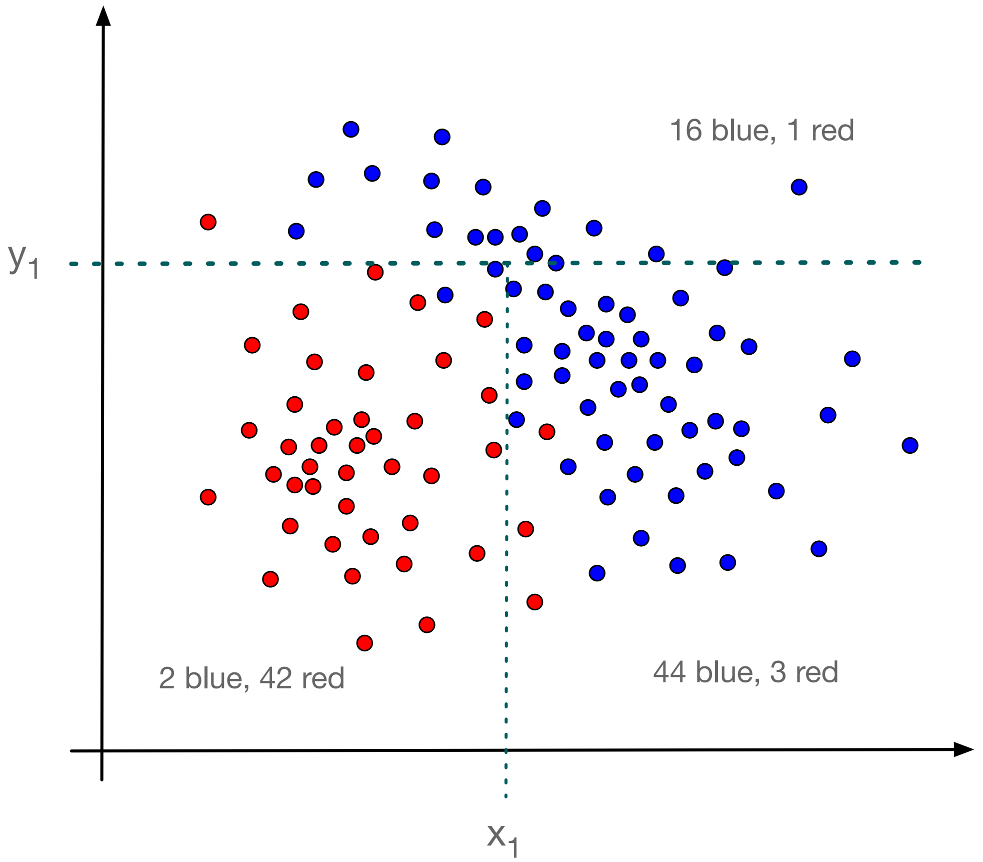
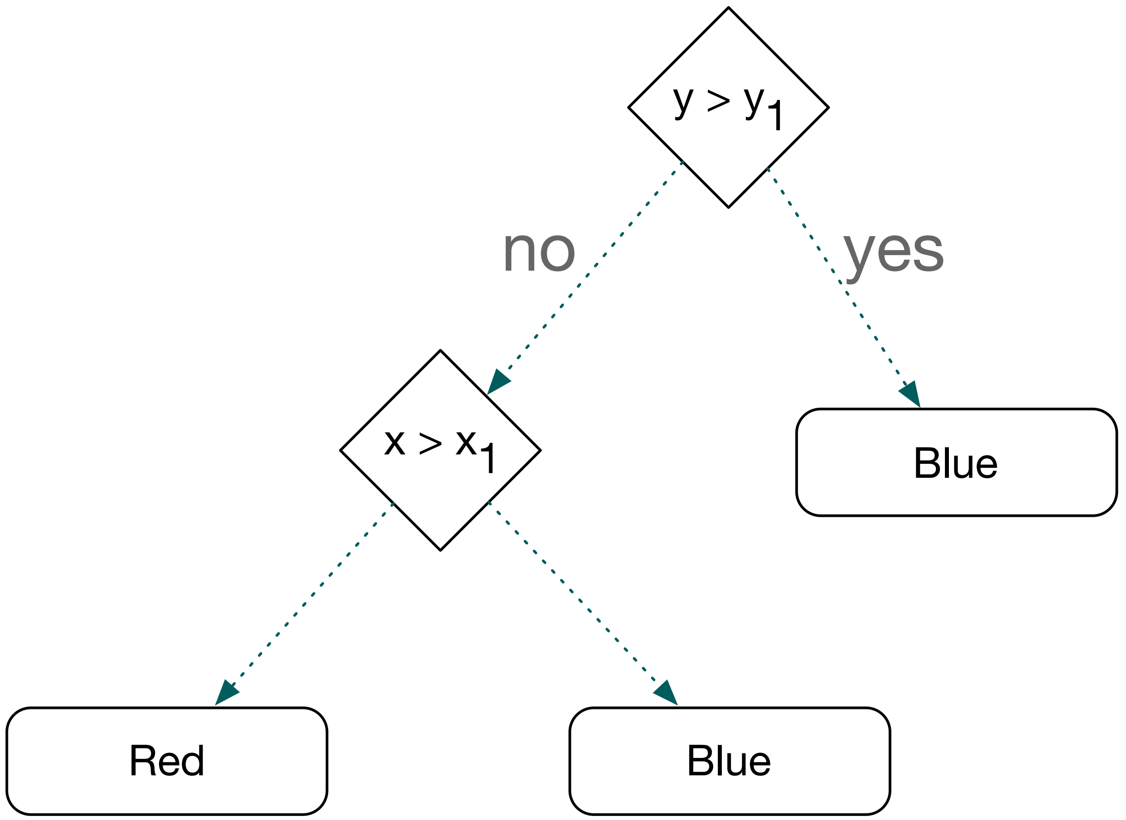
Support Vector Machines

Thomas Schwarz

Introduction

- Recall Decision Trees
  - Use the features in order to make recursively decisions
  - End up with a classification





Introduction

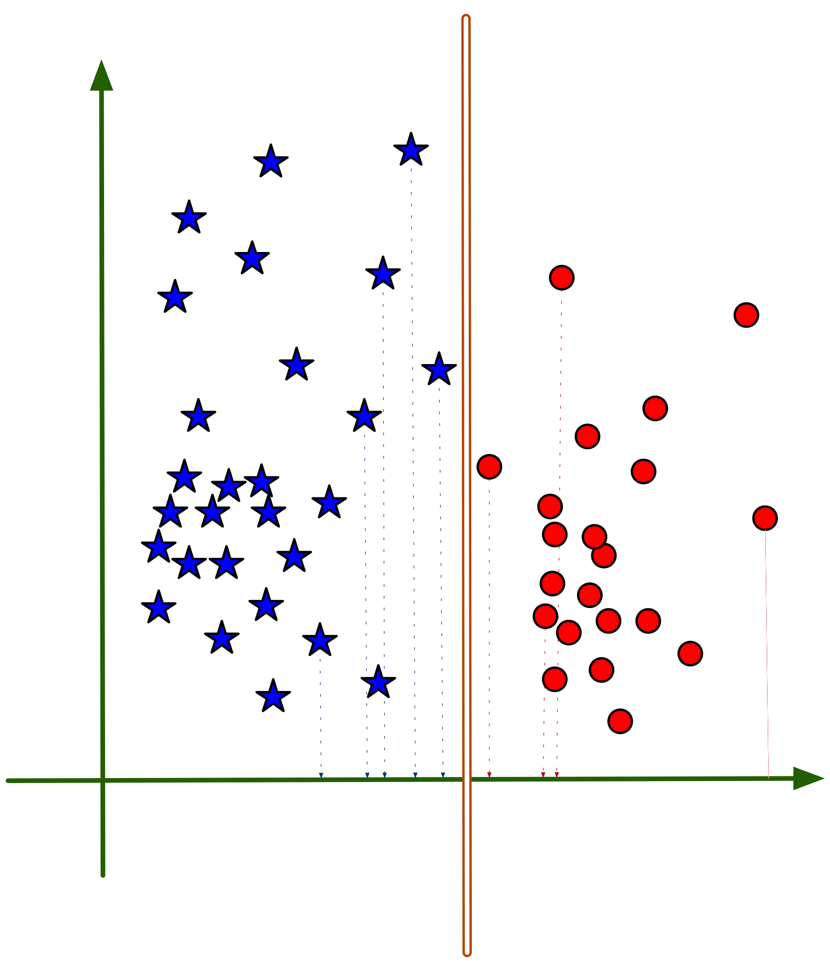
- Decision trees are a type of supervised learning
  - Limited by using only a single feature

