Support Vector Machines

Linear classifier

This time will write b explicitly.

$$h(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x} + b)$$

Which outputs +1 or –1.

Say:

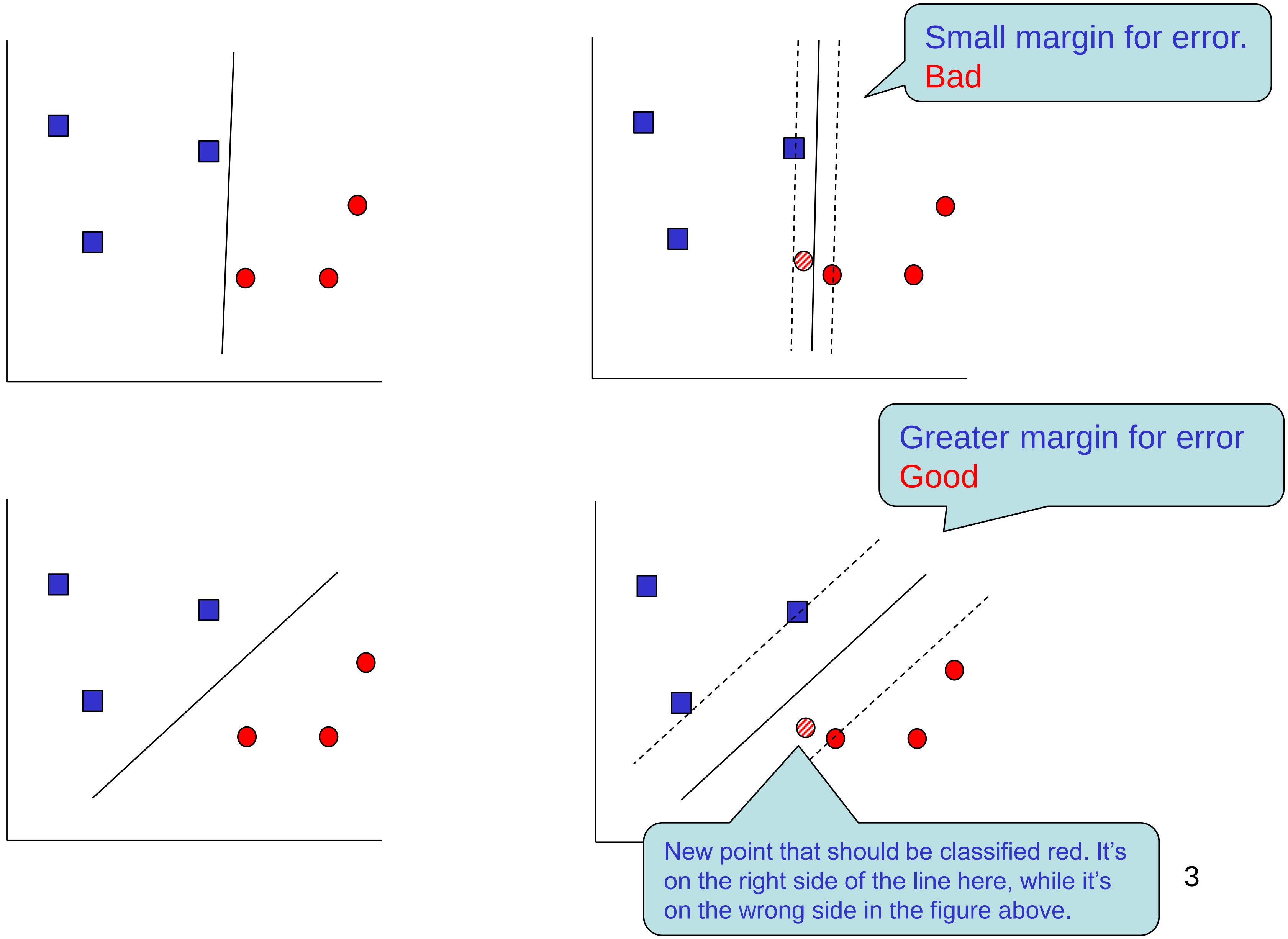
+1 corresponds to blue, and

-1 to red, or vice versa.

There are many lines however that do the job.

Which one to choose?

Margin for different separators



Small margin for error.
Bad

Greater margin for error
Good

New point that should be classified red. It’s on the right side of the line here, while it’s on the wrong side in the figure above.

