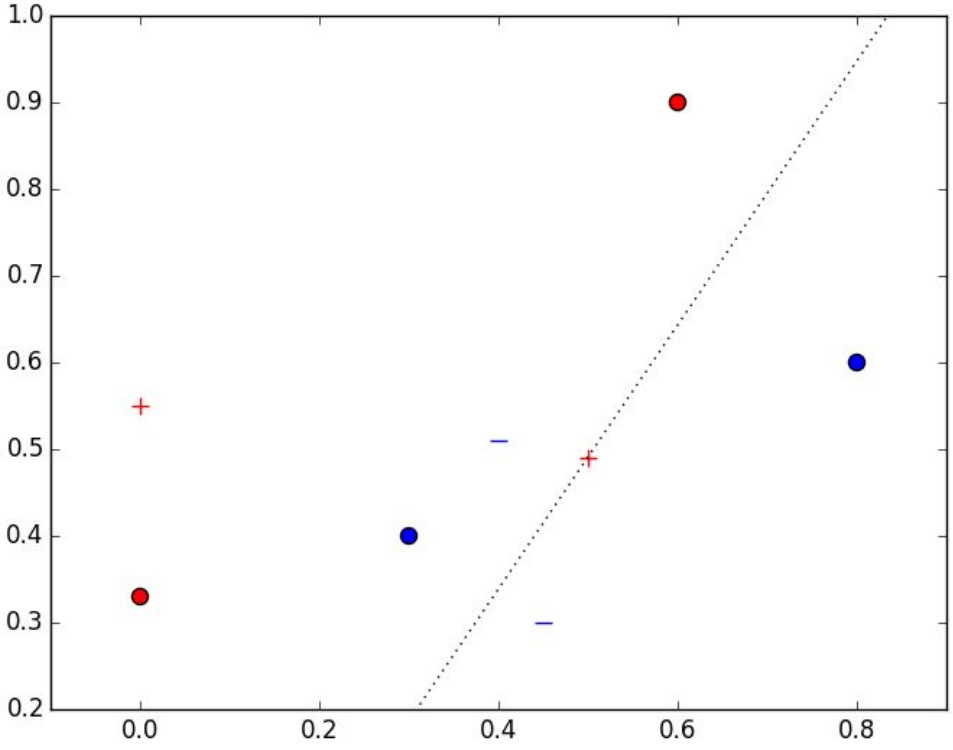
Algorithm Walkthrough (Transductive SVM)



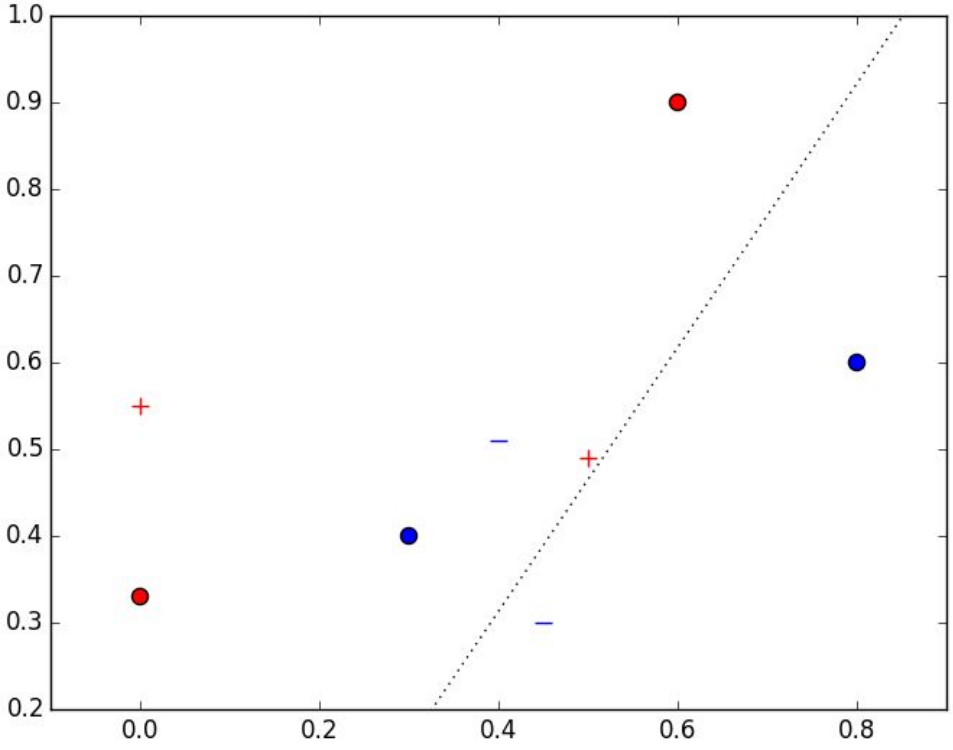
Algorithm Walkthrough (Transductive SVM)



