support vector machines

logistic regression:

$$J(\theta) = -[\frac{1}{n}\sum_{i=1}^{n} y^{(i)} log(f(x^{(i)}, \theta)) + (1 - y^{(i)}) log(1 - f(x^{(i)}, \theta))] + \frac{\lambda}{2m}\sum_{j=1}^{m} \theta_j^2$$

We know that $y^{(i)}$ is either 0 or 1. If $y^{(i)} = 1$ then the cost function $J(\theta)$ is incremented by

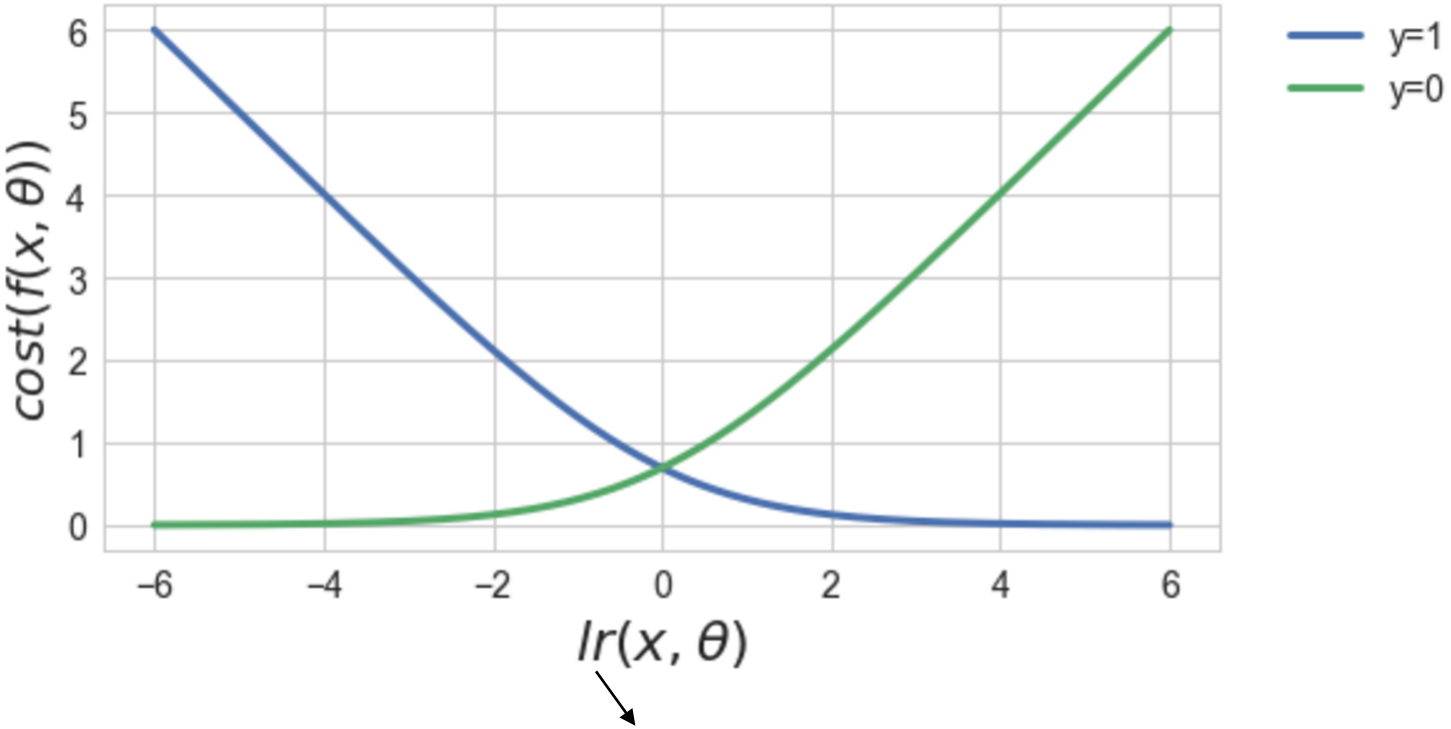
$-log(f(x^{(i)}, \theta)).$

Similarly, if $y^{(i)} = 0$ then the cost function $J(\theta)$ is incremented by

$-log(1 - f(x^{(i)}, \theta)).$

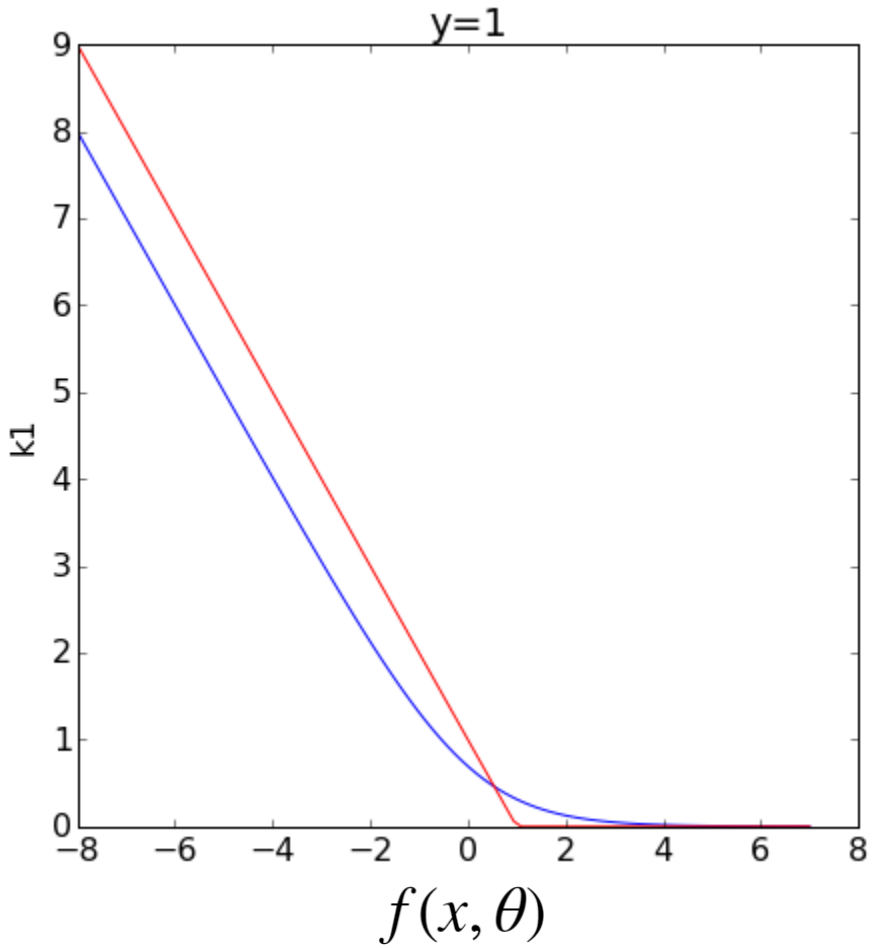
support vector machines

logistic regression:



f()

support vector machines

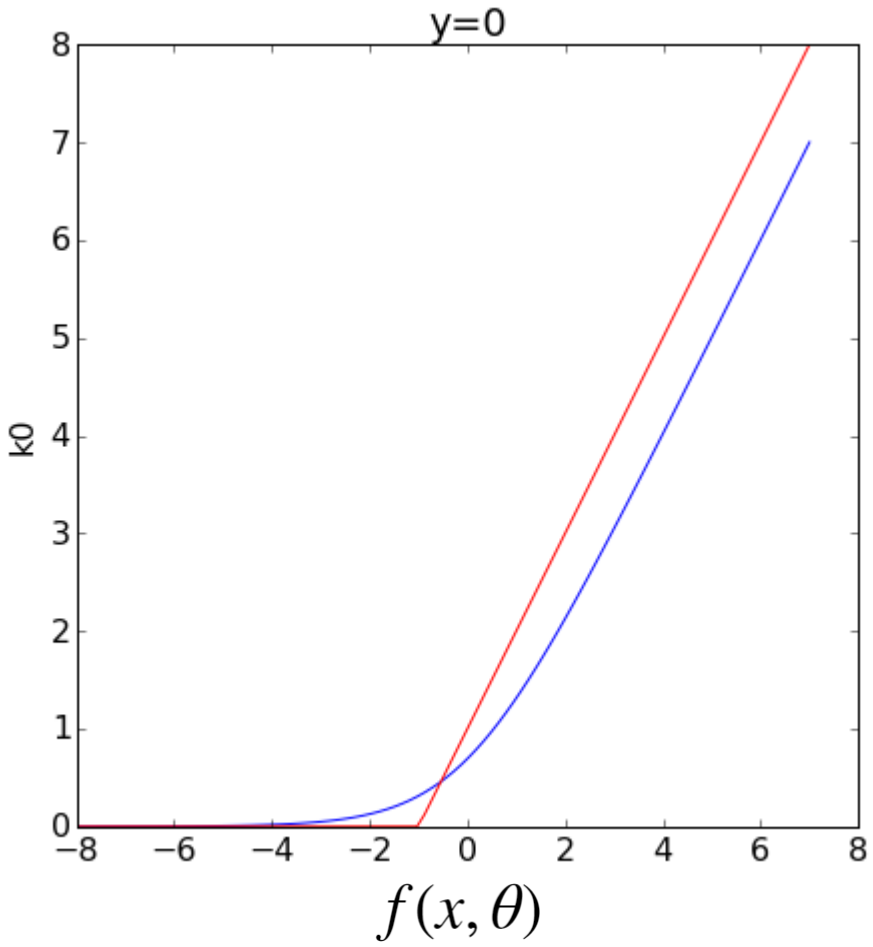


- replace cost function by piecewise linear function

- if *y* = 1 then the contribution to the cost is

$$k_1(f(x,\theta)) = max(0, 1 - f(x,\theta))$$

support vector machines



- replace cost function by piecewise linear function
- if *y* = 1 then the contribution to the cost is