Scale Invariance

- We rescale **w** and *b* (without changing the line) such that:

$$\mathbf{w} \cdot \mathbf{x}^{k_1} + b = 1$$

for the closest point(s) to the line on the +1 side, and

$$\mathbf{w} \cdot \mathbf{x}^{k_2} + b = -1$$

for the closest point(s) to the line on the -1 side.

Closest points are called "support vectors".

Margin

- **Digression.** The distance of a point to a line: $\mathbf{w}\cdot\mathbf{x}+b=0$

$$\frac{|\mathbf{w}\cdot\mathbf{x}^k+b|}{\|\mathbf{w}\|}=\frac{|\mathbf{w}\cdot\mathbf{x}^k+b|}{\sqrt{\mathbf{w}\cdot\mathbf{w}}}$$

Consider the **closest** points to the line we are building **on each side** respectively. Their distances to the line are:

$$\frac{|\mathbf{w}\cdot\mathbf{x}^{k_1}+b|}{\|\mathbf{w}\|}=\frac{1}{\|\mathbf{w}\|} \qquad \frac{|\mathbf{w}\cdot\mathbf{x}^{k_2}+b|}{\|\mathbf{w}\|}=\frac{1}{\|\mathbf{w}\|}$$

$\frac{1}{\|\mathbf{w}\|}$ is called *margin*.

We want a line that maximizes margin.

Optimization problem

- Maximizing margin is equivalent to minimizing:

$$\frac{1}{2}\mathbf{w}\cdot\mathbf{w}$$

However, without any constraints, the above becomes min for **w**=**0**, which is not useful.

Optimization problem

- Maximizing margin is equivalent to minimizing:

$$\frac{1}{2}\mathbf{w}\cdot\mathbf{w}$$

and this is subject to constraints:

$$y^k\left(\mathbf{w}\cdot\mathbf{x}^k+b\right)\geq 1 \qquad k\in[1,n]$$

y^k is the class, +1 or -1 of training point $\mathbf{x}^k$.

$\mathbf{x}^k$ is a training point. We have *n* of them. They are **constant** vectors, not variable.

The constraints are satisfied iff the line properly separates the training points. We assume for now that the training points are linearly separable.

Example

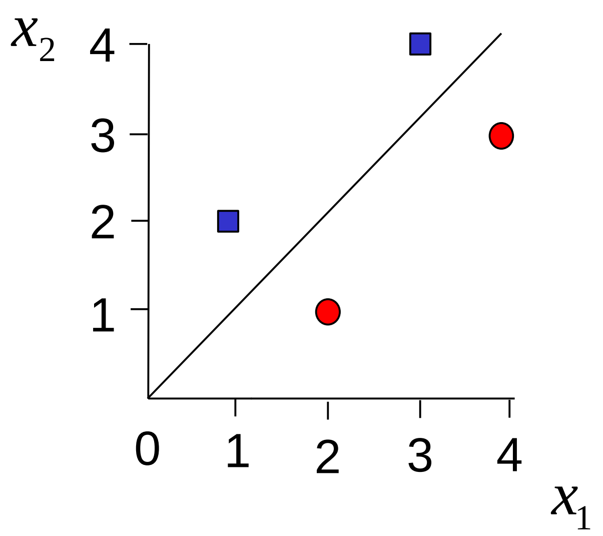
Training tuples:

([1, 2], +1)

([2, 1], −1)

([3, 4], +1)

([4, 3], −1)



$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}\cdot\mathbf{w}$$

subject to

$$y^k\left(\mathbf{w}\cdot\mathbf{x}^k + b\right) \geq 1, \quad \forall k \in [1, n]$$

For this example, solution is easy to see:

b = 0 and **w** = [−1, +1]

i.e. the line is: $-x_1+x_2=0$

All conditions are satisfied (see on the right).