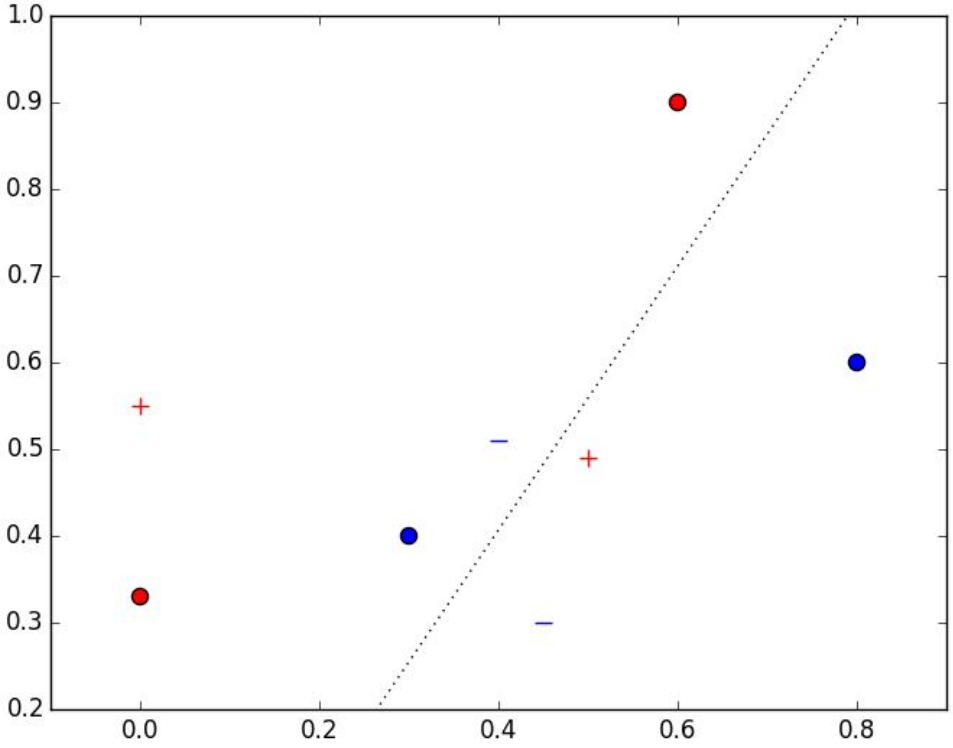
Algorithm Walkthrough (Transductive SVM)



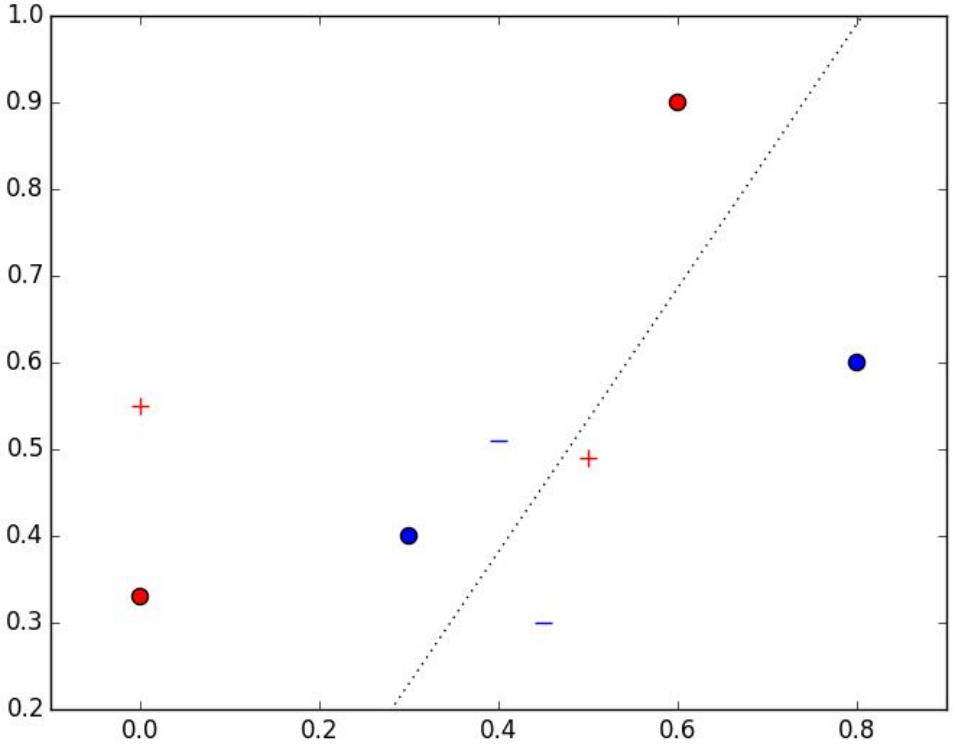
Algorithm Walkthrough (Transductive SVM)



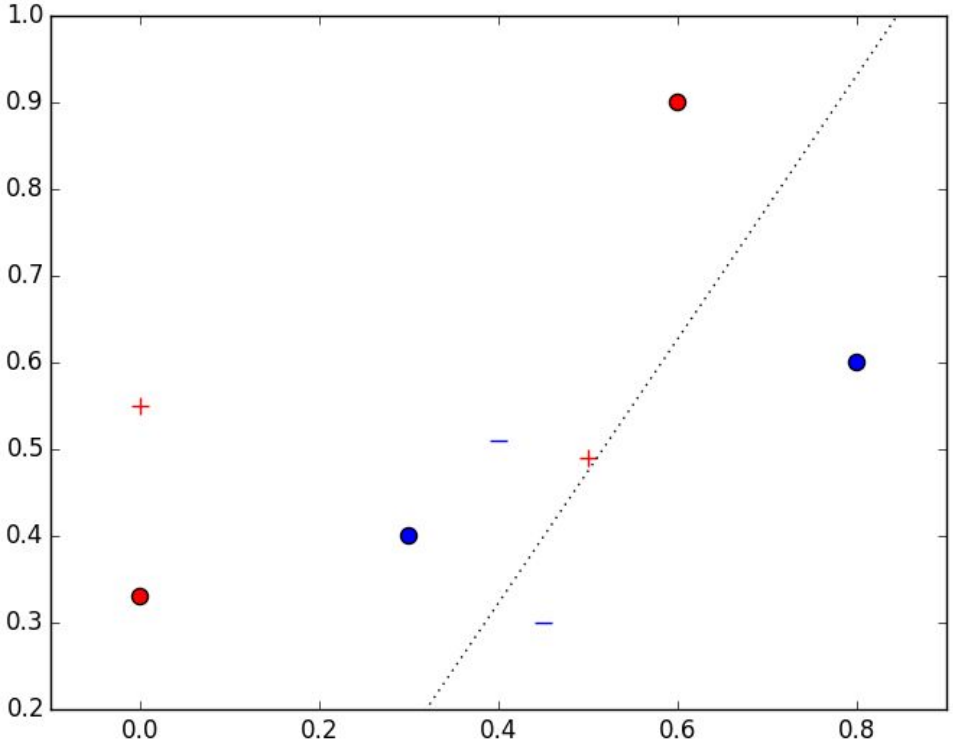
Algorithm Walkthrough (Transductive SVM)