Introduction

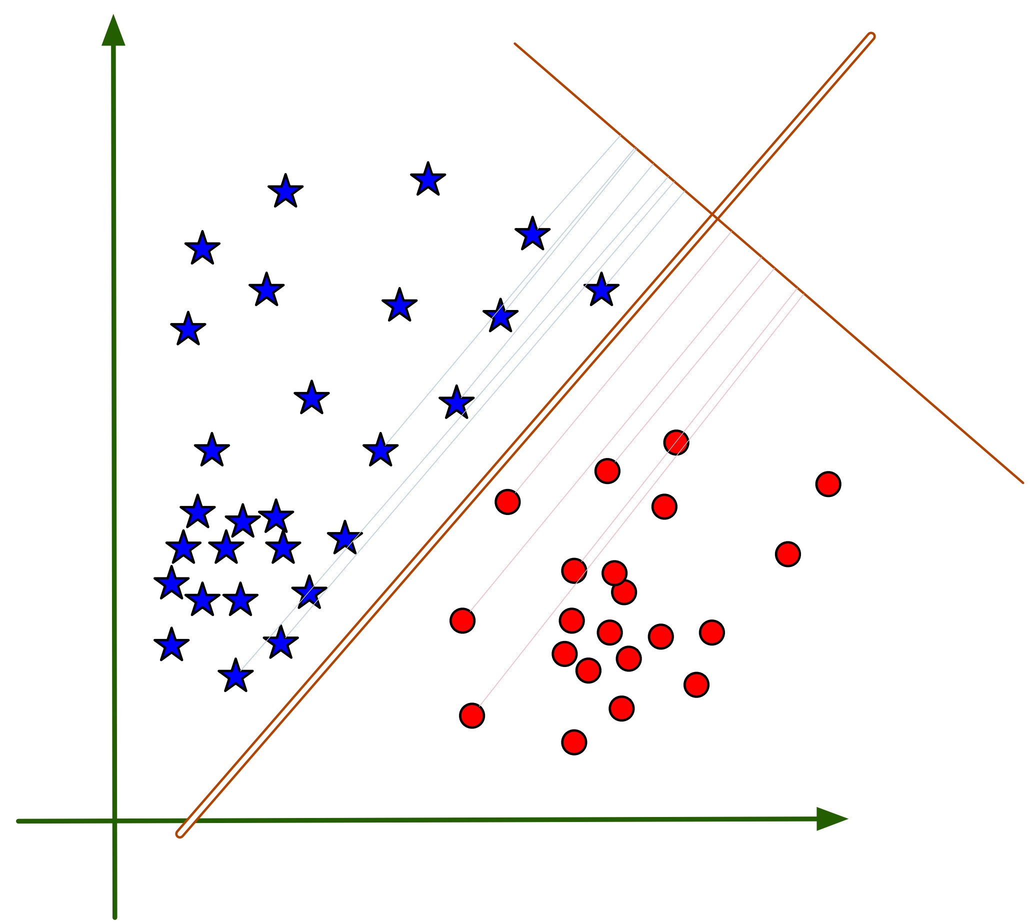
- KNN — *k* nearest neighbors
  - Supervised learning
    - to classify a feature point:
      - we look at all elements in the training set
  - Drawback: Not scalable if training set is large

Introduction

- SVM:
  - Uses ***any*** hyperplane to separate sets



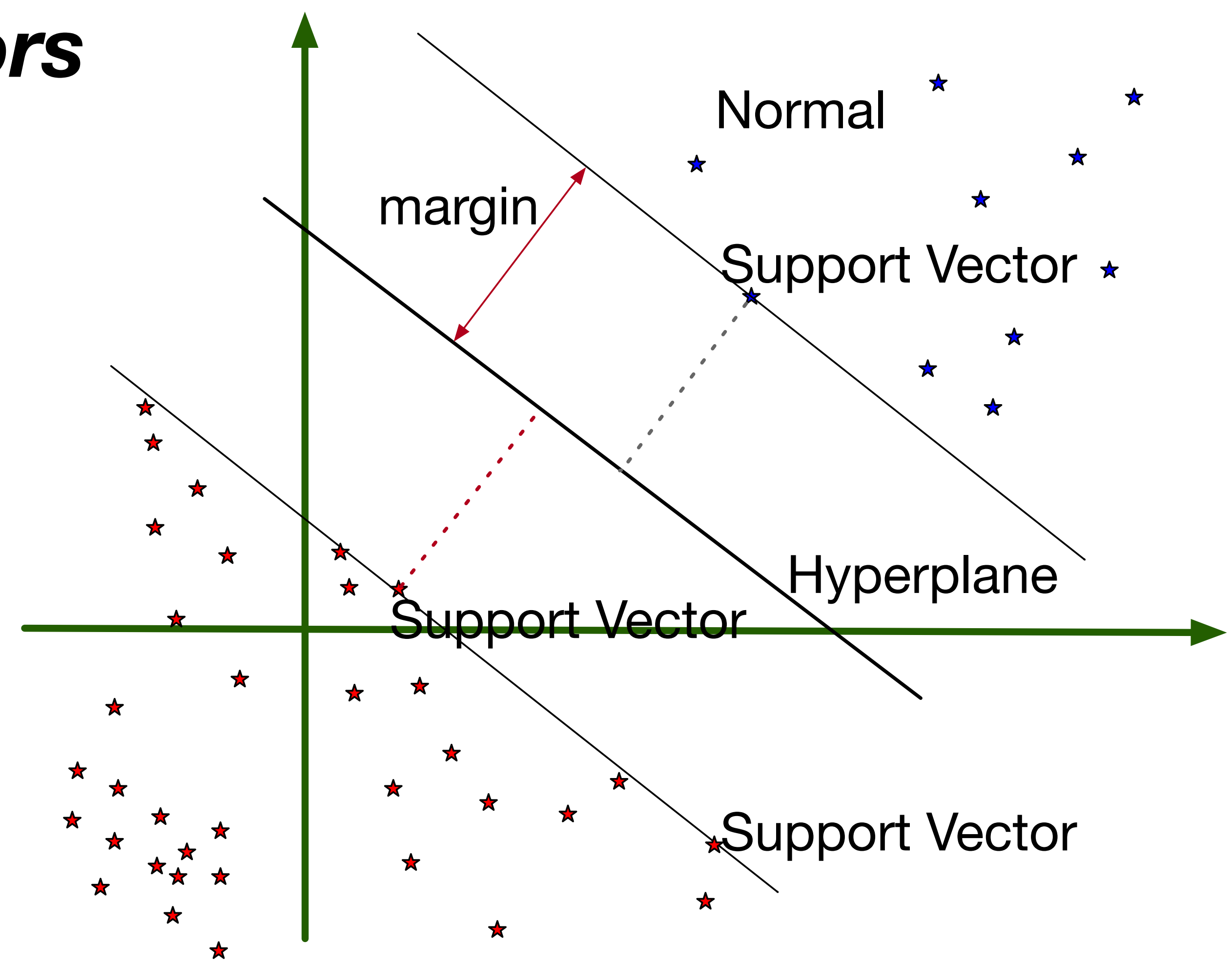
Decision tree boundaries are
parallel to axes



SVM boundaries can
any hyper-planes

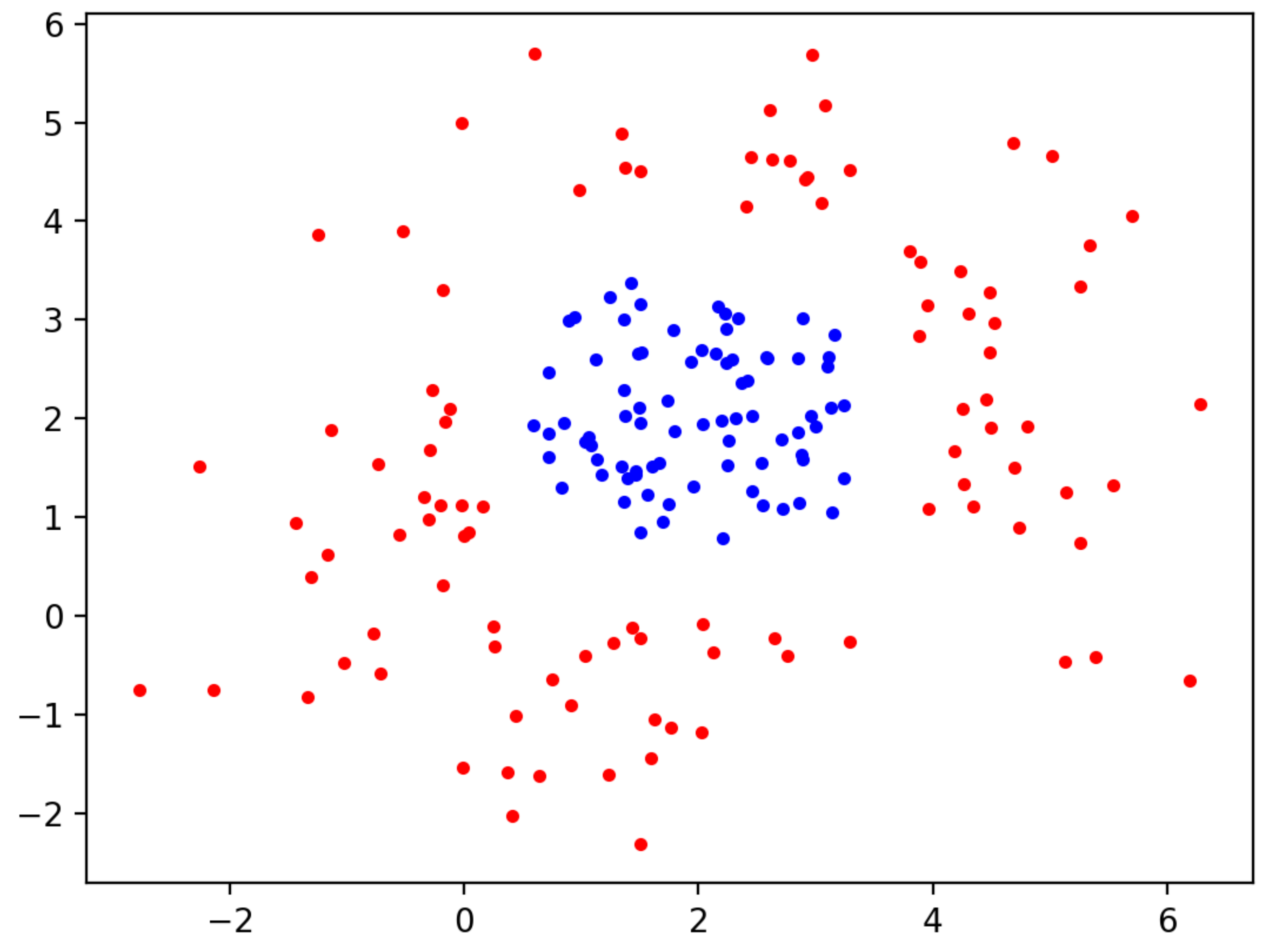
Introduction

- SVM:
  - Can encode the hyper-plane using a few data points:
    - The ***support vectors***