$$\min_{w_1, w_2} \frac{1}{2}\left(w_1^2 + w_2^2\right)$$

subject to

$$(+1)(w_1 + 2w_2) \geq 1$$

$$(-1)(2w_1 + w_2) \geq 1$$

$$(+1)(3w_1 + 4w_2) \geq 1$$

$$(-1)(4w_1 + 3w_2) \geq 1$$

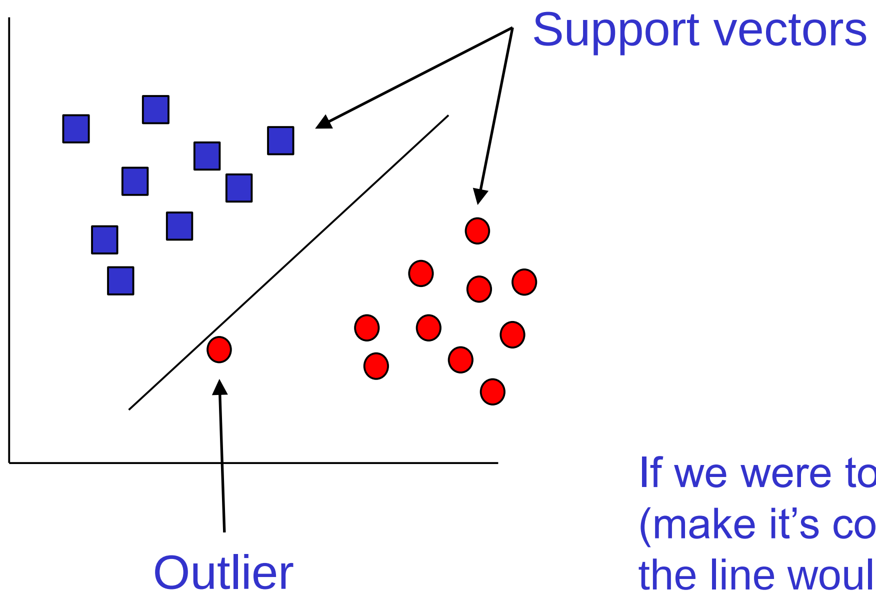
$$\text{margin} = \frac{1}{\|\mathbf{w}\|} = \frac{1}{\sqrt{1+1}} = \frac{1}{\sqrt{2}}$$

w* and *b**

- We denote by **w*** and *b** the solutions to the constrained optimization problem.

SOFT MARGIN

When there are a few outlier points



If we were to “fully” consider the outlier (make it’s constraint inequality true), the line would have to be more on the left, with much less margin.

To address this, we will cut some slack to constraint inequalities, so that they don’t push the line to much.

Modified Optimization – Soft Margin

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{k=1}^{n}\xi_k$$

Subject to

$$y^k\left(\mathbf{w}\cdot\mathbf{x}^k + b\right) \geq 1 - \xi_k \qquad k \in [1,n]$$

$$\xi_k \geq 0$$

If we don't add this term, ξ_k's can take any big value, making the right-hand side of the inequalities arbitrary small, and such, rendering the constraints useless because we again can produce **w=0** as a solution.

C is a hyper parameter that controls how much "slack" we want to give the inequalities to be true.

WHERE COULD THE TRAINING POINTS BE?

1st Case – Well Classified

$$y^k\left(\mathbf{w}^* \cdot \mathbf{x}^k + b^*\right) > 1$$

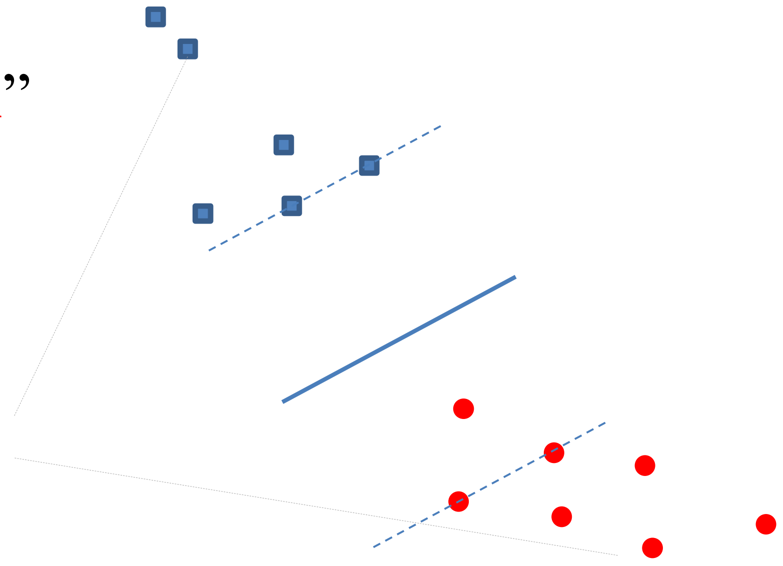
$$y^k(\mathbf{w}^* \cdot \mathbf{x}^k + b^*) > 1 - \xi_k^*$$

$$\xi_k^* = 0$$

No need for slack.

Optimization will set the slack to 0 for these points.