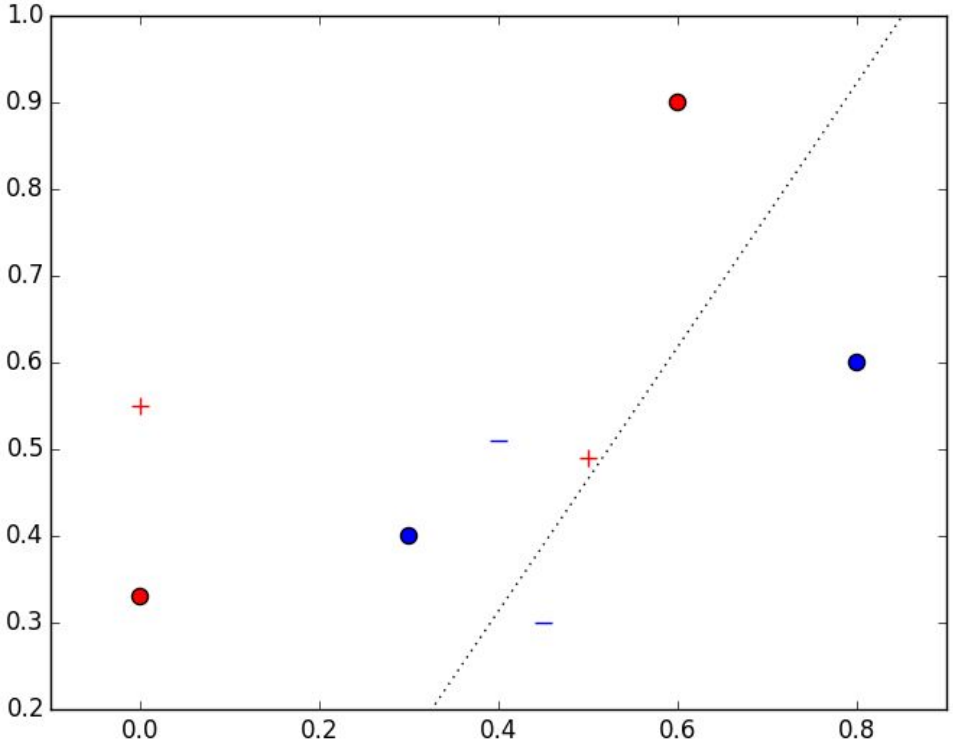
Algorithm Walkthrough (Transductive SVM)



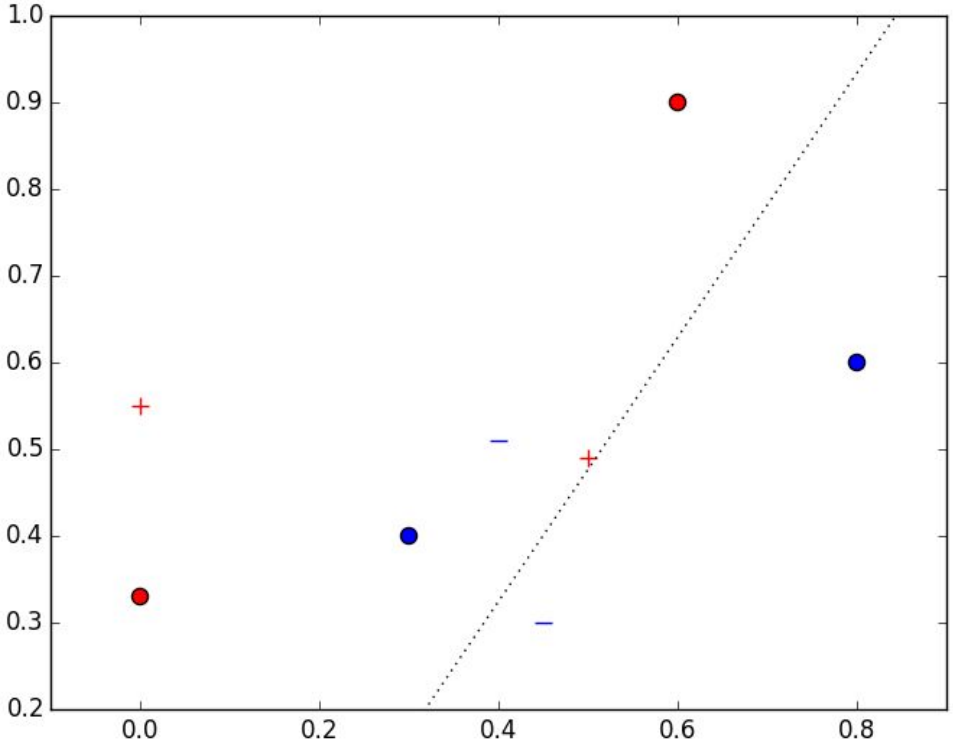
Algorithm Walkthrough (Transductive SVM)