$$k_1(f(x,\theta)) = max(0, 1 - f(x,\theta))$$

- if *y* = 0 then the contribution to the cost is

$$k_0(f(x,\theta)) = max(0, 1 + f(x,\theta))$$

support vector machines

Fit a linear model

$$f(x, \theta) = \theta' x$$

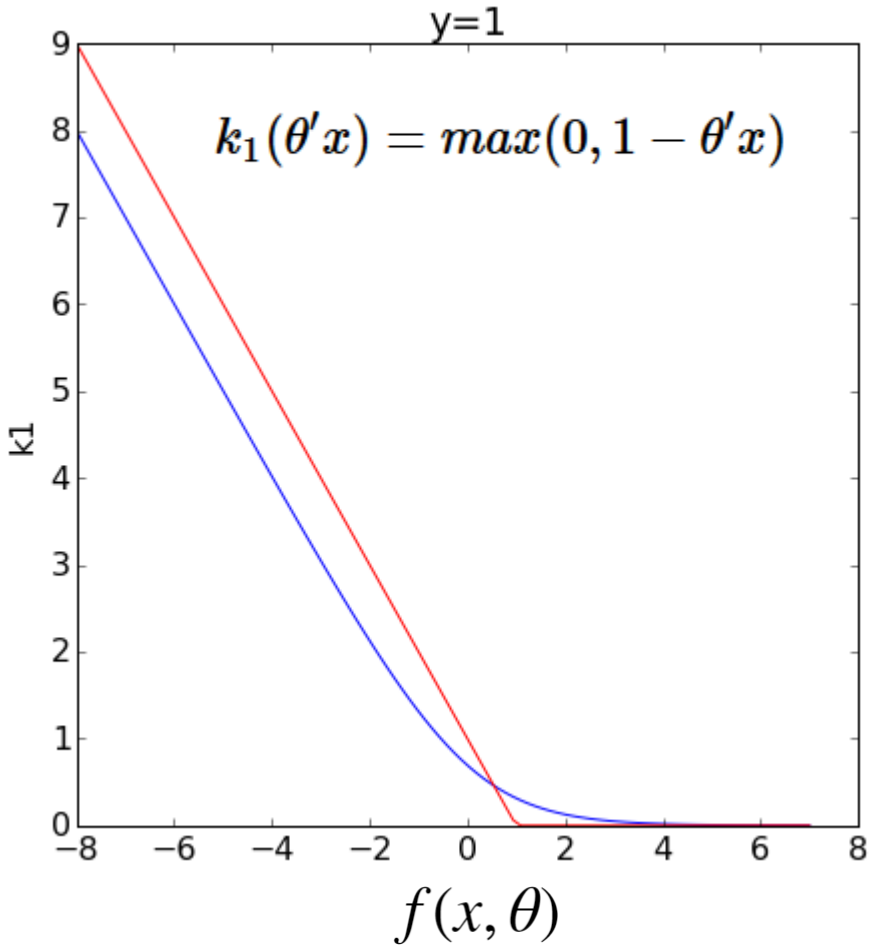
such that

$$J(\theta) = \left[C \sum_{i=1}^{n} y^{(i)} k_1(\theta' x^{(i)}) + (1 - y^{(i)}) k_0(\theta' x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$$

with $k_1(\theta' x) = max(0, 1 - \theta' x)$ and $k_0(\theta' x) = max(0, 1 + \theta' x)$

is minimized.

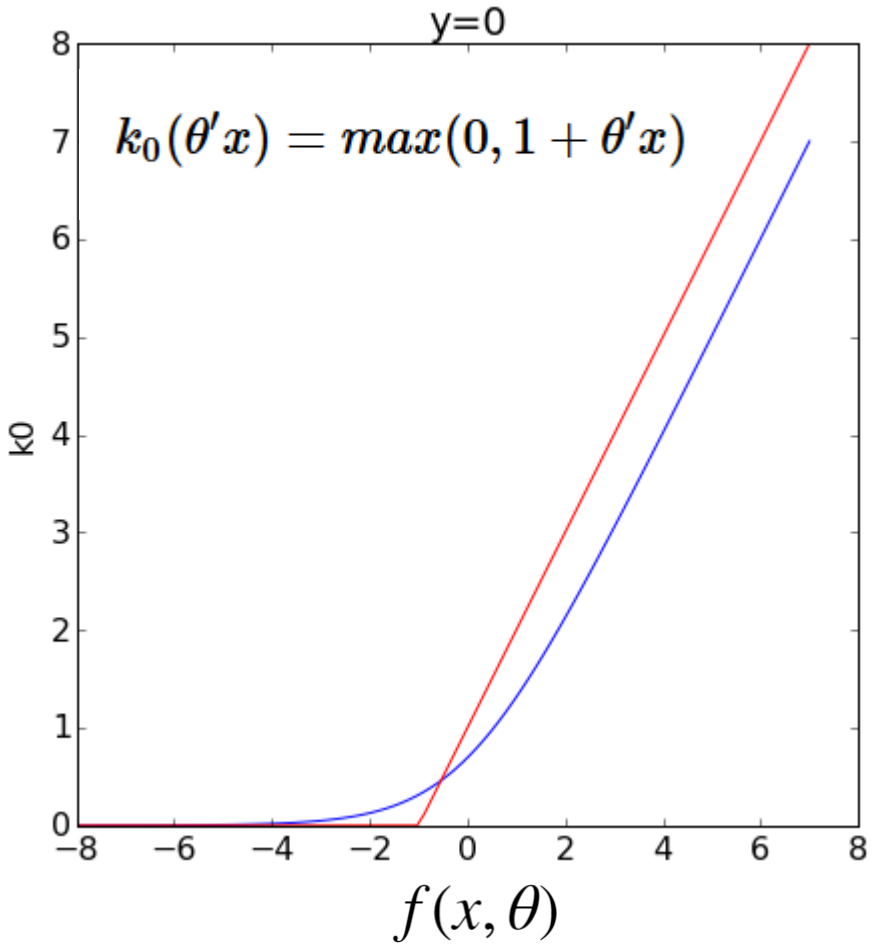
support vector machines



- In this case the contribution to the cost needs to be small when the model predicts high values (>0) and large when the model predicts low values (<0).
- For SVMs we see that the contribution to cost decreases linearly and becomes zero when

$$\theta' x \geq 1$$

support vector machines



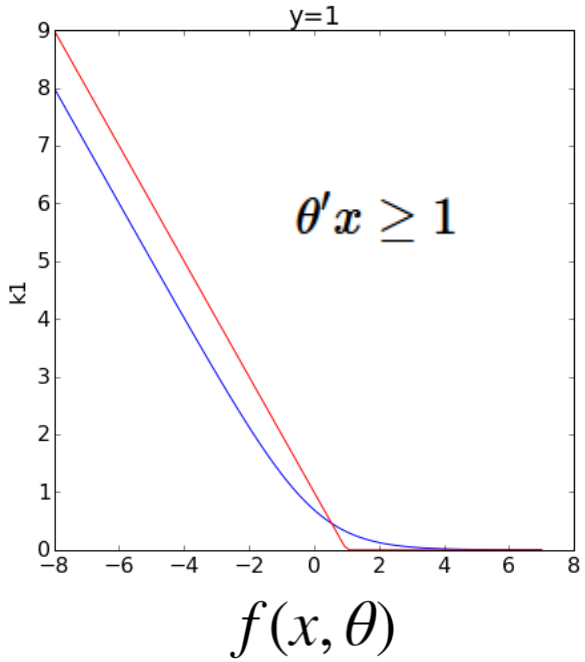
- In this case the contribution to the cost needs to be large when the model predicts high values (>0) and small when the model predicts low values (<0).
- For SVMs we see that the contribution to the cost is zero for

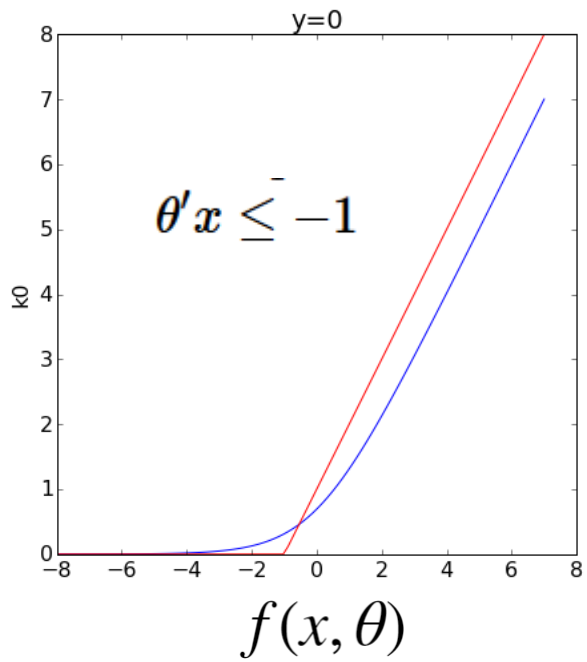
$$\theta' x \leq -1$$

and then increases linearly.