These points are called "Well Classified"

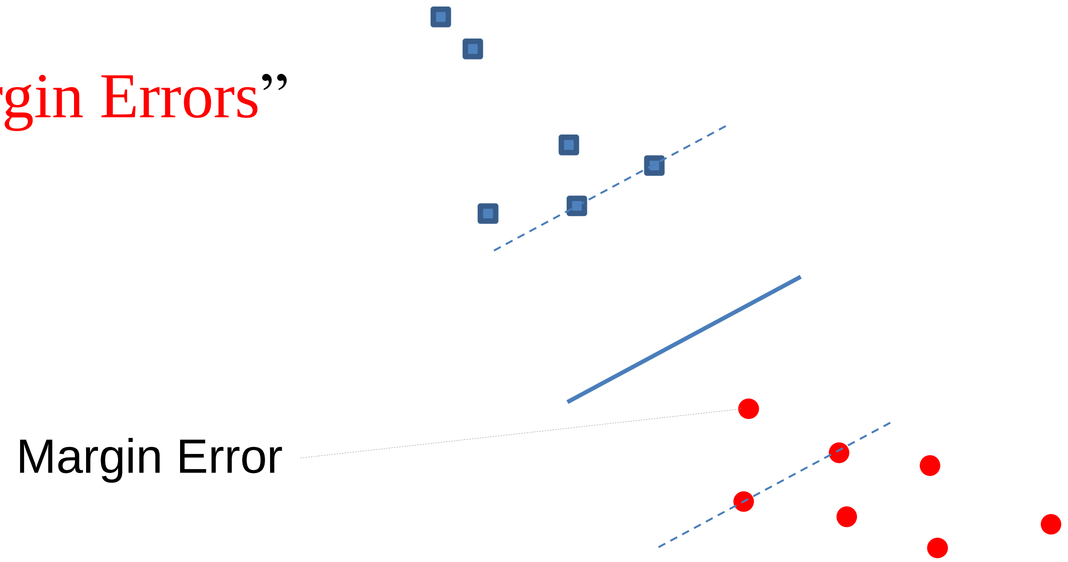


2nd Case – Margin Errors

$$y^k\left(\mathbf{w}^* \cdot \mathbf{x}^k + b^*\right) < 1 \qquad y^k\left(\mathbf{w}^* \cdot \mathbf{x}^k + b^*\right) = 1 - \xi_k^*$$

$$\xi_k^* > 0$$

These points are called “Margin Errors”