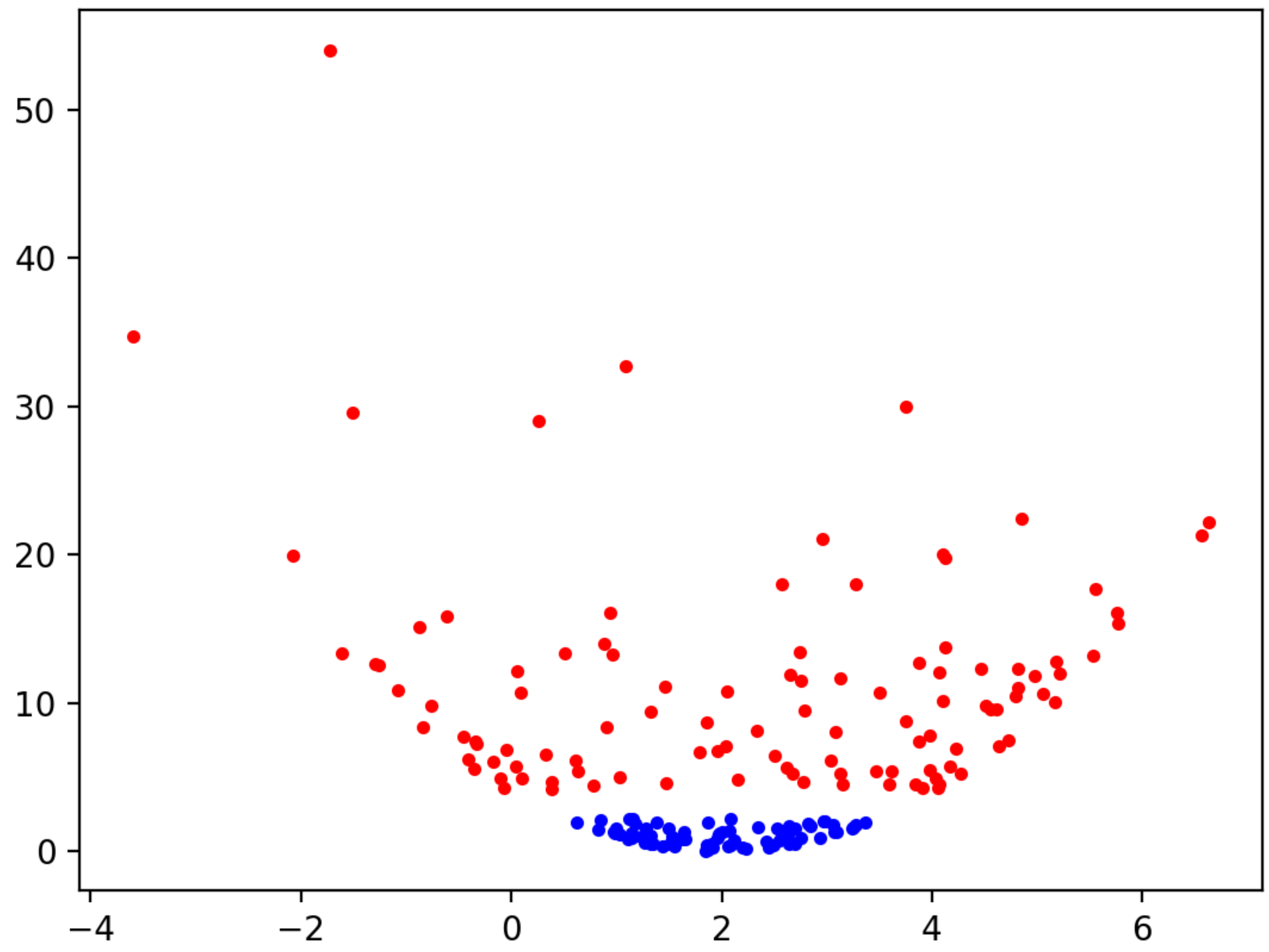
Introduction

- SVM:
  - Can solve classification even if the data set is not linearly separable



Introduction

- Use a (non-linear transform of the data)
    - Kernel function: $(x, y) \mapsto (x, (x-2)^2 + (y-2)^2)$

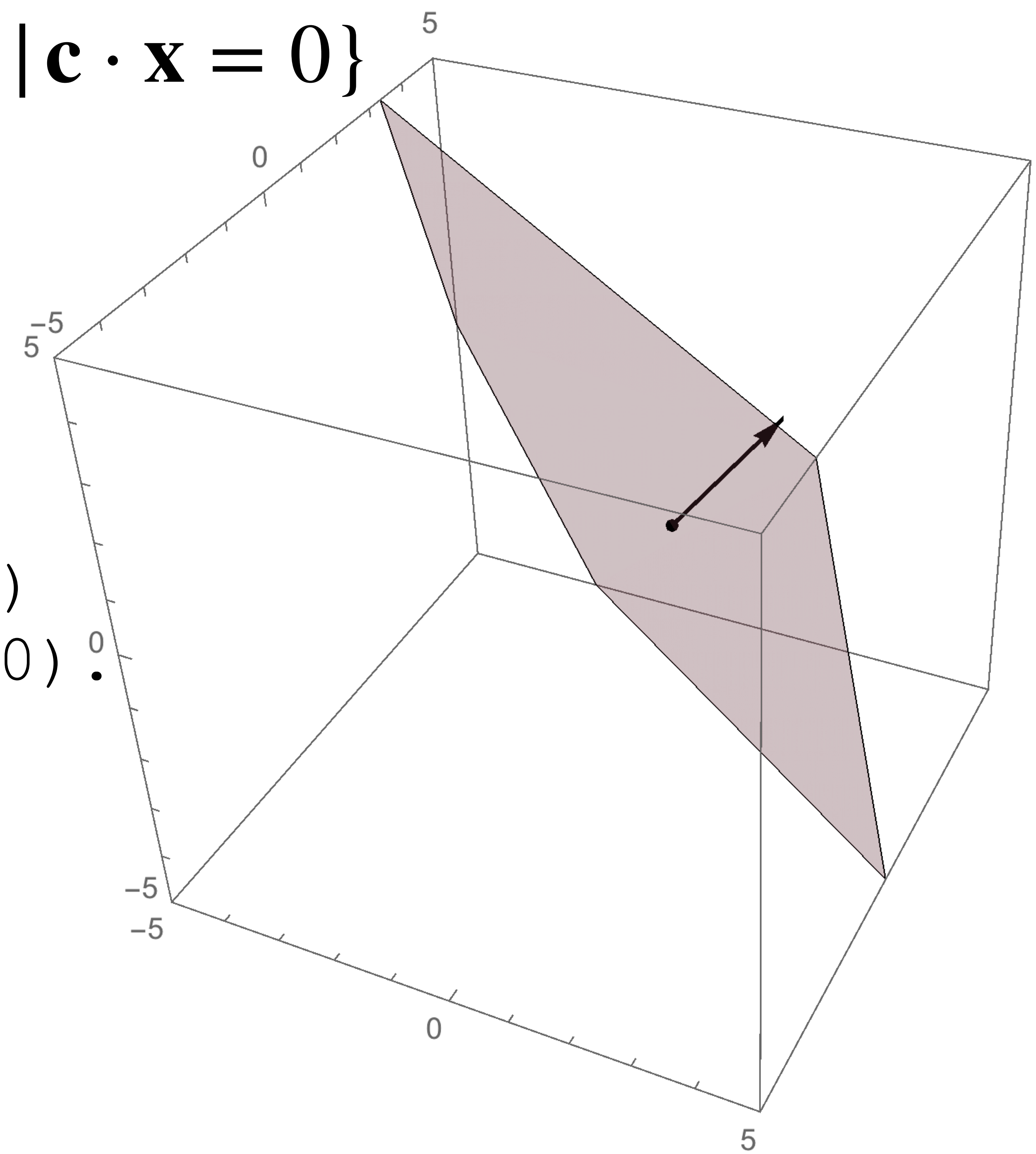


Mathematics of SVM

- Hyper-plane through the origin is an $n-1$ -dimensional subspace of $\mathbb{R}^n$
    - $H = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{c} \cdot \mathbf{x} = 0\}$
  - $\mathbf{c}$ is a "normal" vector, usually of length 1
  - Dot is the dot product
  - Can also use the matrix product
    - $H = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{c}^t \cdot \mathbf{x}\}$

Mathematics

$$H = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{c} \cdot \mathbf{x} = 0\}$$



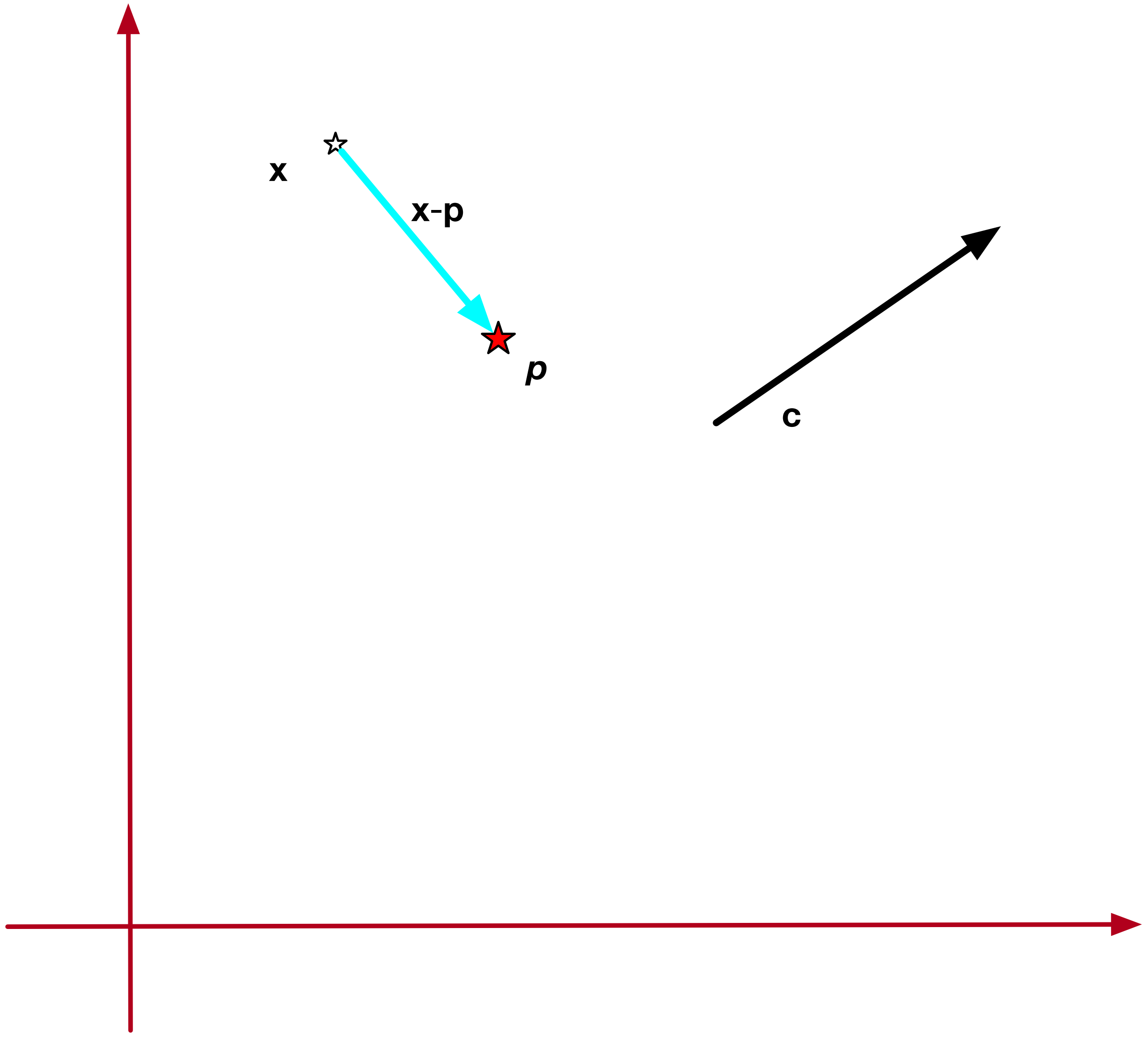
Hyperplane defined
by normal vector (2,3,1)
through the point (1,1,0).