support vector machines

$$J(\theta) = \left[C \sum_{i=1}^{n} y^{(i)} k_1(\theta' x^{(i)}) + (1 - y^{(i)}) k_0(\theta' x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$$





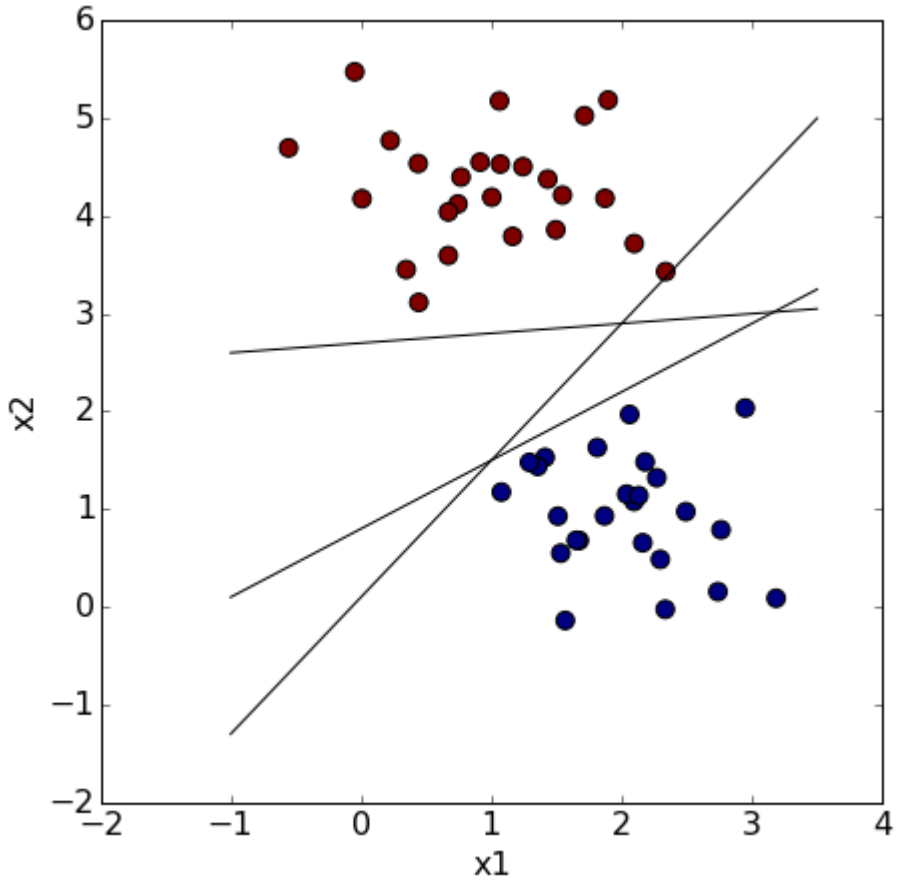
- Consider a train set with two classes that are perfectly linearly separable.
- The piecewise cost function can be made zero.
- The SVM objective can be written as

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$$

$$\text{subject to } \theta' x^{(i)} \geq 1 \text{ if } y^{(i)} = 1$$

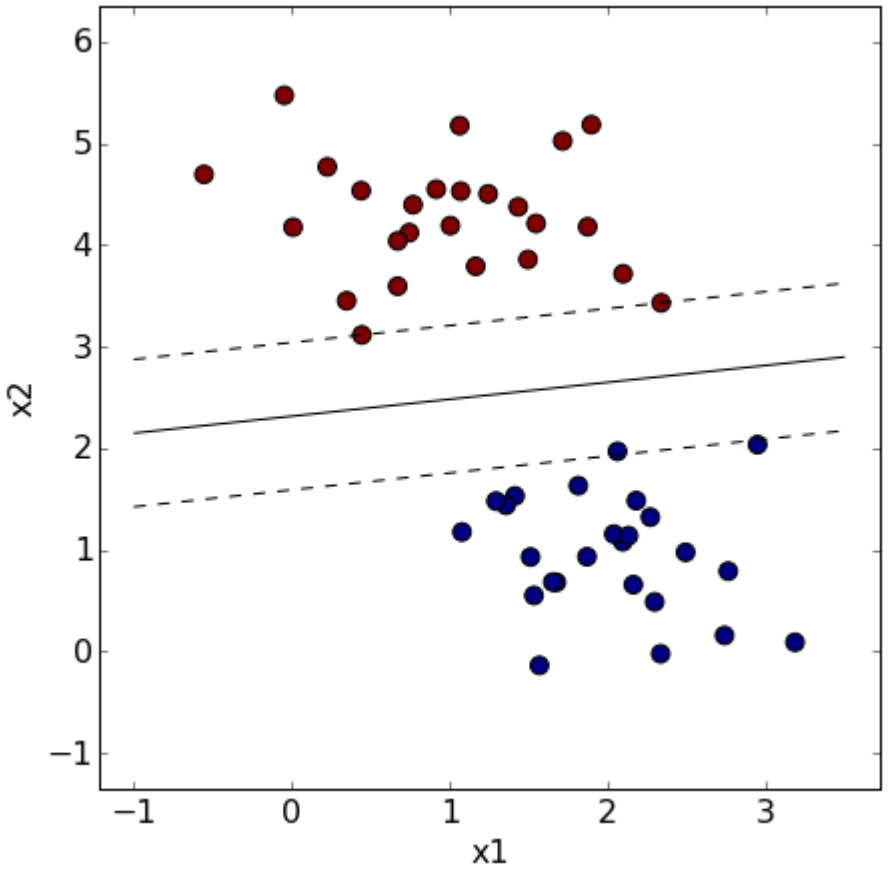
$$\theta' x^{(i)} \leq -1 \text{ if } y^{(i)} = 0$$

support vector machines



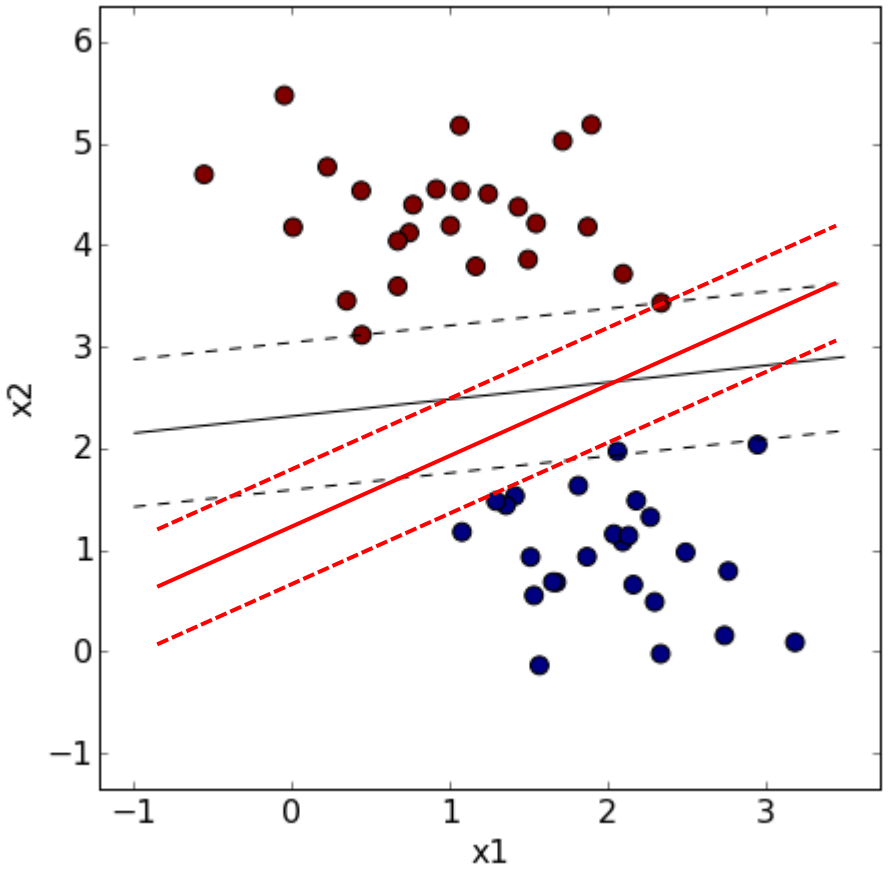
- Consider a train set with two classes that are perfectly linearly separable.
- many possible decision boundaries