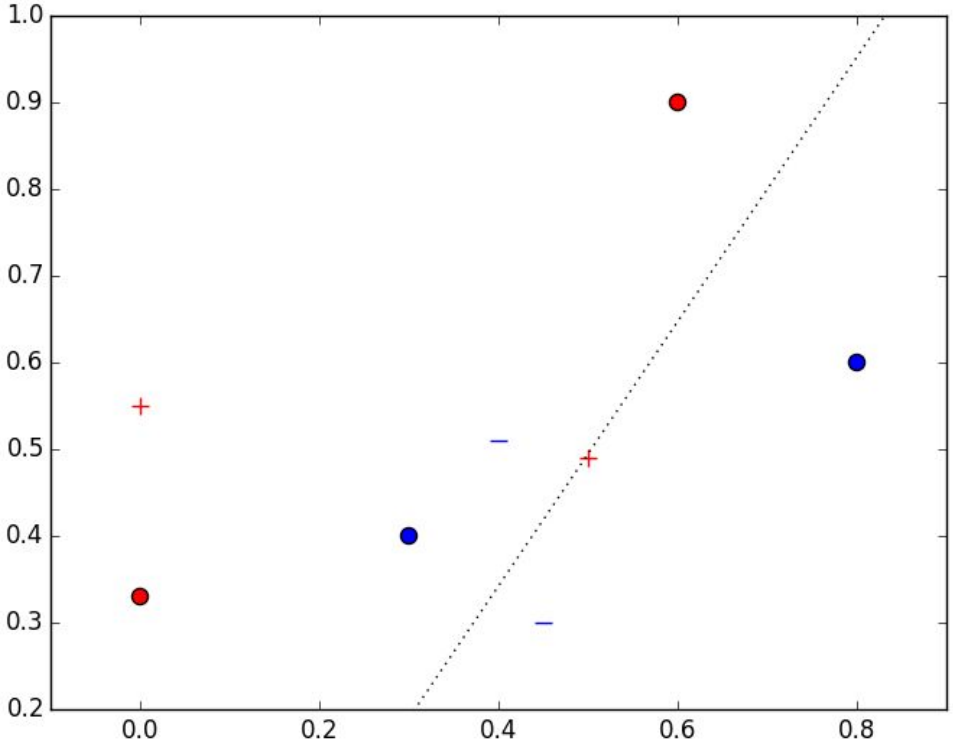
Algorithm Walkthrough (Transductive SVM)



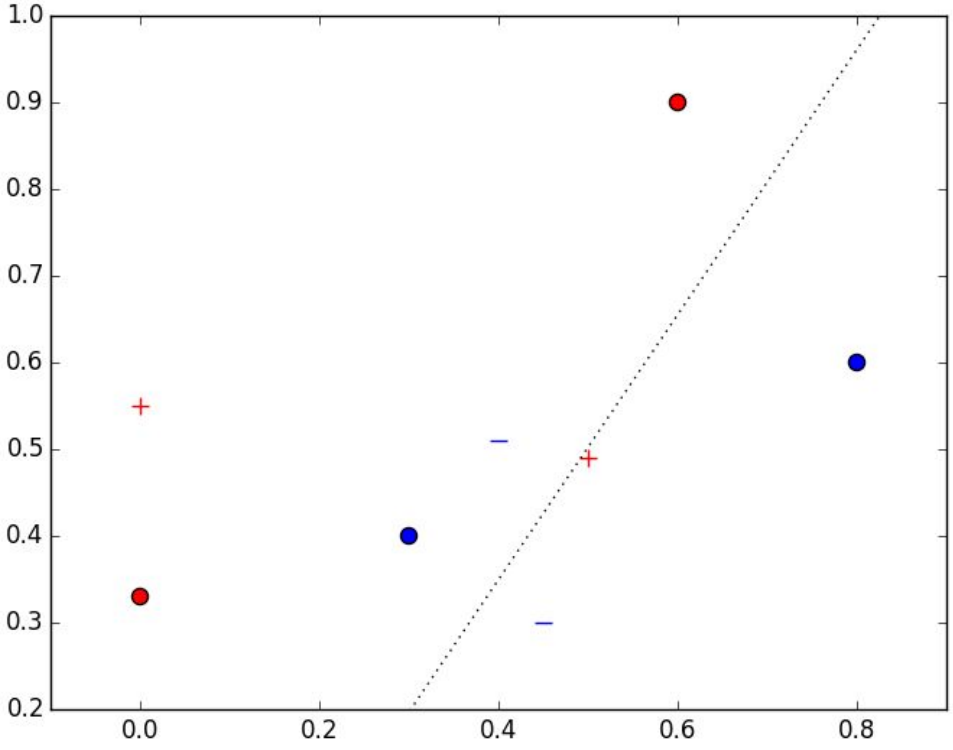
Algorithm Walkthrough (Transductive SVM)