Mathematics

- Task: Find $H_{\mathbf{c},\mathbf{p}}$
  - Orthogonal to vector $\mathbf{c}$
  - Passing through a point $\mathbf{p}$

Mathematics

- $\mathbf{x} \in H_{\mathbf{c},\mathbf{p}}$
  - if and only if $(\mathbf{x} - \mathbf{p})$ is orthogonal to $\mathbf{c}$
  - $(\mathbf{x} - \mathbf{p}) \cdot \mathbf{c} = 0$



Mathematics

- Hyperplane given by
  - $$H_{\mathbf{c},\mathbf{p}} = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{x} \cdot \mathbf{c} - \mathbf{c} \cdot \mathbf{p} = 0\}$$
  - $$= \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{x} \cdot \mathbf{c} + b = 0\}$$
- Defined by a linear functional $\lambda : \mathbb{R}^n \longrightarrow \mathbb{R}$
  - $H_{\mathbf{c},\mathbf{p}} = \{(\mathbf{x} \in \mathbb{R}^n \,|\, \lambda(\mathbf{x}) = b\}$

Mathematics

- Hyperplane separates space into two halves:
  - $H^{-}_{\mathbf{c},\mathbf{p}} = \{(\mathbf{x} \in \mathbb{R}^n \,|\, \lambda(\mathbf{x}) + b < 0\}$
  - $H^{+}_{\mathbf{c},\mathbf{p}} = \{(\mathbf{x} \in \mathbb{R}^n \,|\, \lambda(\mathbf{x}) + b > 0\}$

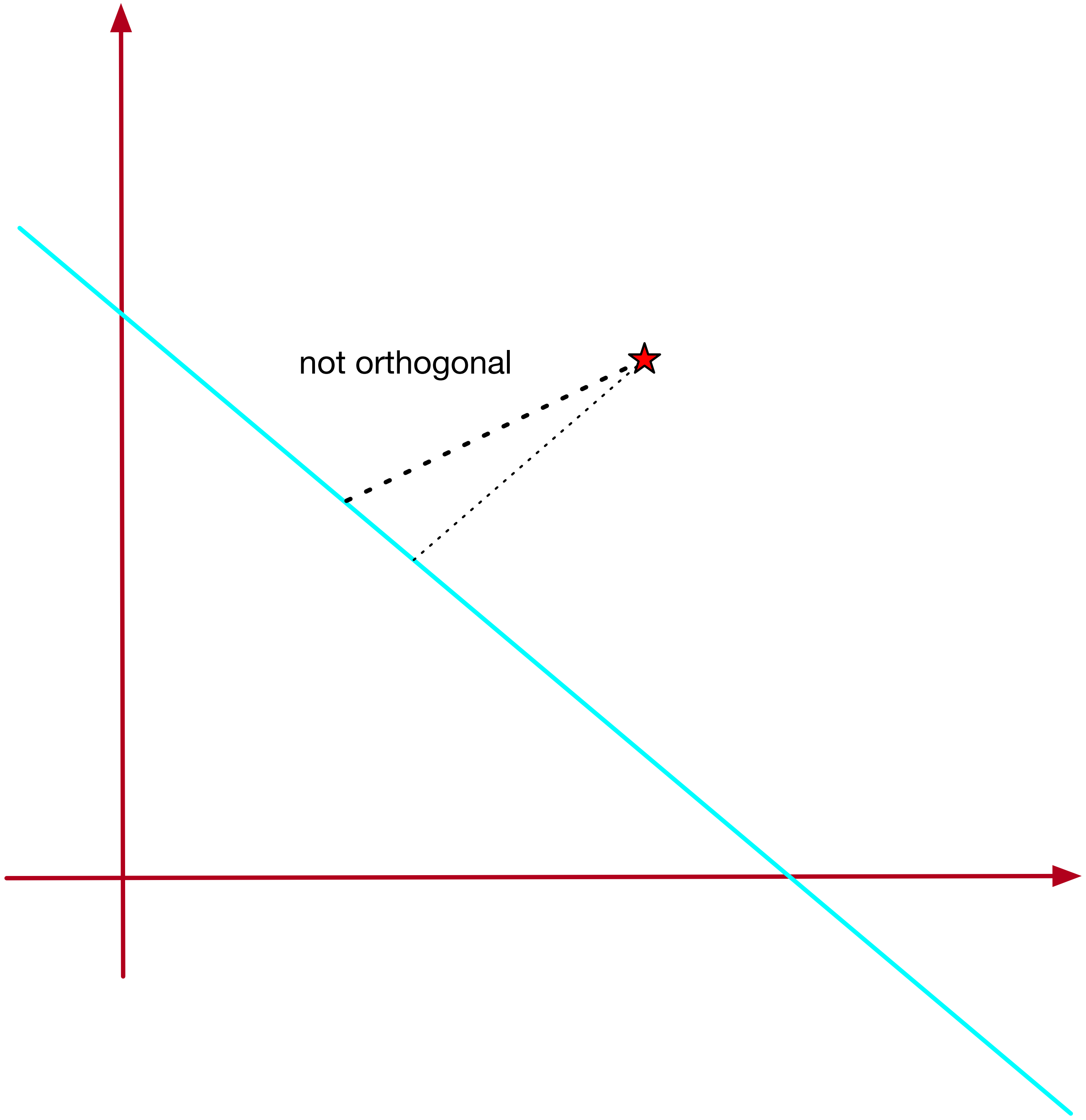
Mathematics

- Learning task:
  - Find a hyperplane such that
    - All feature vectors of one category are in $H^{-}_{\mathbf{c},\mathbf{p}}$
    - All feature vectors of the other category are in $H^{+}_{\mathbf{c},\mathbf{p}}$
  - The best hyperplane
    - (and most likely to achieve good results)
  - Maximizes distance of feature vectors from the plane

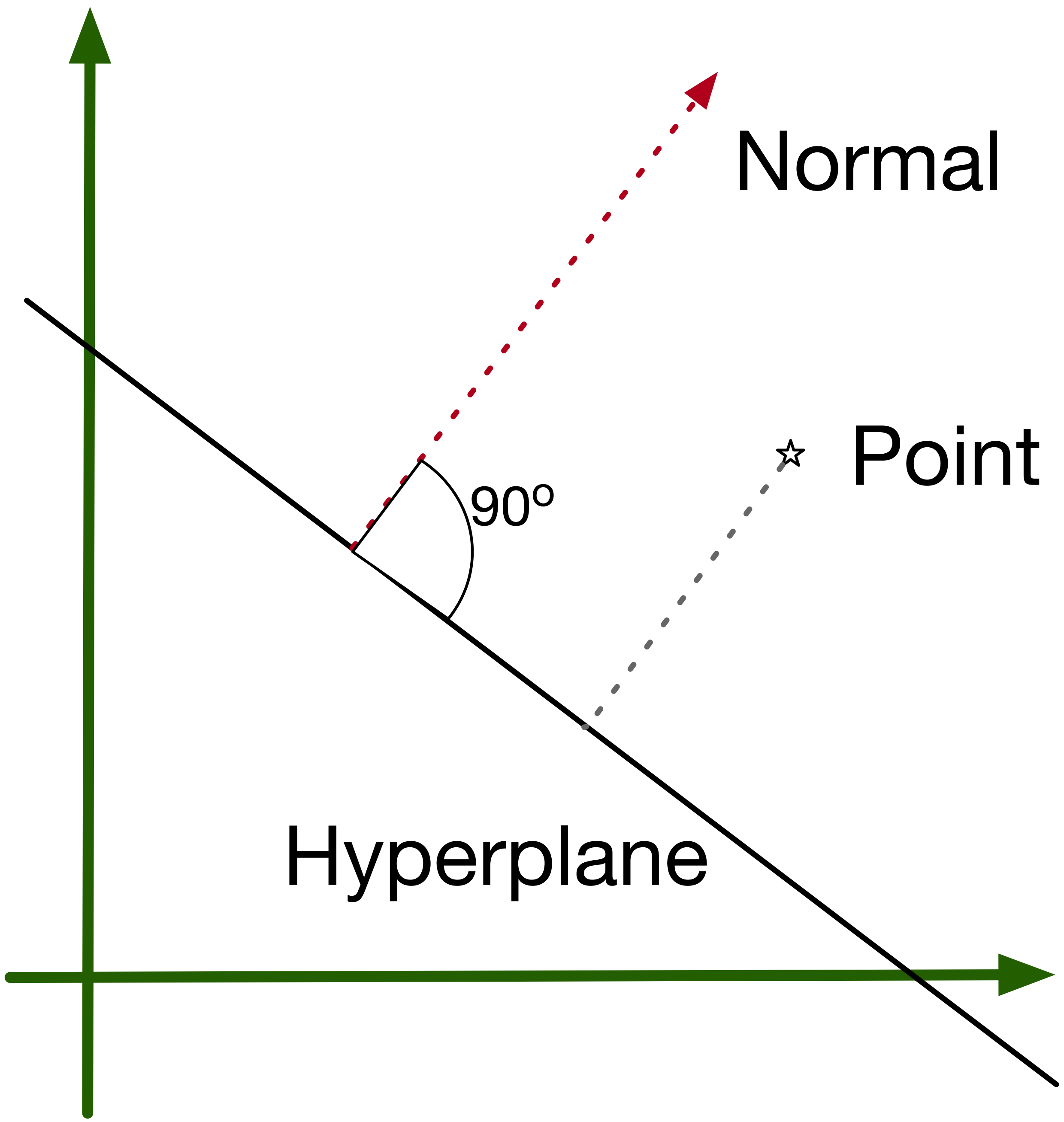
Mathematics

- How do we determine the distance of a point from the hyper-plane?
    - Distance is length of a line between point and closest point on the hyperplane
    - This line needs to be orthogonal to the hyperplane
        - (Otherwise can find something closer