support vector machines



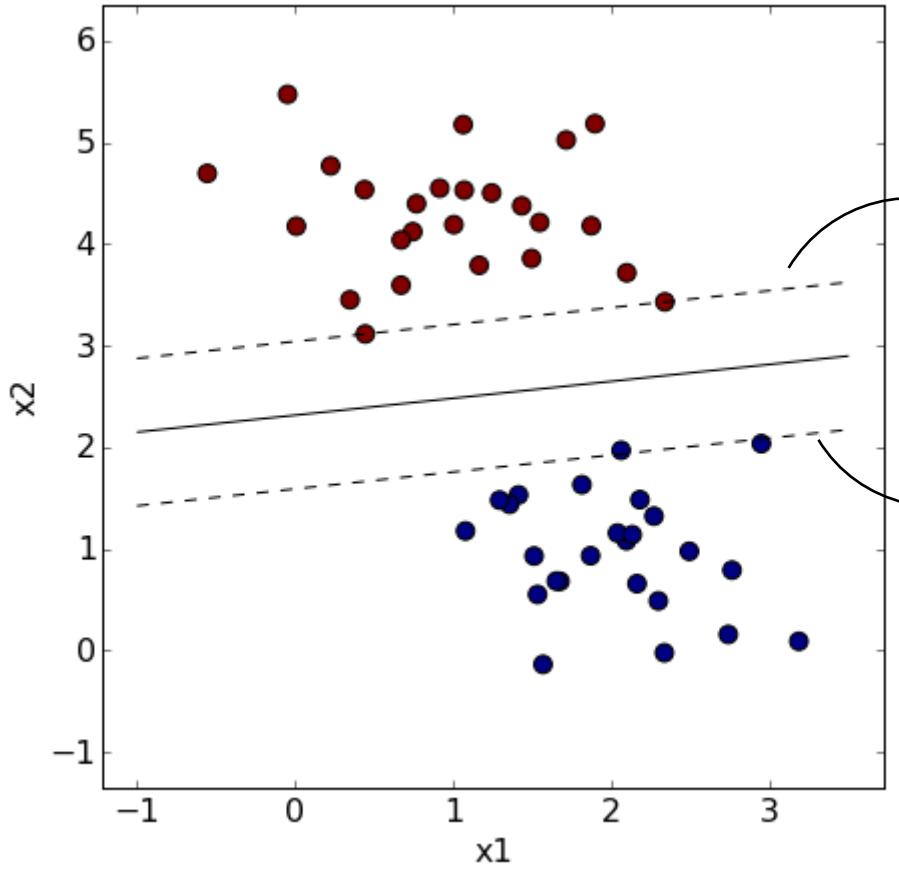
- Consider a train set with two classes that are perfectly linearly separable.
- many possible decision boundaries
- SVM picks to one that maximizes the margin between the classes

support vector machines



- Consider a train set with two classes that are perfectly linearly separable.
- many possible decision boundaries
- SVM picks to one that maximizes the margin between the classes

support vector machines



$$\theta' x^{(i)} = 1$$

$$\theta' x^{(i)} = -1$$

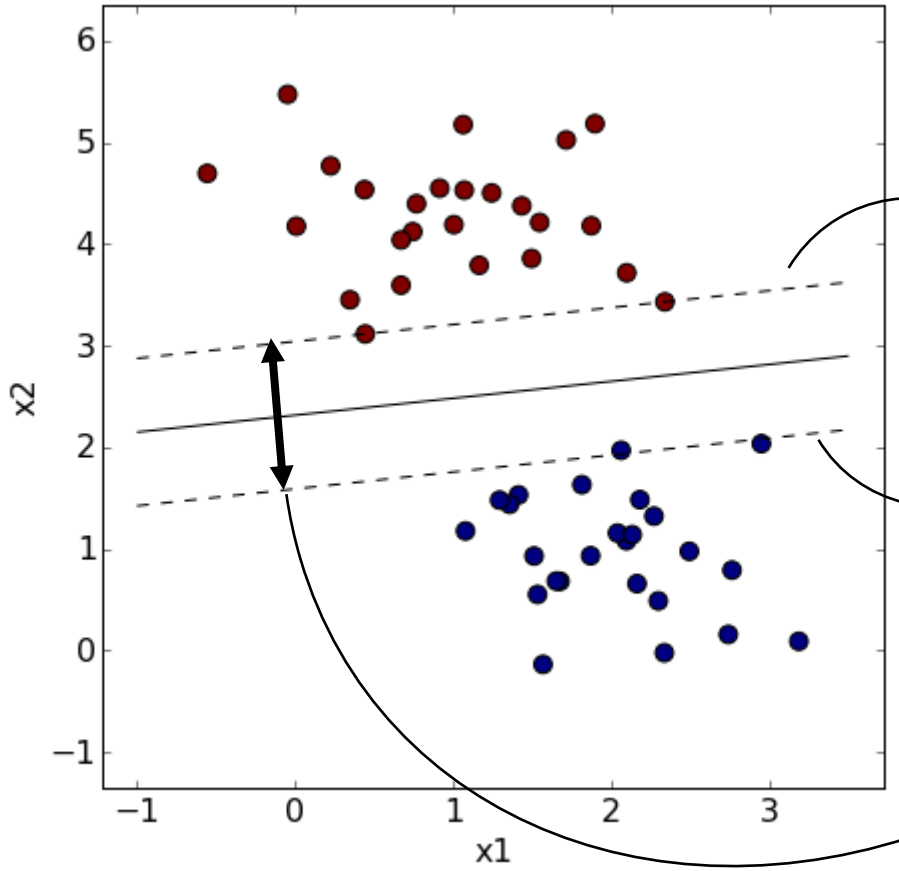
- red points statisfy $\theta' x^{(i)} \geq 1$
- blue points satisfy $\theta' x^{(i)} \leq -1$
- distance bewteen two dashed lines is

$$\frac{2}{\sqrt{\sum_{j=1}^{m} \theta_j^2}}$$

- SVM objective was

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$$

support vector machines



$\theta' x^{(i)} = 1$

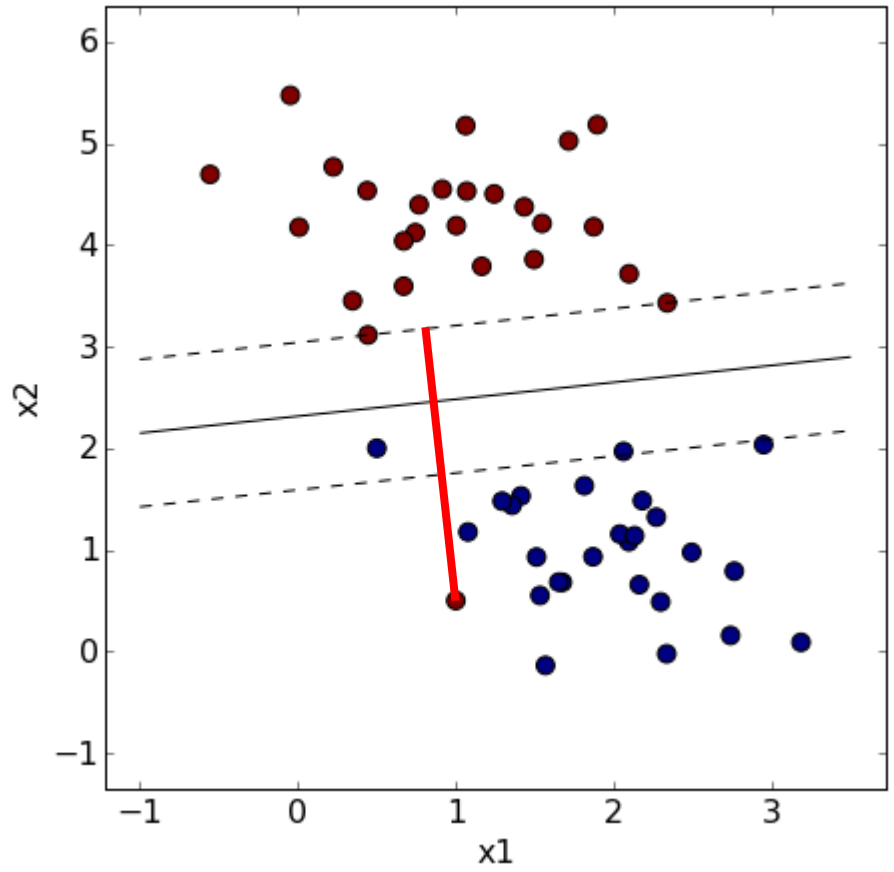
$\theta' x^{(i)} = -1$

margin

- red points statisfy $\theta' x^{(i)} \geq 1$
- blue points satisfy $\theta' x^{(i)} \leq -1$
- distance bewteen two dashed lines is