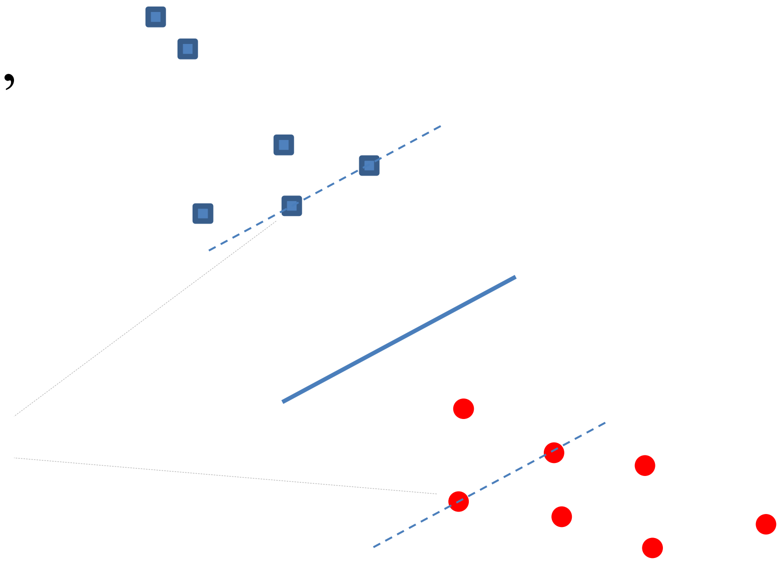


3rd Case – On the Margin

$$y^k\left(\mathbf{w}^* \cdot \mathbf{x}^k + b^*\right) = 1$$

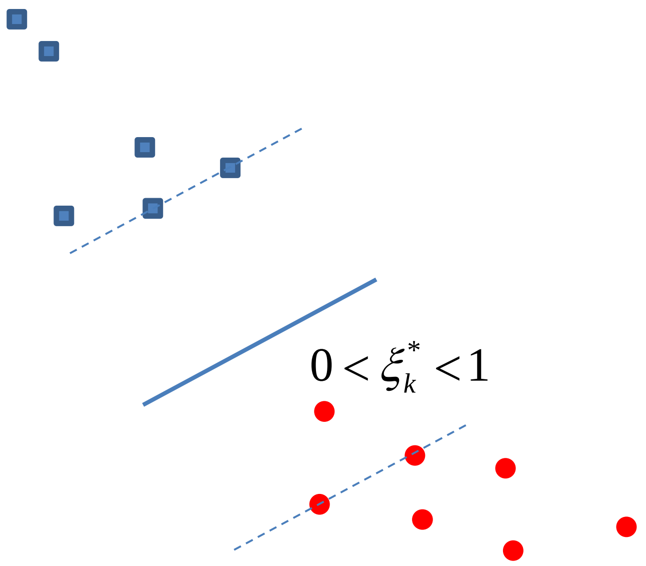
i.e. $\xi_k^* = 0$

These points are called "On the Margin" or "Support Vectors".



More on the 2nd Case – Margin Errors

$\xi_k^* > 0$



Because $\mathbf{x}^k$ still on the right side of the separator line, i.e.

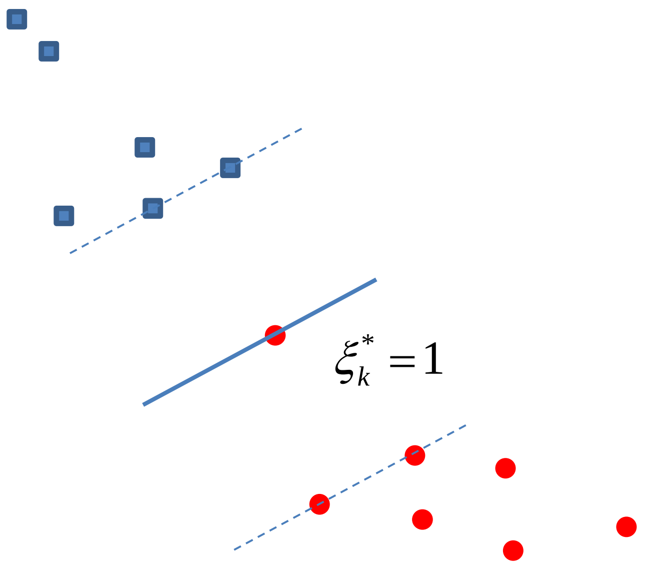
$$0 < y^k\left(\mathbf{w}^* \cdot \mathbf{x}^k + b^*\right) < 1$$

Recall

$$y^k\left(\mathbf{w}^* \cdot \mathbf{x}^k + b^*\right) = 1 - \xi_k^*$$

More on the 2nd Case – Margin Errors

$\xi_k^* > 0$



Because $\mathbf{x}^k$ is on the separator line, i.e.

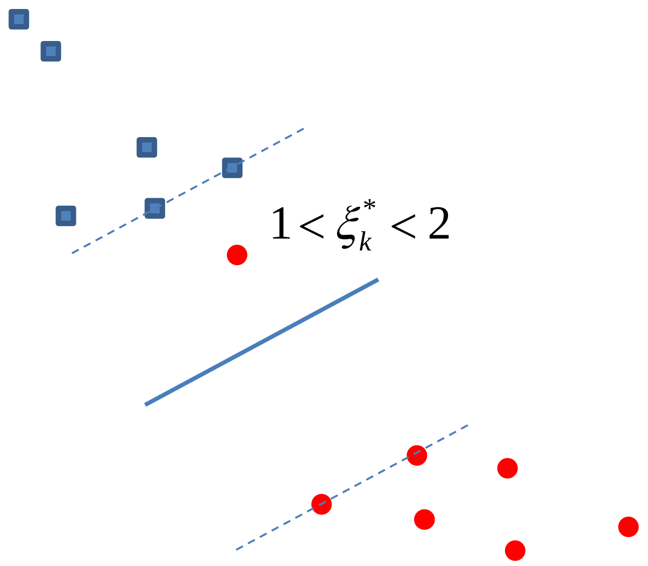
$$0 = y^k\left(\mathbf{w}^* \cdot \mathbf{x}^k + b^*\right) < 1$$

Recall

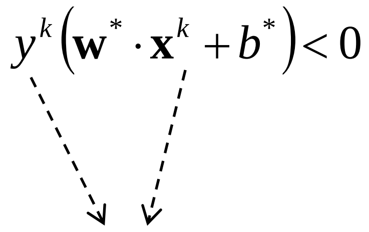
$$y^k\left(\mathbf{w}^* \cdot \mathbf{x}^k + b^*\right) = 1 - \xi_k^*$$

More on the 2nd Case – Margin Errors

$\xi_k^* > 0$



Because $\mathbf{x}^k$ is on the wrong side of the separator line, i.e.