Mathematics



Mathematics



Distance of a point from a hyperplane is the length of a normal of the hyperplane through the point.

Mathematics

- Let $\mathbf{x}$ be a point on the hyperplane
- Hyperplane defined by $H_{\mathbf{w},b} = \{(\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{w} \cdot \mathbf{x} + b = 0\}$
- Write
  - $\mathbf{x} - \mathbf{p} = \alpha \mathbf{w} + \beta \mathbf{v}$, with $\mathbf{w} \cdot \mathbf{v} = 0$
- Multiply with $\mathbf{w}$
  - $\mathbf{w} \cdot (\mathbf{x} - \mathbf{p}) = \alpha \mathbf{w} \cdot \mathbf{w} + \beta \mathbf{w} \cdot \mathbf{v} = \alpha$ since $\mathbf{w} \cdot \mathbf{w} = |\mathbf{w}|^2 = 1$

Mathematics

- Therefore
  - Projection of $\mathbf{x} - \mathbf{p}$ on normal $\mathbf{w}$ is $\mathbf{w} \cdot (\mathbf{x} - \mathbf{p})$.
- This is the distance between $\mathbf{p}$ and the hyperplane:

$$\text{dist}(\mathbf{p}, H_{\mathbf{w},b)} = |\,\alpha\,| = |\,\mathbf{w} \cdot (\mathbf{x} - \mathbf{p})\,|$$

$$= |\,\mathbf{w} \cdot \mathbf{x} - \mathbf{w} \cdot \mathbf{p}\,| = |\,-b - \mathbf{w} \cdot \mathbf{p}\,| = |\,b + \mathbf{w} \cdot \mathbf{p}\,|$$

Mathematics

- Summary:
  - Want all the feature vectors of first category in
    - $H^{-}_{\mathbf{w},b} = \{(\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{w} \cdot \mathbf{x} + b < 0\}$
  - Want all the feature vectors of the second category in
    - $H^{+}_{\mathbf{w},b} = \{(\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{w} \cdot \mathbf{x} + b > 0\}$

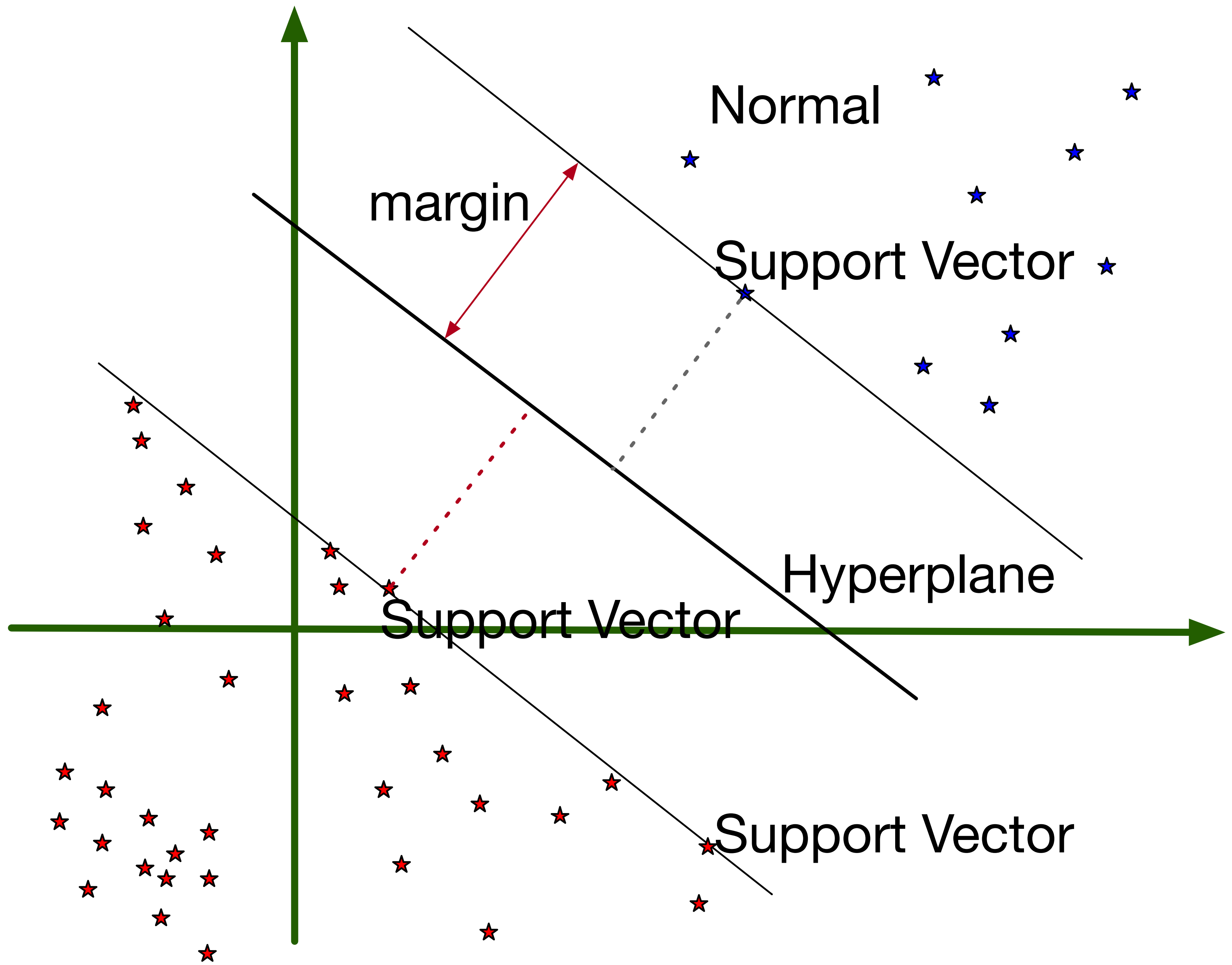
Mathematics

- First category gets label -1 , second category gets label +1
- Condition becomes:
  - label$(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) > 0$
- In addition:
  - $\min\limits_{i \in I}\{$label$(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b)\} \rightarrow$ max
    - where we maximize over all normals $\mathbf{w}$ of length 1 and all scalars b
- All data points $\{\mathbf{x}_i \,|\, i \in I\}$ contribute to the optimization

Mathematics

- Allow normals to have length other than 1
- Replace b with $|\mathbf{w}|b$
- Optimization becomes
  - $$\min_{i \in I}\{\text{label}(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b)/|\mathbf{w}|\} \to \text{max}$$
- The points $\mathbf{x}_i$ where the minima are attained are called the support vectors.

Mathematics



Mathematics

- This even works if the two categories are **not** linearly separable

Mathematics

- Problem: $\min_{i \in I}\{\frac{\text{label}(\mathbf{x}_i)(\mathbf{w}\cdot\mathbf{x}_i + b)}{|\mathbf{w}|}\} \to$ max
- Support vectors: $\text{label}(\mathbf{x})(\mathbf{w}\cdot\mathbf{x} + b)/|\mathbf{w}|$ is minimum
- Can multiply $\mathbf{w}$ and b with a scalar
    - Set scalar to $s = (\text{label}(\mathbf{x})(\mathbf{w}\cdot\mathbf{x} + b))^{-1}$
- Then:
    - $(\text{label}(\mathbf{x})(\mathbf{w}\cdot\mathbf{x} + b)) = 1$
    - Distance of $\mathbf{x}$ to hyperplane is $1/|\mathbf{w}|$
- All other feature vectors: $\text{label}(\mathbf{x})\big(\mathbf{w}\cdot\mathbf{x} + b\big) \geq 1$

Mathematics

- Can now reformulate optimization problem

$$\min_{\mathbf{w},b}\{\frac{|\mathbf{w}|^2}{2}\} \to \min$$

subject to

$$\forall i \in I : \text{label}(\mathbf{x}_i)(\mathbf{w}\mathbf{x}_i + b) \geq 1$$

Mathematics

- Solve with Lagrange multiplier, traditionally called α
  - Solve subject to constraints
    - $\forall i \in I : \alpha_i(\text{label}(\mathbf{x}_i)(\mathbf{w}\mathbf{x}_i + b) - 1) = 0$, $\alpha_i \geq 0$.
- $L = \dfrac{|\mathbf{w}|^2}{2} - \sum_{i \in I} \alpha_i(\text{label}(\mathbf{x}_i)(\mathbf{w}\mathbf{x}_i + b) - 1) \to \min$