$$\frac{2}{\sqrt{\sum_{j=1}^{m} \theta_j^2}}$$

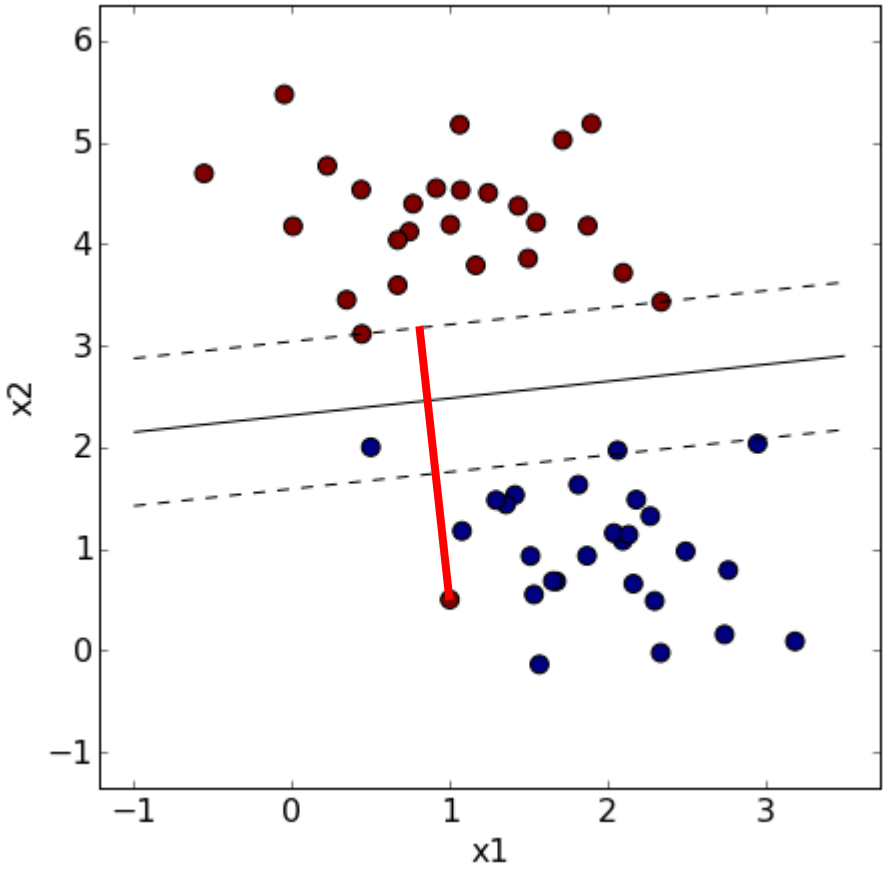
support vector machines



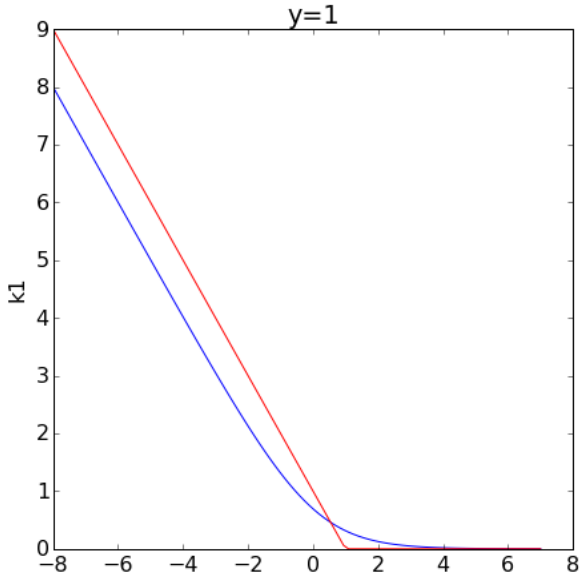
- no model parameters that satisfy

$$\theta' x^{(i)} \geq 1 \text{ if } y^{(i)} = 1$$
$$\theta' x^{(i)} \leq -1 \text{ if } y^{(i)} = 0$$

support vector machines

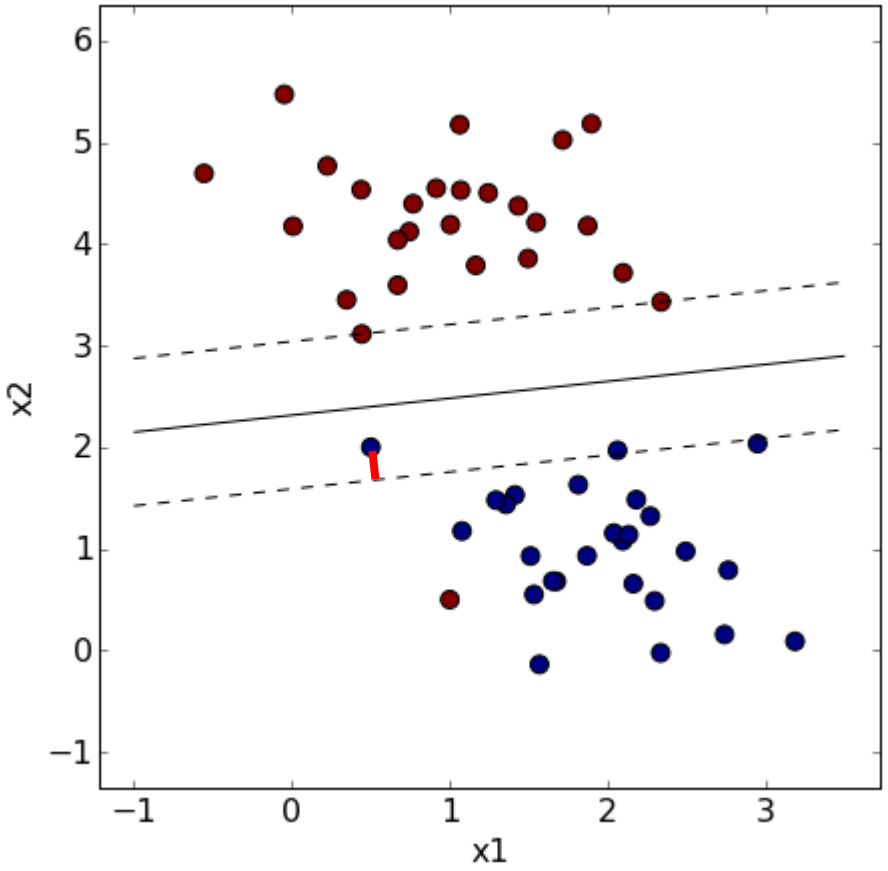


- For misclassified red points (*y*=1) the contribution to the cost increases linearly with the distance from the upper dashed line.