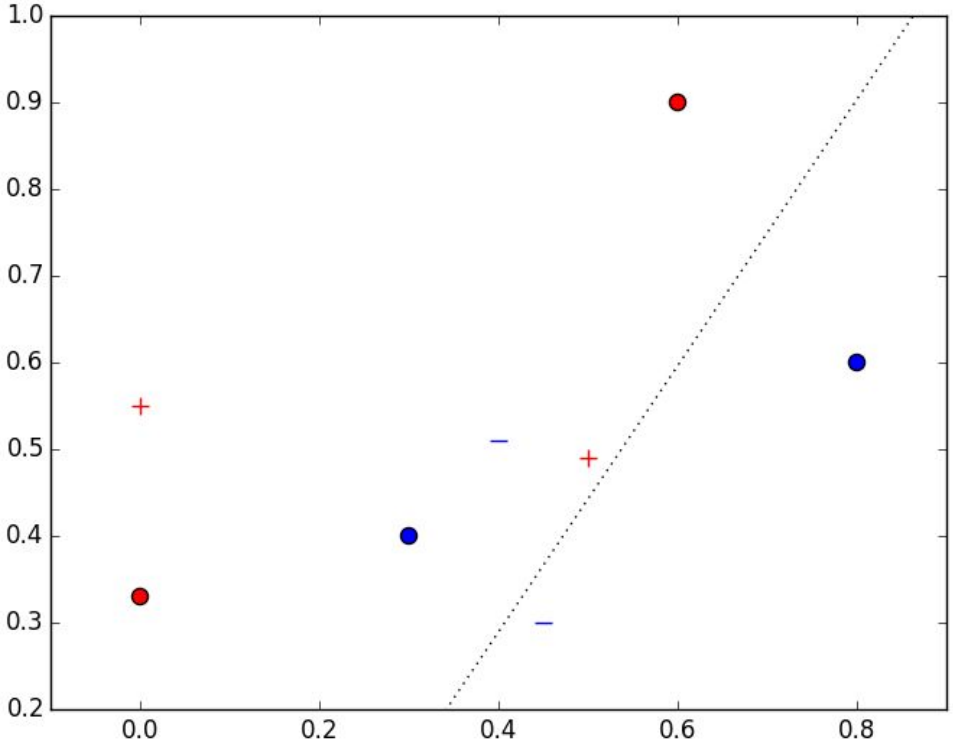
Algorithm Walkthrough (Transductive SVM)



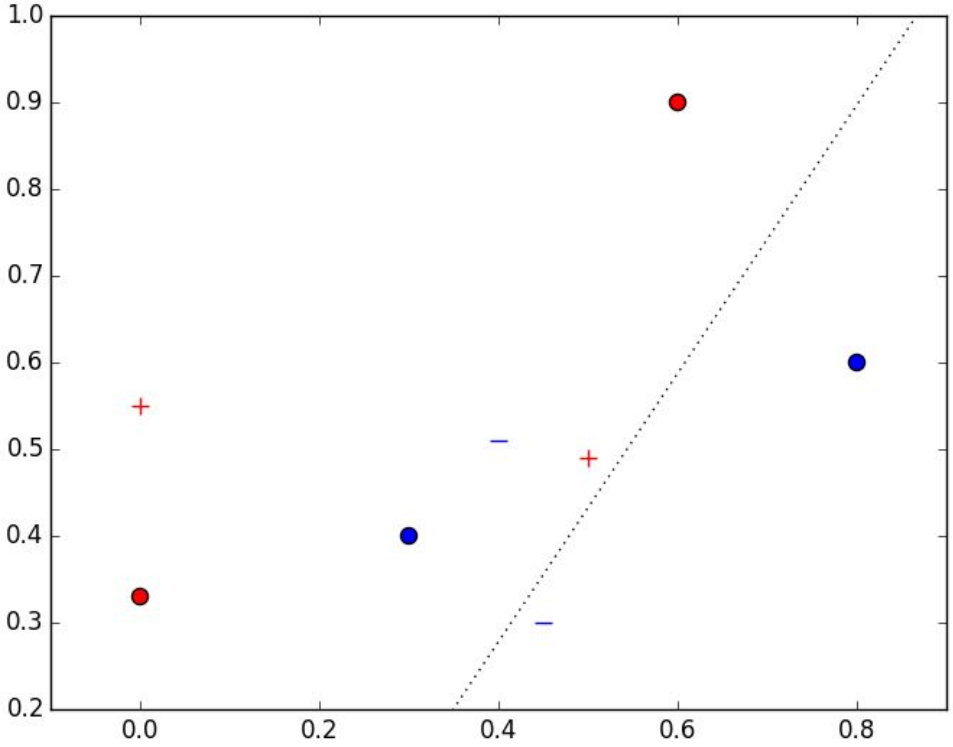
Algorithm Walkthrough (Transductive SVM)



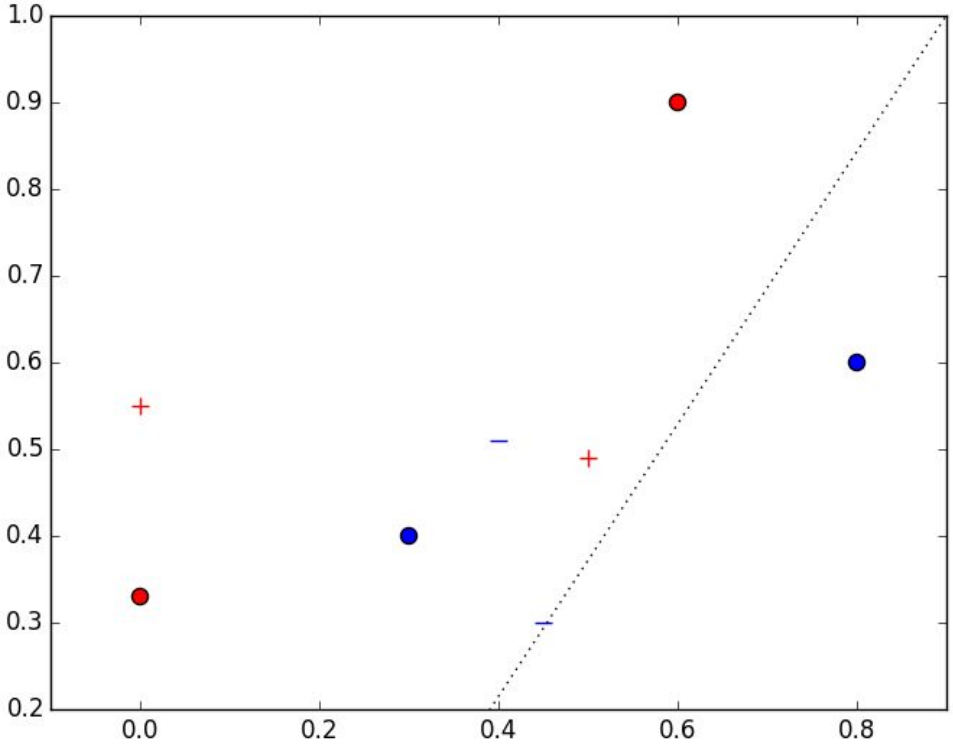
Algorithm Walkthrough (Transductive SVM)