Mathematics

- Take partial derivatives with respect to $\mathbf{w}$ and b
- Since $\dfrac{\delta}{\delta w_i}\left(\dfrac{1}{2}\sum_{i=1}^{n} w_i^2\right) = w_i$
- We obtain
- $$\frac{\delta}{\delta \mathbf{w}} L = \mathbf{w} - \sum_{i=1}^{n} \alpha_i \text{label}(\mathbf{x}_i)\mathbf{x}_i$$
- $$\frac{\delta}{\delta b} L = \sum_{i=1}^{n} \alpha_i \text{label}(\mathbf{x}_i)$$

Mathematics

- Setting them to zero for the minimum, we get
- $$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \text{label}(\mathbf{x}_i)\mathbf{x}_i$$
- and therefore
- $$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \text{label}(\mathbf{x}_i)\mathbf{x}_i$$

Mathematics

- $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \text{label}(\mathbf{x}_i)\mathbf{x}_i$ implies
  - $\mathbf{w}$ is a linear combination of features

Mathematics

- Use $\frac{|\mathbf{w}|^2}{2} = \frac{\mathbf{w}\cdot\mathbf{w}}{2}$ in our optimization problem
  - $L = \frac{|\mathbf{w}|^2}{2} - \sum_{i\in I} \alpha_i(\text{label}(\mathbf{x}_i)(\mathbf{w}\mathbf{x}_i + b) - 1) \to \min$
- Use

$$-\sum_{i\in I} \alpha_i(\text{label}(\mathbf{x}_i)(\mathbf{w}\mathbf{x}_i + b) - 1)$$

$$= -\mathbf{w}\cdot(\sum_{i=1}^{n} \alpha_i \text{label}(\mathbf{x}_i)) - \sum_{i=1}^{n} \alpha_i \text{label}(\mathbf{x}_i) b + \sum_{i=1}^{n} \alpha_i$$

$$= -\mathbf{w}\cdot\mathbf{w} + \sum_{i=1}^{n} \alpha_i$$

Mathematics

- Function L is now simplified:
- $L = -\frac{1}{2}\mathbf{w} \cdot \mathbf{w} + \sum_{i=1}^{n} \alpha_i$

Mathematics

- Plugging in again
- $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \text{label}(\mathbf{x}_i)\mathbf{x}_i$
- we obtain
- $L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \text{label}(\mathbf{x}_i)\text{label}(\mathbf{x}_j)\mathbf{x}_i \cdot \mathbf{x}_j$
- which we want to maximize subject to constraints
- $\forall i \in I : \alpha_i \geq 0$ and $\sum_{i=1}^{n} \alpha_i \text{label}(\mathbf{x}_i) = 0$

Mathematics

- This is the "dual" optimization problem, but it is ***quadratic*** in the alphas.
    - This means that it can be solved using Kuhn Tucker

Mathematics: Soft Margin SVM

- The preceding works if the data set is linearly separable
  - If not, we introduce slack variables
    - label$(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$
  - They measure the violation of the separation condition
    - If zero: separation condition is fulfilled and the point lies $\geq \dfrac{1}{|\mathbf{w}|}$ away from the hyperplane
    - If $0 < \xi_i < 1$: point lies inside the margin, but point is classified
    - If $\xi_i \geq 1$, point is mis-classified

Mathematics

- Choosing an optimization function is no longer straight-forward
    - Do we want a hyperplane with a few violations or do we want to minimize the total amount of violations

Mathematics

- One possibility:
  - $\min_{\mathbf{w},b,\xi_i} \left(\frac{|\mathbf{w}|^2}{2} + C \sum_{i=1}^{n} \xi_i^k \right)$
  - subject to
  - $\forall i \in I :$ label$(\mathbf{x}_i)\big(\mathrm{w} \cdot \mathbf{x}_i + b\big) \geq 1 - \xi_i$
  - $\forall i \in I : \; \xi_i \geq 0$

Mathematics

- The power *k* in
  - $$\min_{\mathbf{w},b,\xi_i}\left(\frac{|\mathbf{w}|^2}{2}+C\sum_{i=1}^{n}\xi_i^k\right)$$
- describes our policy:
  - *k*=1: Hinge loss
  - *k*=2: Quadratic losss
- *C* describes the trade-off between large margins and loss minimization

Mathematics

- Much research has been spent on optimizing for each case
- Luckily, we do not have to use them

SVM with Scikit-Learn

- Has a whole module svm
- `from sklearn import svm`

SVM with Scikit-Learn

- Generate data

```
d_a  =  np.random.multivariate_normal(
          mean = [1,3],
          cov = [ [.1,.02], [.02,.1]],
          size=100)
d_b  =  np.random.multivariate_normal(
          mean = [3,4],
          cov = [ [.2,.005], [.005,.2]],
          size=100)
```

SVM with Scikit-Learn

- Data are the feature, target are the labels

```
data = np.concatenate((d_a, d_b), axis=0)
target = np.concatenate((np.zeros(100), np.ones(100)))
```

- Now fit the whole data set

```
clf = svm.SVC(kernel='linear', C=3)
clf.fit(data, target)
```

SVM with Scikit-Learn

- Now print stuff:
  - To see the coefficient, we needed to use the linear kernel

```
print(clf.support_vectors_)
w = clf.coef_[0]
a = -w[0] / w[1]
xx = np.linspace(1.5, 2.25)
yy = a * xx - (clf.intercept_[0]) / w[1]
print('w',w)
print(a)
```

SVM with Scikit-Learn

- Support vectors

```
[[1.83311697 3.13576805]
 [1.49922547 3.41011599]
 [1.80136907 3.32299734]
 [2.10066171 3.82740692]
 [2.16113553 3.35389843]
 [2.48658474 2.79743539]]
```

- Normal and intercept:

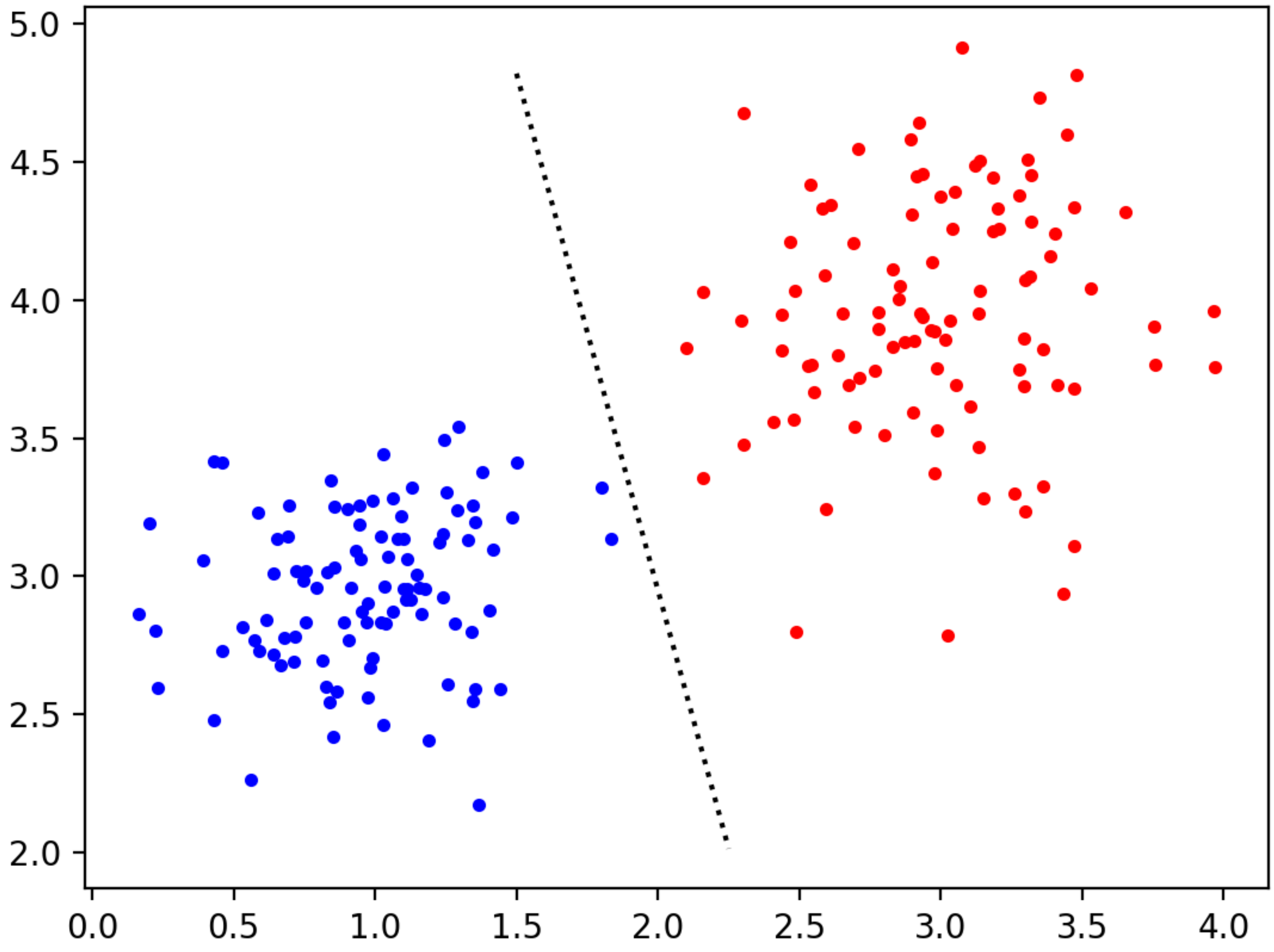
```
w [2.63976203 0.98909858]
-2.6688563564106373
```