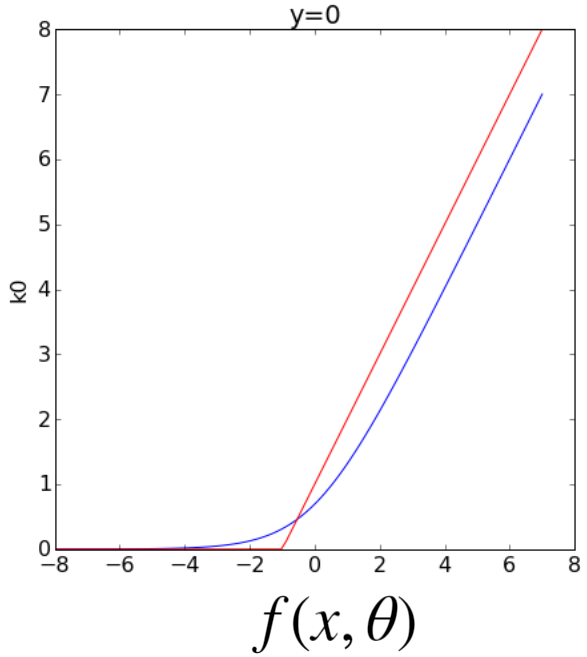


$f(x, \theta)$

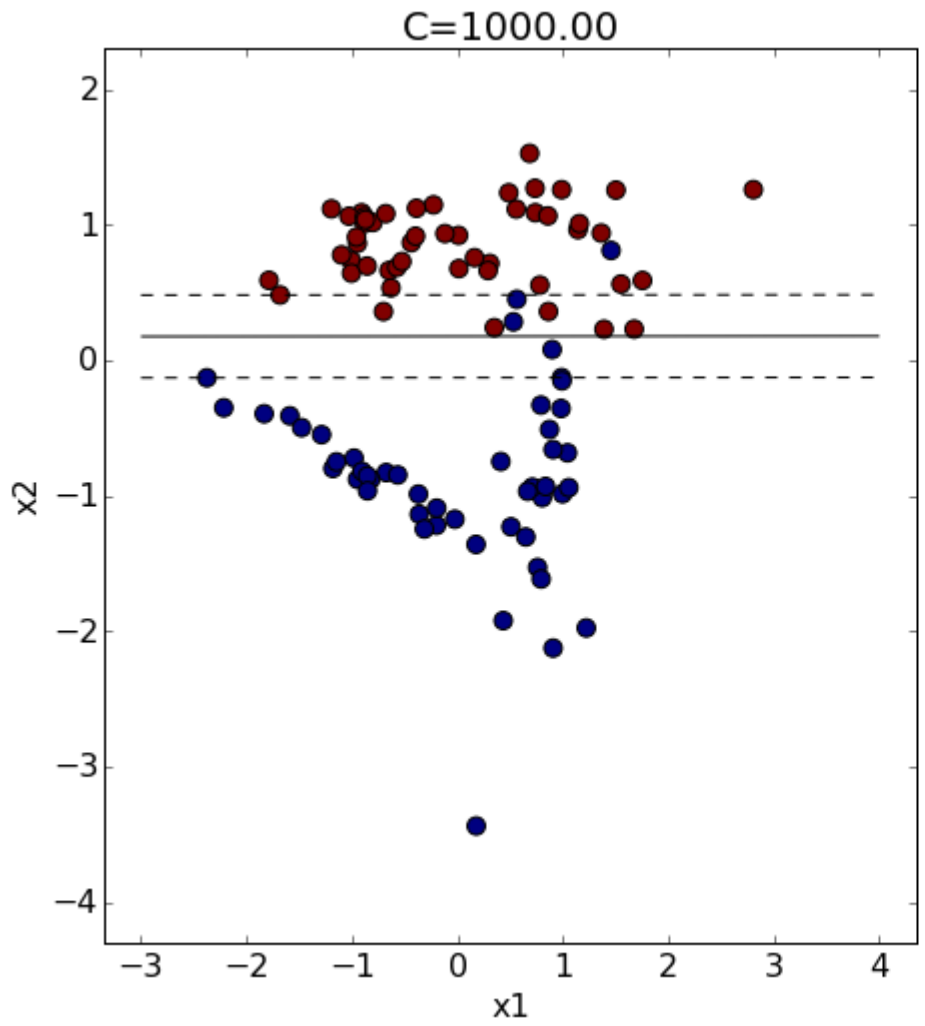
support vector machines

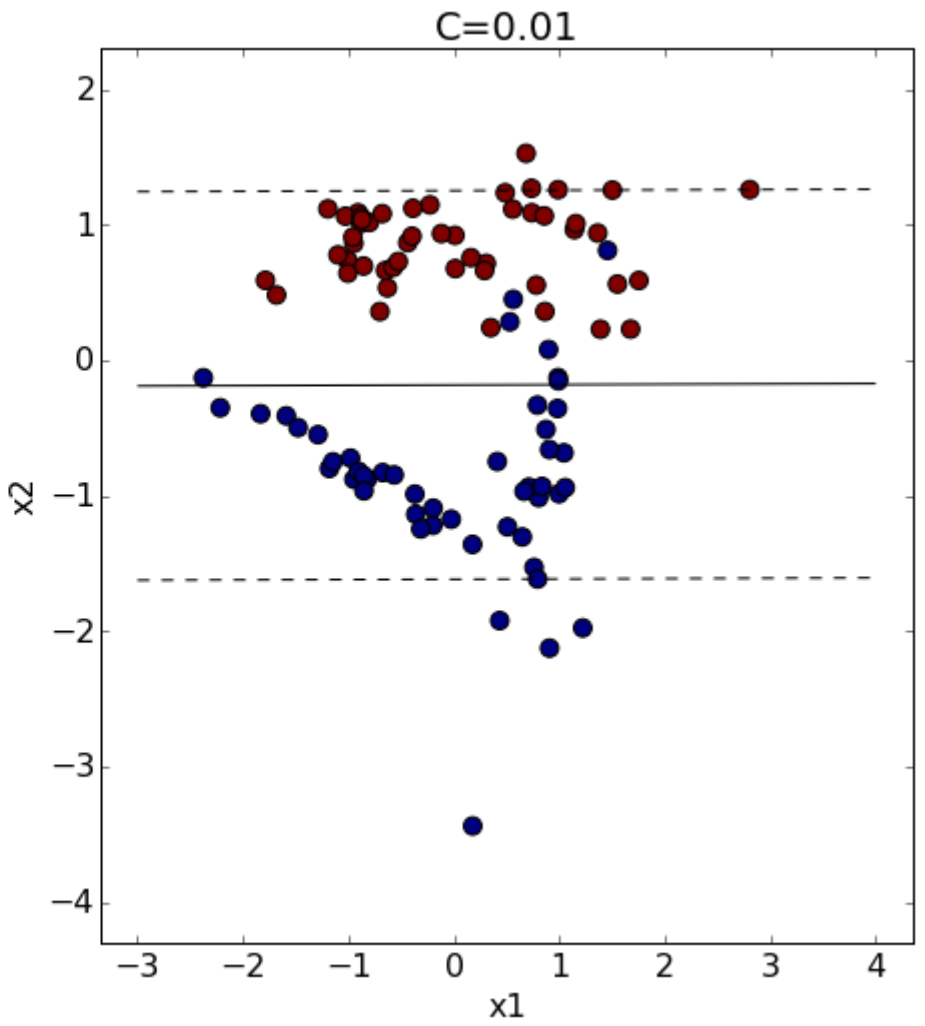


- For misclassified blue points (*y*=0) the contribution to the cost increases linearly with the distance from the lower dashed line.

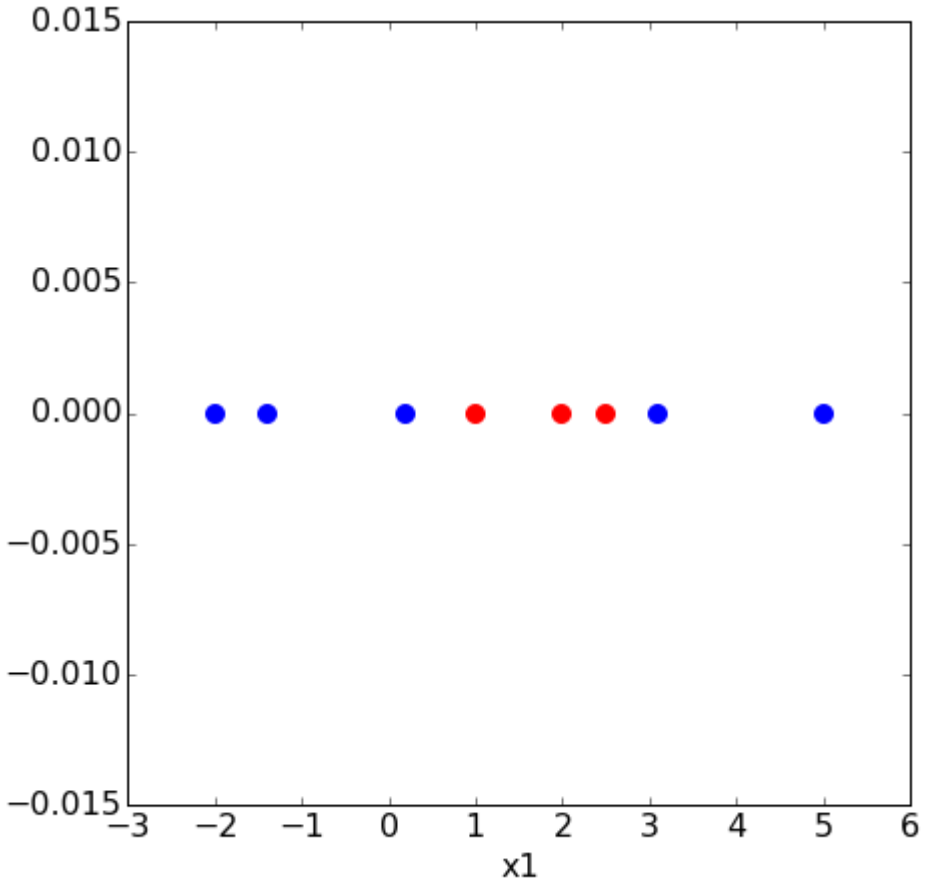


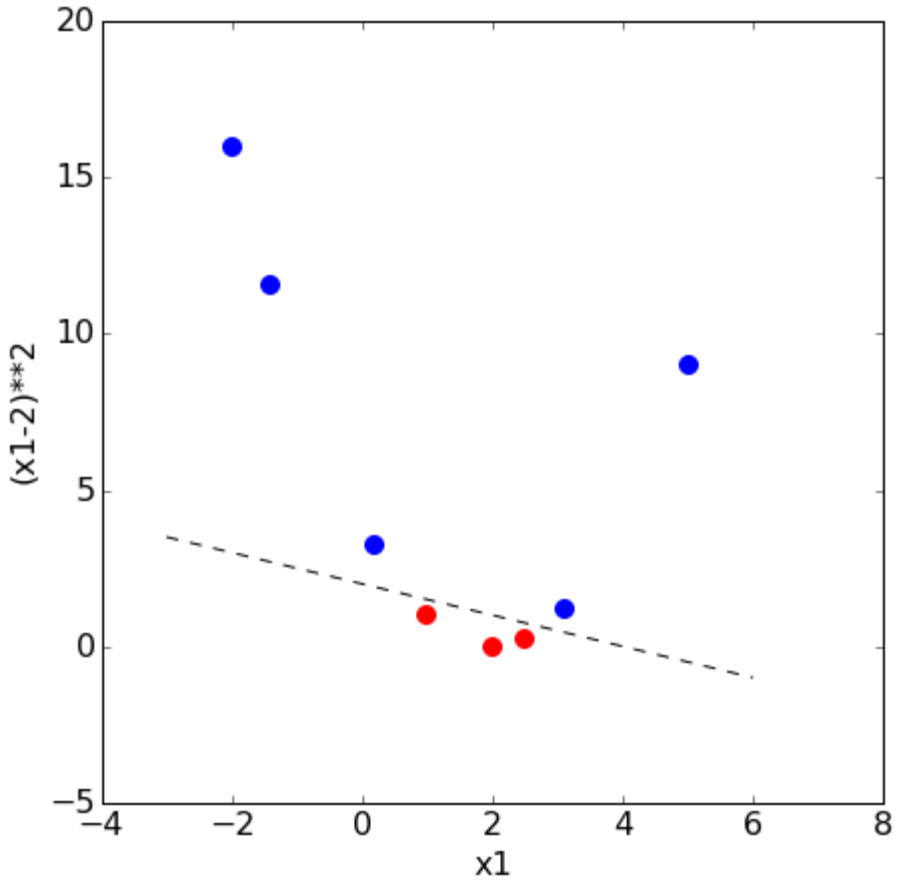
support vector machines





kernel support vector machines





kernel support vector machines

SVMs can also be formulated as a linear function of the samples (dual form) instead of the features as

$$f(x, \theta) = \sum_{i=1}^{n} \theta_i (x \cdot x^{(i)}) + \theta_0$$

that can be reformulated as a non-linear function using what is know as a kernel function