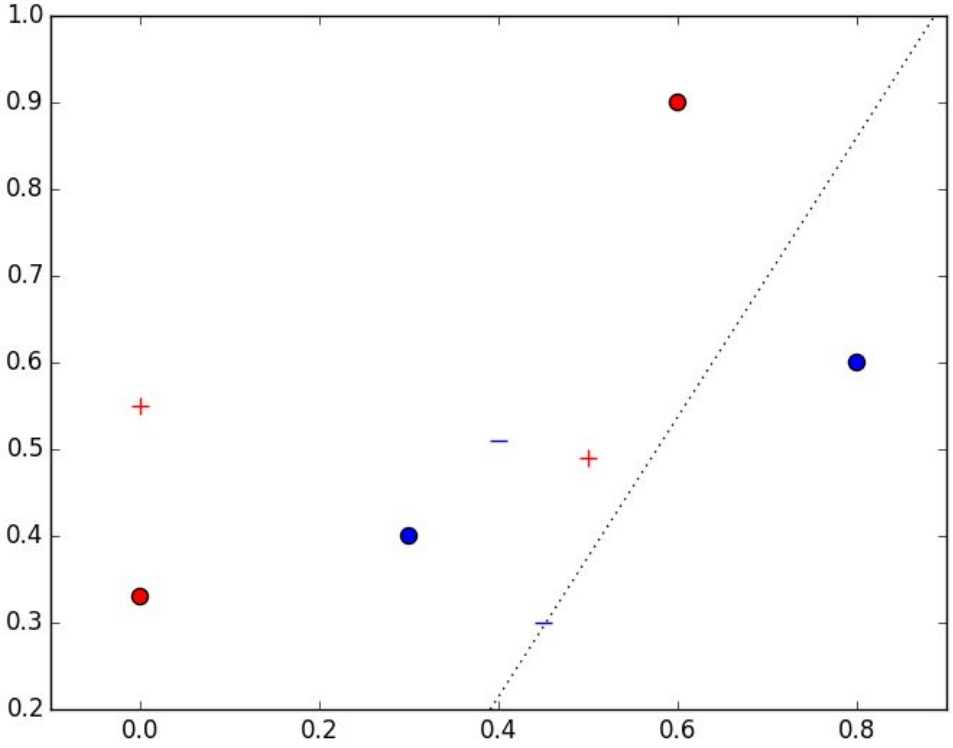
Algorithm Walkthrough (Transductive SVM)



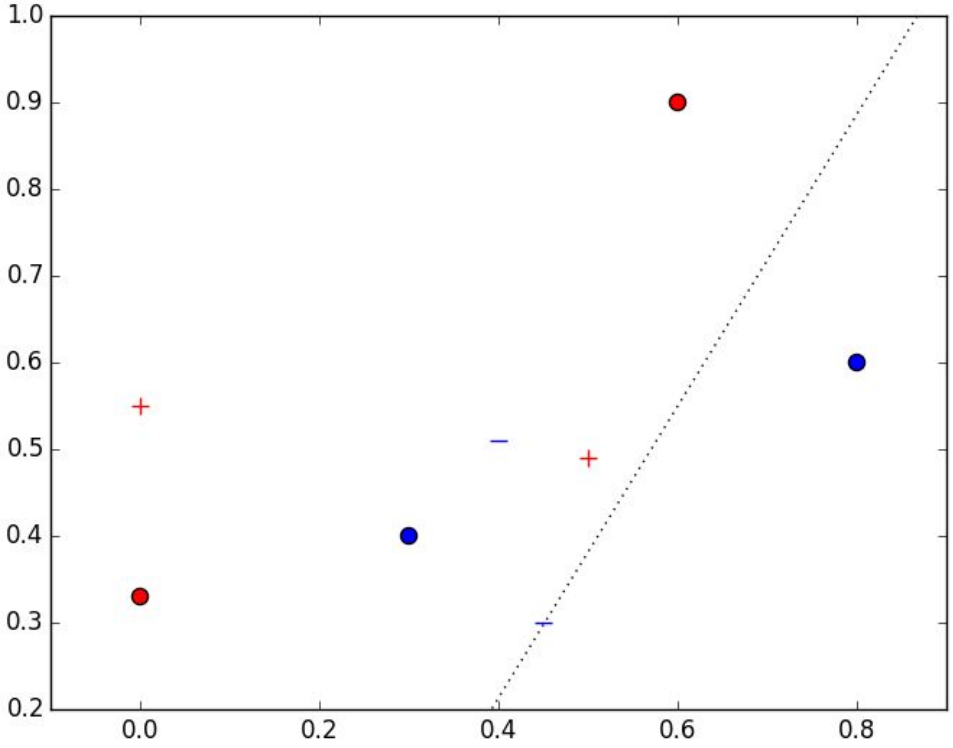
Algorithm Walkthrough (Transductive SVM)