SVM with Scikit-Learn

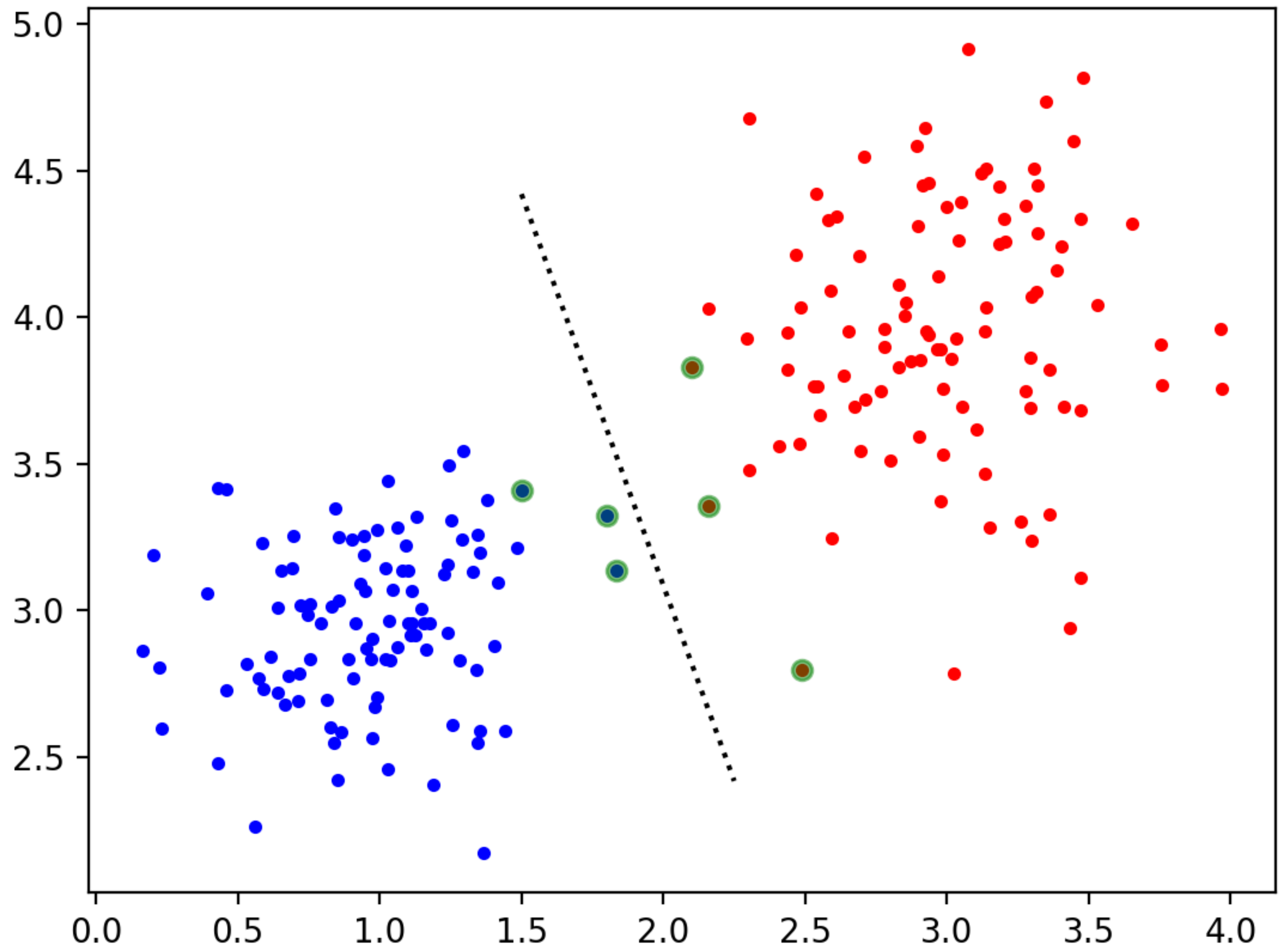
- Draw the data and the hyperplane

```
plt.figure(1)
plt.plot(d_a[:,0], d_a[:,1], 'b.')
plt.plot(d_b[:,0], d_b[:,1], 'r.')
plt.plot(xx, yy, 'k:')
plt.show()
```

SVM with Scikit-Learn



SVM with Scikit-Learn

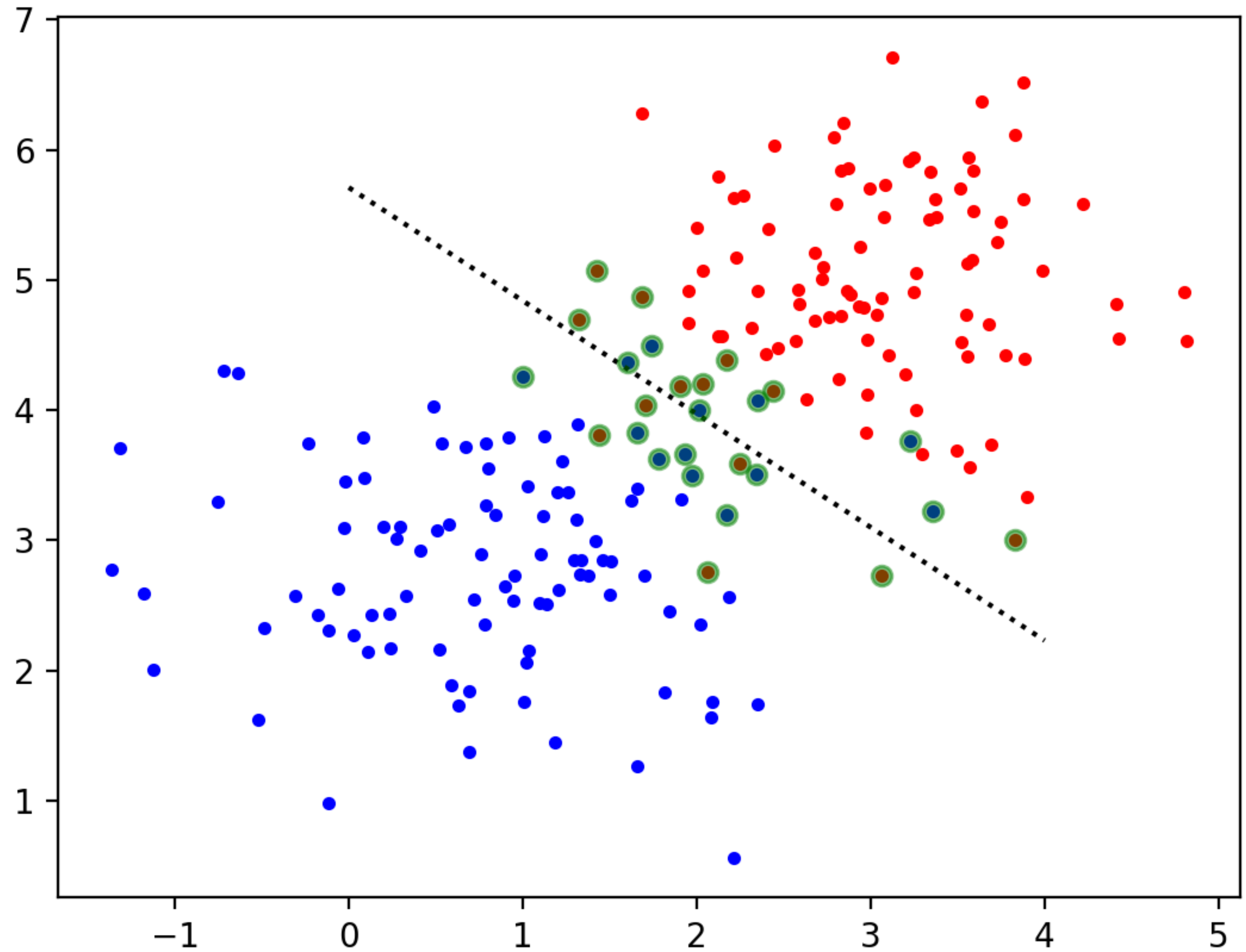


SVM with Scikit-Learn

- New points are evaluated only using the support vectors
  - This makes SVM more efficient