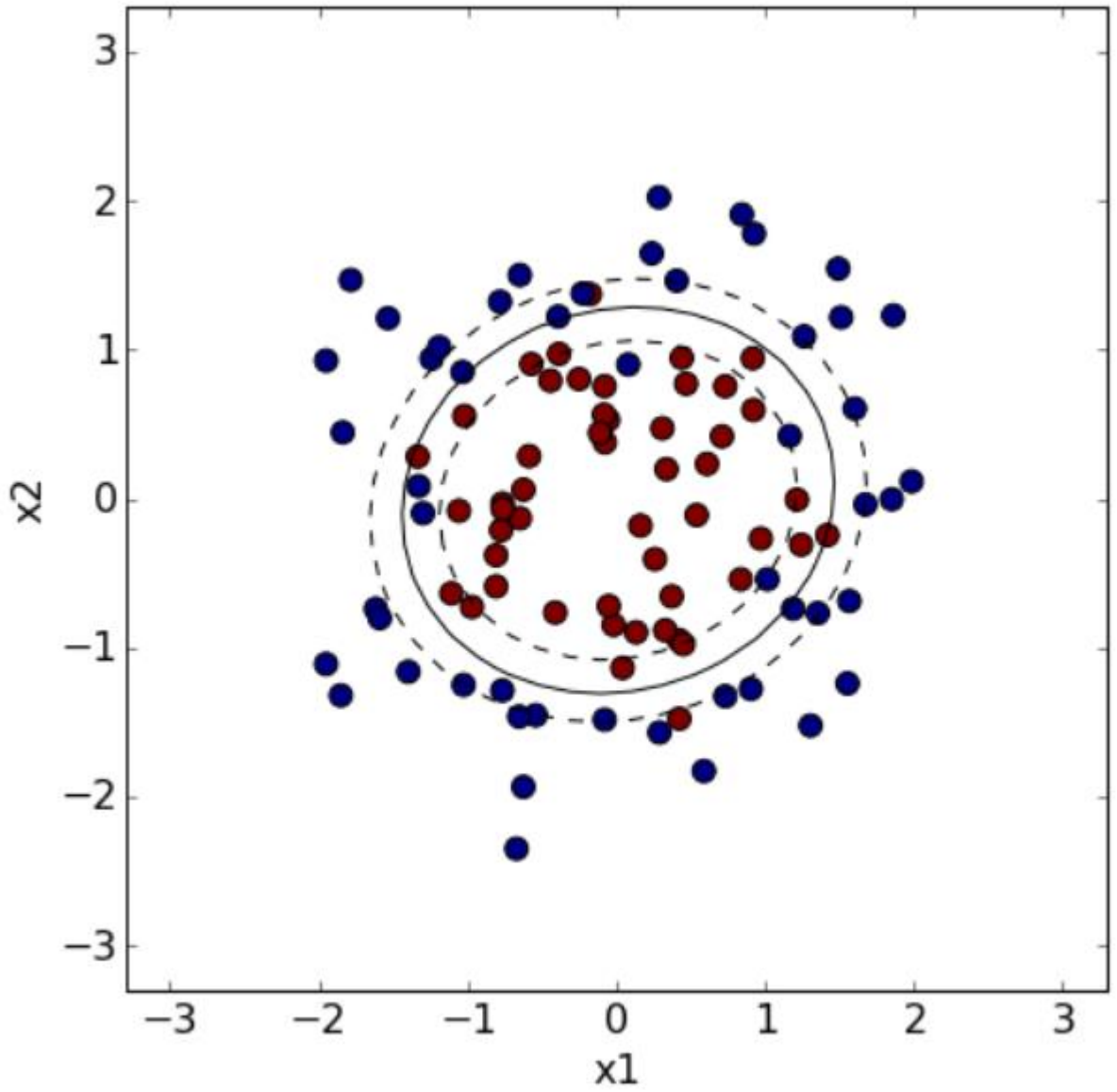
$$K(x^{(i)}, x^{(j)})$$

to become

$$f(x, \theta) = \sum_{i=1}^{n} \theta_i K(x, x^{(i)}) + \theta_0$$

The data points $x^{(i)}$ for which $\theta_i > 0$ are called the support vectors.

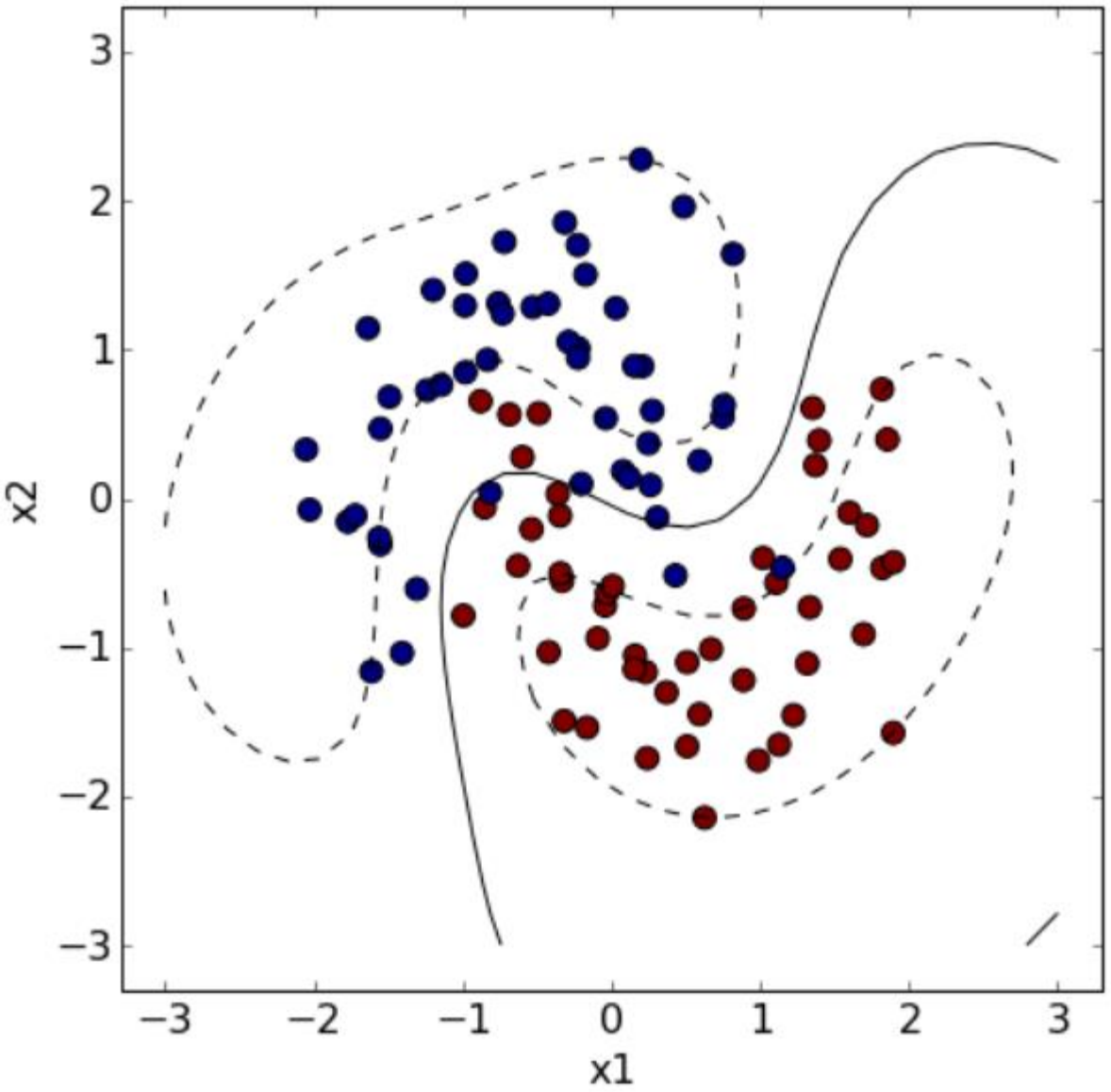
kernel support vector machines



$$f(x,\theta) = \sum_{i=1}^{n} \theta_i K(x, x^{(i)}) + \theta_0$$

$$K(x^{(i)}, x^{(j)}) = (x^{(i)} \cdot x^{(j)} + c)^d$$

kernel support vector machines



$$f(x,\theta) = \sum_{i=1}^{n} \theta_i K(x, x^{(i)}) + \theta_0$$

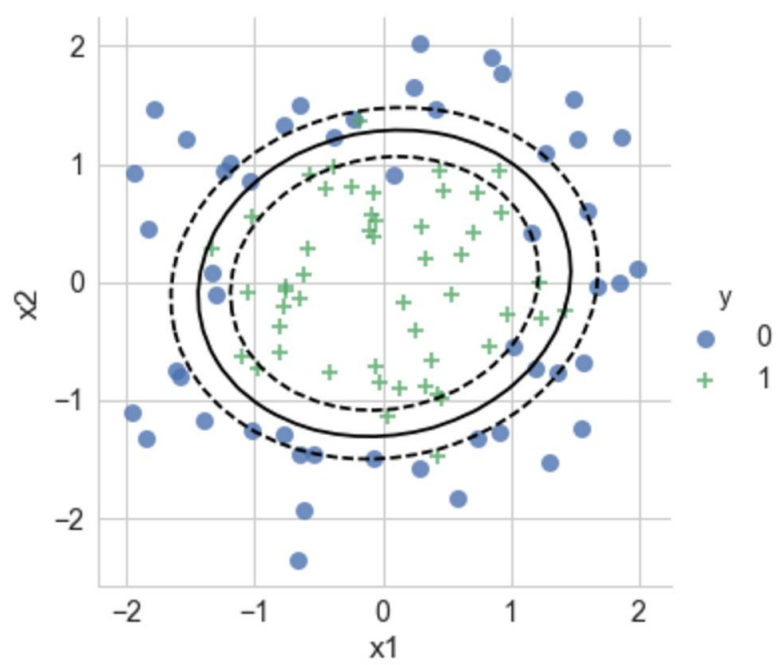
$$K(x^{(i)}, x^{(j)}) = \exp\left[-\frac{\|x^{(i)} - x^{(j)}\|^2}{2\sigma^2}\right]$$