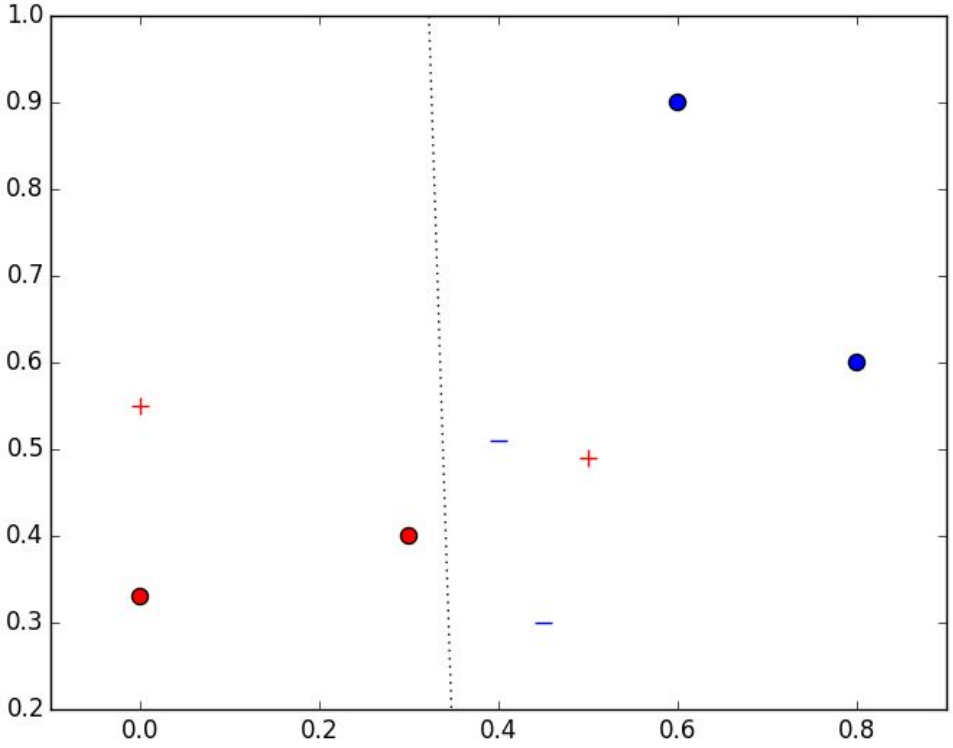
Algorithm Walkthrough (Transductive SVM)



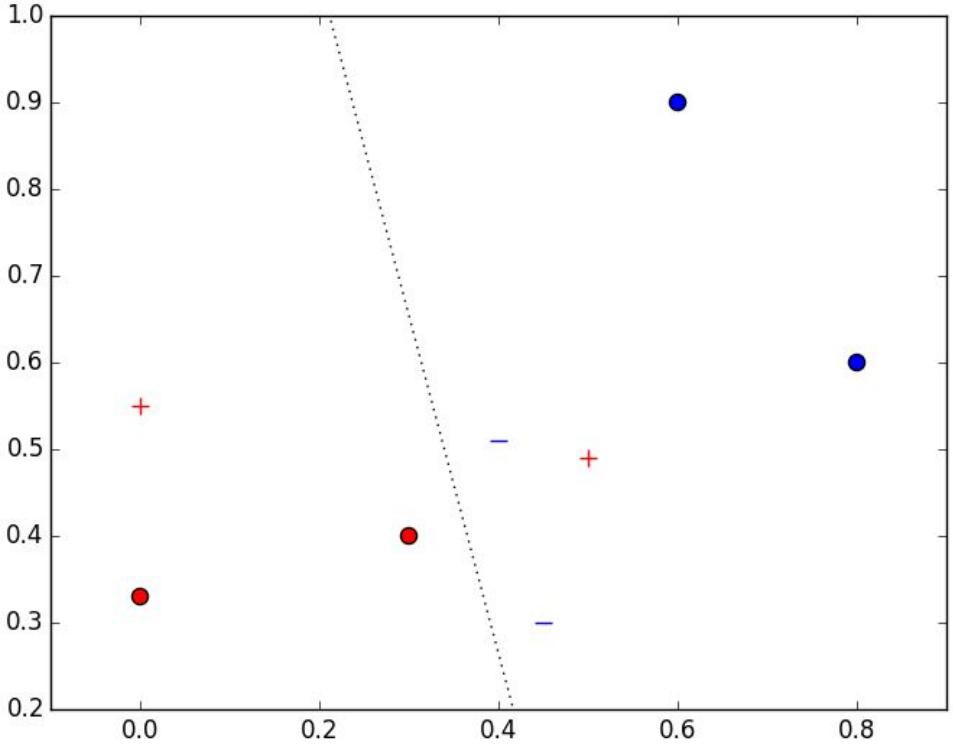
Algorithm Walkthrough (Transductive SVM)



Algorithm Walkthrough (Transductive SVM)