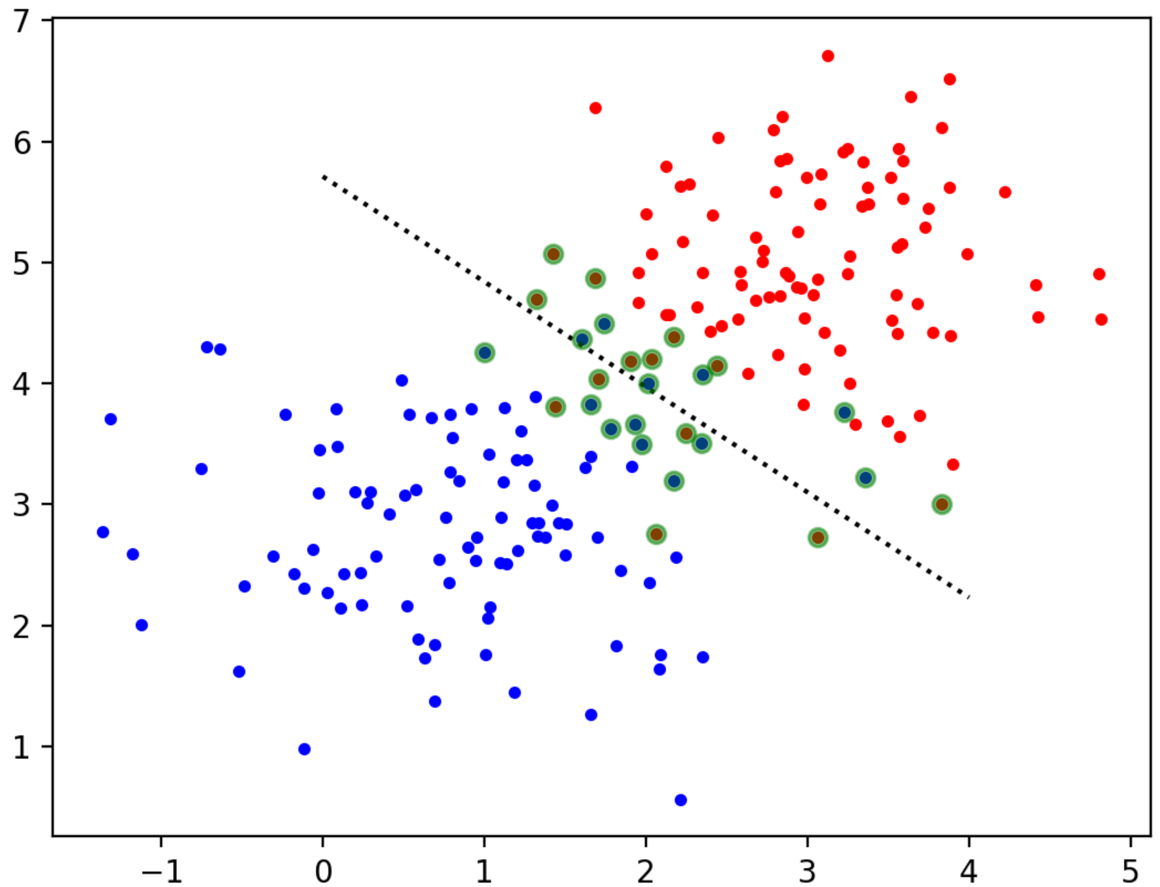
SVM with Scikit-Learn

- Example with non-separable data sets

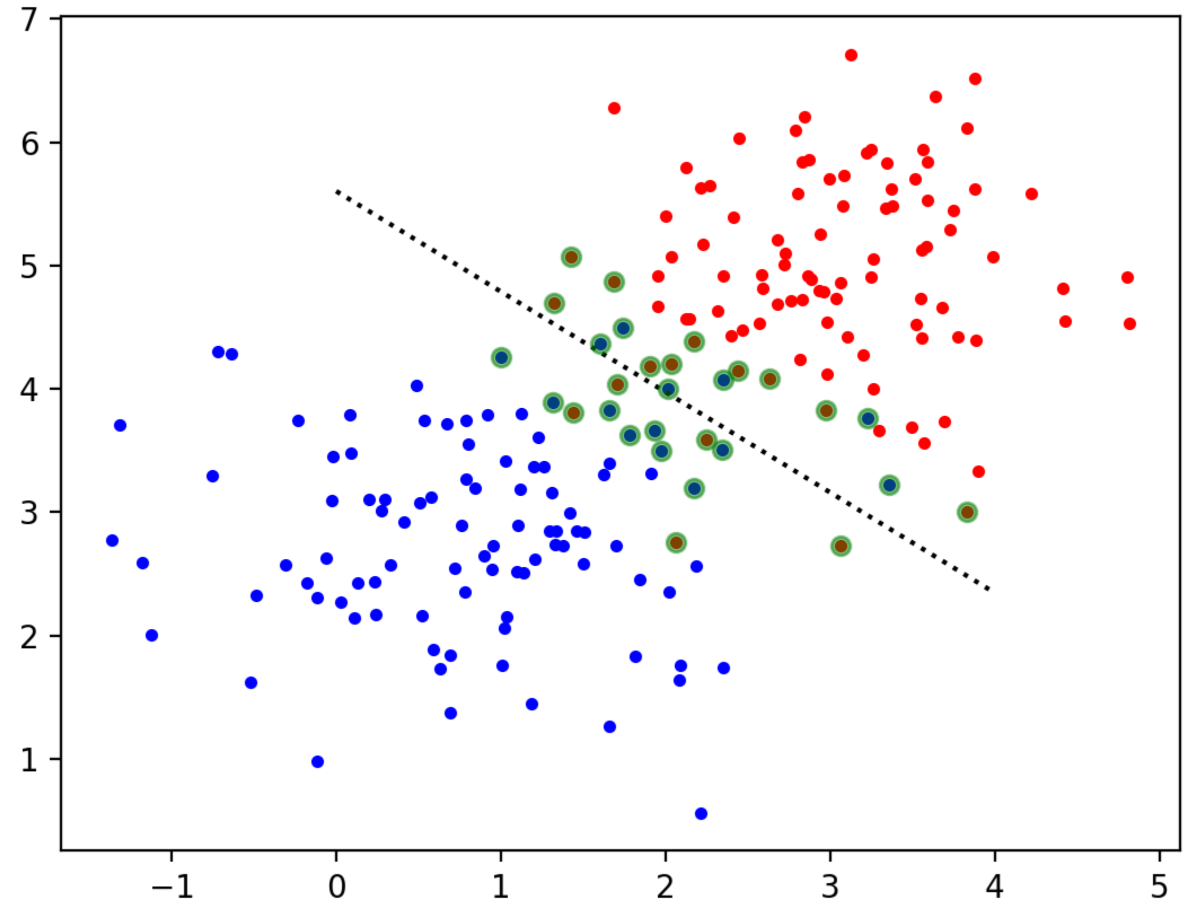


SVM with Scikit-Learn

- In this case, the C-value does not change much



C=3

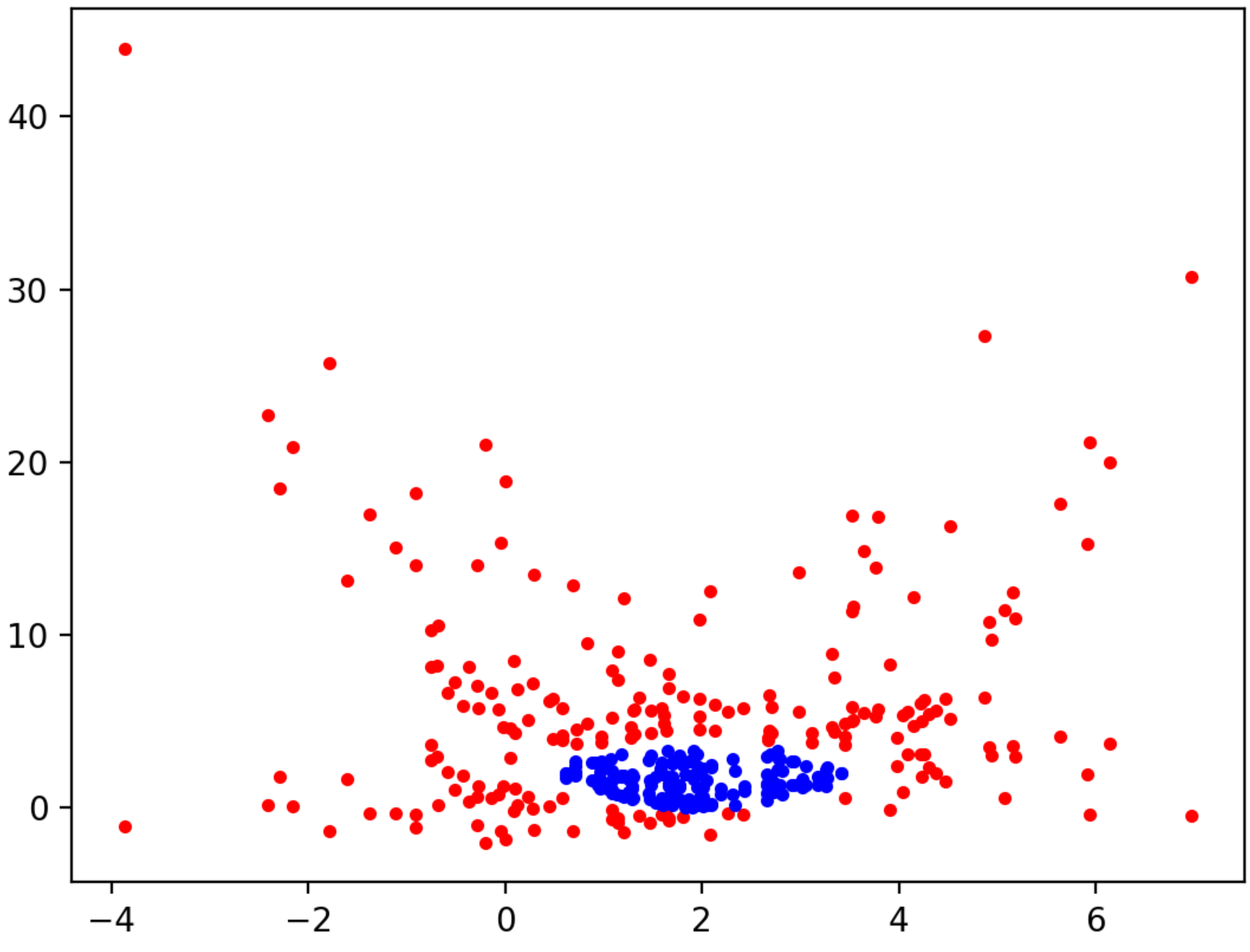


C=0.5

SVM with Scikit-Learn

- This is a very difficult set to classify without transformation

SVM with Scikit-Learn



SVM with Scikit-Learn

- If we try with several kernels, results are not so good
  - Import some stuff

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn import metrics
from sklearn.model_selection import train_test_split
np.random.seed(12)
```

SVM with Scikit-Learn

- Generate data set

```
d_a  =  np.random.multivariate_normal(
     mean = [2,2],
     cov = [ [.2,.15], [.15,.2]],
     size=150)
d_b  =  np.random.multivariate_normal(
     mean = [2,2],
     cov = [ [3,1], [1,3]],
     size=220)

one = np.array( [x for x in d_b if
      np.linalg.norm(x-[2,2])<1.5])
two = np.array( [x for x in d_b if
      np.linalg.norm(x-[2,2])>2])
```

SVM with Scikit-Learn

- Split dataset into training and test set (70% training)

```
X_train, X_test, y_train, y_test =
    train_test_split(
            data,
            target,
            test_size=0.3)
```

SVM with Scikit-Learn

- Train

```
clf = svm.SVC(kernel = 'sigmoid')
clf.fit(X_train, y_train)
```

SVM with Scikit-Learn

- Predict and get accuracy:

```
y_pred = clf.predict(X_test)

print("Accuracy:",
       metrics.accuracy_score(y_test, y_pred))
```

SVM with Scikit-Learn

```
Accuracy: linear 0.62
Accuracy: poly 0.7666666666666667
Accuracy: rbf 1.0
Accuracy: sigmoid 0.34
```

SVM with Scikit-Learn

- Could use a custom kernel

```
def my_kernel(X, Y):
    return np.dot((X-[2,2])**2,((Y-[2,2])**2).T)
```

```
clf = svm.SVC(kernel = my_kernel)
clf.fit(X_train, y_train)
```

SVM with Scikit-Learn

- Not surprisingly, this works completely

```
Accuracy: 1.0
```

SVM with Scikit-Learn

- Radial basis function kernel
- $K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{||\mathbf{x} - \mathbf{y}||^2}{2\sigma^2}\right)$