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline

dataset1 = pd.read_csv('svm_example1.csv')
X = dataset1.copy()
y = X.pop('y')

clf = SVC(kernel='linear',C=1)
polynomial_features = PolynomialFeatures(degree=2)
model = Pipeline([("polynomial_features", polynomial_features),
                  ("SVC", clf)])
model.fit(X,y)

sns.lmplot(x="x1", y="x2", data=dataset1, hue='y', markers=['o','+'],
           fit_reg=False, size=5, scatter_kws={"s": 80})
compomics_import.plot_svm_decision_function(model)
plt.show()
```



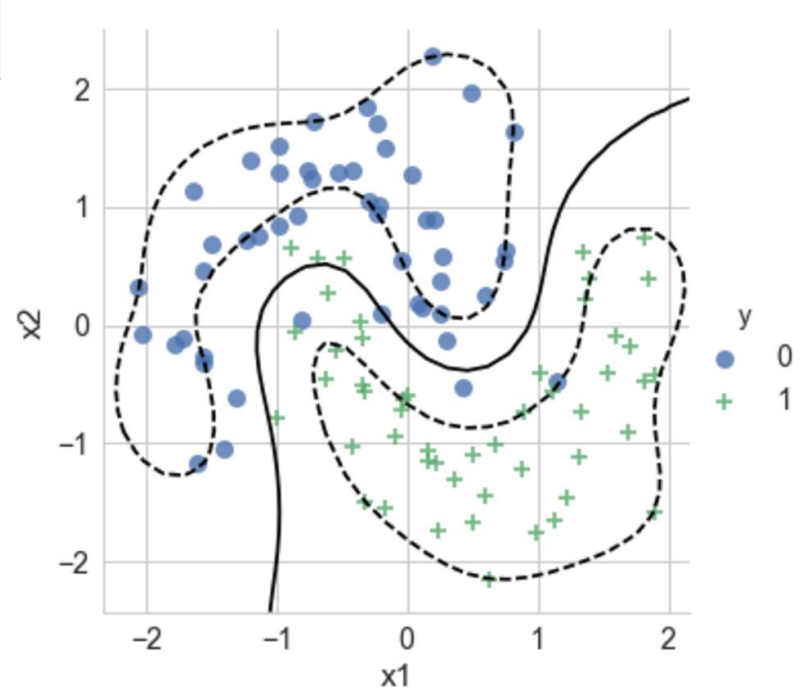

```
model = SVC(kernel='poly',C=1,degree=2)
model.fit(X,y)

sns.lmplot(x="x1", y="x2", data=dataset1, hue='y', markers=['o','+'],
           fit_reg=False, size=5, scatter_kws={"s": 80})
compomics_import.plot_svm_decision_function(model)
plt.show()
```

```
dataset2 = pd.read_csv("svm_example2.csv")
X = dataset2.copy()
y = X.pop('y')

model = SVC(kernel='rbf',C=1,gamma=1)
model.fit(X,y)

sns.lmplot(x="x1", y="x2", data=dataset2, hue='y', markers=['o','+'],
           fit_reg=False, size=5, scatter_kws={"s": 80})
compomics_import.plot_svm_decision_function(model)
plt.show()
```



```
from sklearn import cross_validation
from sklearn.grid_search import GridSearchCV

search_space = np.logspace(-10, 14, 10, base=2)

params = dict(C=search_space)
grid_search = GridSearchCV(model, param_grid=params)

grid_search.fit(X, y)

for params, mean_score, scores in grid_search.grid_scores_:
    print("%0.3f (+/-%0.03f) for %r" % (mean_score, scores.std() * 2, params))
```

splice site prediction

CGTGTTGTCGCAACATCGTGCGTGACGGACTTGCGTAGCCTCCGACGTGTCAACGCGTACCACGTGCGTGGT

Degroeve,S. et al. (2005) Predicting splice sites from high-dimensional local context representations. Bioinformatics, 21, 1332–1340

splice site prediction

ACTTCGGTAGCCTCC

Positional Information

The information extracted is the presence or absence of a nucleotide at a position in the local context subsequence. We will refer to this feature set as P1. Let $f_{s,v}$ be a binary feature from P1 that has value one if the nucleotide at position s in the local context sequence is v with $v \in \{a, c, g, t\}$. For the candidate donor site (see example) the following features in P1 have a value equal to one (all other features have value zero):
$f_{1,a}$, $f_{2,c}$, $f_{3,t}$, $f_{4,t}$, $f_{5,c}$, $f_{6,g}$, $f_{7,a}$, $f_{8,g}$, $f_{9,c}$, $f_{10,c}$, $f_{11,t}$, $f_{12,c}$ and $f_{13,c}$.

To account for the correlations that exist between nucleotide positions certain types of concatenations are created using the features in P1. Feature sets P2 and P3 extract the presence or absence of a di- or tri-nucleotide at a position in the local context subsequence. Let v be a di-nucleotide then for the feature set P2 (similarly for the feature set P3) the following features have a value equal to one:
$f_{1,ac}$, $f_{2,ct}$, $f_{3,tt}$, $f_{4,tc}$, $f_{5,cg}$, $f_{7,ag}$, $f_{8,gc}$, $f_{9,cc}$, $f_{10,ct}$, $f_{11,tc}$, $f_{12,cc}$.

splice site prediction

ACTTCG<u>GT</u>AGCCTCC

Compositional Information

The information extracted is the presence or absence of individual tri-, tetra-, penta- or hexa-mers in the local upstream context and in the local downstream context separately. We will refer to these feature sets as Ck where k is the length of the oligomer (k-mer) that is considered. Each feature set Ck consists of 4^k features $f_{up,x}$ that indicate the presence or absence of a k-mer x in the upstream context plus 4^k features $f_{down,x}$, that indicate the presence or absence of a k-mer x in the downstream context. For the candidate donor example the following compositional trimer features (C3) have a value equal to one:

$f_{up,act}$, $f_{up,ctt}$, $f_{up,ttc}$, $f_{up,tcg}$, $f_{down,agc}$, $f_{down,gcc}$, $f_{down,cct}$, $f_{down,ctc}$ and $f_{down,tcc}$.

splice site prediction

ACTTCGGTAGCCTCC

Coding Potential

The information extracted is the presence or absence of codons in each of the three possible reading frames of both local upstream and downstream context separately. Let us assume that the complete context of our candidate site is a coding sequence, i.e. the candidate site would be a pseudo splice site. We can then write this context sequence in each of the three possible reading frames. For each of the three reading frames R ($R = 1, 2, 3$) 64 features $f_{R,up,x}$ are computed for the upstream context plus 64 features $f_{R,down,x}$ for the downstream context. A feature $f_{R,up,x}$ has value one if the upstream context in reading frame R contains the trimer or codon x. This totals $128 \times 3 = 384$ features in a set we will denote RF (for Reading Frame). For the candidate donor site presented above the following features in RF have a value equal to one:

$f_{1,up,act}$, $f_{1,up,tcg}$, $f_{2,up,ctt}$, $f_{3,up,ttc}$, $f_{1,down,agc}$, $f_{1,down,ctc}$, $f_{2,down,gcc}$, $f_{2,down,tcc}$ and $f_{3,down,cct}$.

BIOINFORMATICS

Vol. 19 no. 15 2003, pages 1964–1971
DOI: 10.1093/bioinformatics/btg265

Sequence alignment kernel for recognition of promoter regions

Leo Gordon[1,*], Alexey Ya. Chervonenkis[1,2], Alex J. Gammerman[1], Ilham A. Shahmuradov[1] and Victor V. Solovyev[3]

[1]Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK, [2]Institute of Control Science, Profsoyuznaja 65, Moscow, Russia and [3]Softberry Inc., 116 Radio Circle, Suite 400, Mount Kisco, NY, 10549, USA

Received on December 6, 2002; revised on March 2, 2003; accepted on April 29, 2003

ABSTRACT

In this paper we propose a new method for recognition of prokaryotic promoter regions with startpoints of transcription. The method is based on Sequence Alignment Kernel, a function reflecting the quantitative measure of match between two sequences. This kernel function is further used in Dual SVM, which performs the recognition.

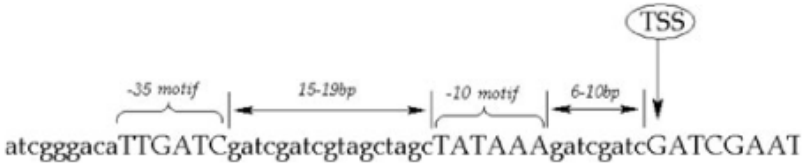


Fig. 1. Prokaryotic σ^{70} promoter region with TSS, '−10' and '−35' binding motifs and two spacers.