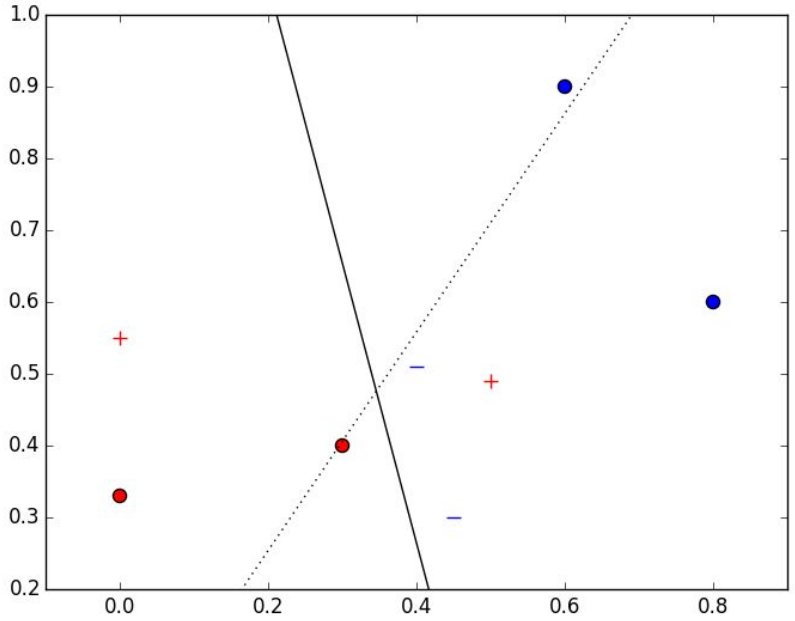


Algorithm Walkthrough (Transductive SVM)



Algorithm Walkthrough (Transductive SVM)

0	0.55	1
0.5	0.49	1
0	0.33	1
0.3	0.4	1
0.6	0.9	-1
0.8	0.6	-1
0.4	0.51	-1
0.45	0.3	-1



Implementation Results

Experimental setup:

- C = 1.0 = penalty for training example margins
- C^{*} = 1.0 = penalty for test example margins
- num_{+} = percentage of positive examples in training set multiplied by number of examples in test set = expected number of positive examples in test set

Implementation Results

Experimental setup (continued):

- 10 and 4 refer to 10-fold and 4-fold cross validation
- small refers to random 10% as training set, averaged over 10 runs
- linear svm and rbf svm are from sklearn (Python)

Implementation Results

Successful Dataset:

- forests = forest type mapping
  - https://archive.ics.uci.edu/ml/datasets/Forest+type+mapping
  - 27 attributes
  - 523 instances

Implementation Results

Swapping step got stuck going back and forth (bad solver?):

- vert = vertebral column
  - https://archive.ics.uci.edu/ml/datasets/Vertebral+Column
  - 6 attributes
  - 310 instances
- hiv = HIV-1 protease cleavage
  - https://archive.ics.uci.edu/ml/datasets/HIV-1+protease+cleavage
  - 1 attribute (8 amino acids)
  - 6590 instances

Implementation Results

	forests 10	forests 4	forests small	vert 10	vert 4	vert small
tsvm	0.90801	0.91202	0.84968	0.83871	**0.84514**	0.81362
julia svm	0.89304	0.90218	0.85499	0.83871	0.83224	**0.81433**
linear svm	**0.92184**	**0.91214**	**0.86879**	**0.84194**	0.75262	0.78996
rbf svm	0.69599	0.69599	0.69639	0.67742	0.67741	0.6362

Implementation Results

What I learned:

Implementation Results

What I learned: Don't trust theories

Theorem 2 *Algorithm 1 converges in a finite number of steps.*

Proof: To prove this, it is necessary to show that loop 2 is exited after a finite number of iterations. This

Strengths and Weaknesses of TSVM

Strengths:

- Explicitly accounts for test set examples

Weaknesses:

- Takes a long time to train
- Requires really good solver

Bells and Whistles

- Train dual instead of primal (SVMLight)
- Adjust parameters (C, C^{*}, num_{+}) to find best settings
- Kernel Trick

Current Trajectory

- Latest paper I could find was from 2013:
  - Bresson et al., Multi-class Transductive Learning Based on ℓ 1 Relaxations of Cheeger Cut and Mumford-Shah-Potts Model, Journal of Mathematical Imaging and Vision (2013)
- Graph-based (reminiscent of Locally Linear Embedding)
  - Yu et al., Adaptive Hypergraph Learning and its Application in Image Classification, IEEE Transactions on Image Processing, vol 21: issue 7 (2012)
  - Zhang et al., Transductive Learning on Adaptive Graphs, AAAI 2010

Conclusion

- Vapnik established fundamental theories for transduction
- Transductive Support Vector Machine is the first conscientious effort at transduction
- Current efforts in transduction seem to be in graph-based approaches
- Line between semi-supervised learning and transduction is very blurry

Bibliography

- *Support Vector Machine*. Wikipedia. Accessed 17 Mar 2016.
- *VC Dimension*. Wikipedia. Accessed 17 Mar 2016.
- V. Vapnik. "Transductive Inference and Semi-Supervised Learning." In *Semi-supervised Learning*, 1st ed. Cambridge, Mass.: MIT Press, 2006, ch. 24.
- T. Joachims. "Transductive inference for text classification using support vector machines." ICML, 1999.
- P. Talukdar and K. Crammer. "New regularized algorithms for transductive learning." ECML PKDD, 2009.
- Yu et al. "Adaptive hypergraph learning and its application in image classification." IEEE Transactions on Image Processing, vol 21: issue 7, 2012.
- Bresson et al., "Multi-class Transductive Learning Based on ℓ 1 Relaxations of Cheeger Cut and Mumford-Shah-Potts Model," Journal of Mathematical Imaging and Vision, 2013
- Zhang et al., "Transductive Learning on Adaptive Graphs," AAAI 2010