They disagree in sign

Recall

$$y^k\left(\mathbf{w}^* \cdot \mathbf{x}^k + b^*\right) = 1 - \xi_k^*$$

KERNEL TRICK

What if data is like this:

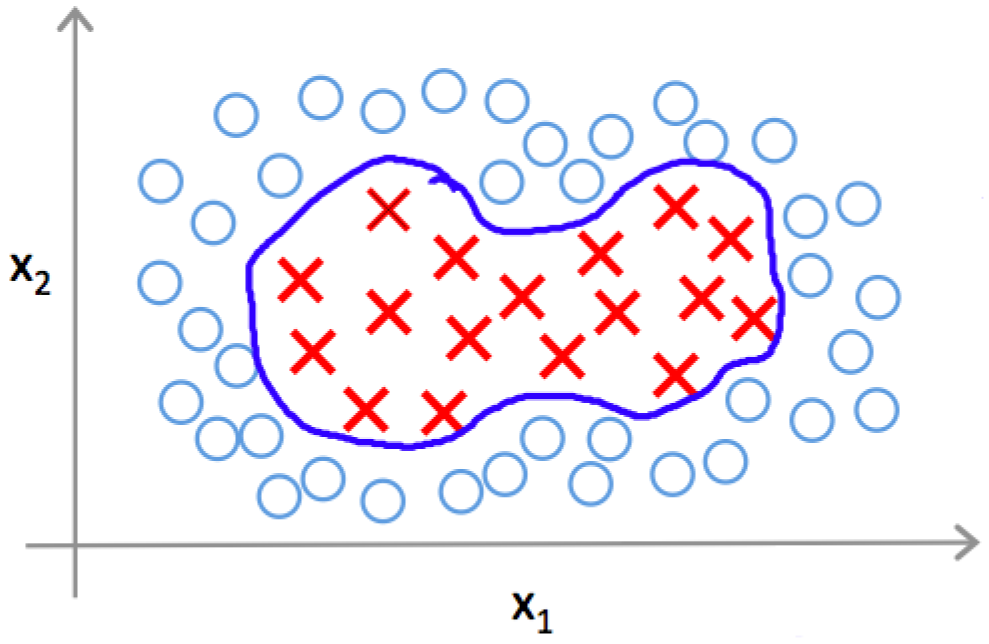
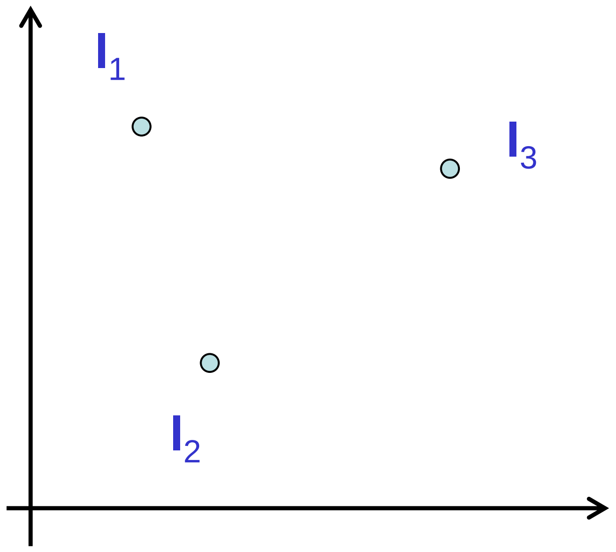


Fig. from: Andrew Ng, Stanford University

Idea - Kernel

- Create new attributes (features) from the existing ones.
  - One way is to add polynomials based on the existing features
  - Another (preferred) way is to add as new features the **closeness** of data points from some **landmark points**.

We will use Gaussian closeness (or similarity)



$$\mathrm{sim}(\mathbf{x}, \mathbf{l}_1) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{l}_1\|^2}{2\sigma^2} \right)$$

$$\mathrm{sim}(\mathbf{x}, \mathbf{l}_1) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{l}_1\|^2}{2\sigma^2} \right)$$

$$\mathrm{sim}(\mathbf{x}, \mathbf{l}_1) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{l}_1\|^2}{2\sigma^2} \right)$$

Kernel

Kernels and similarity

$$f_1 = \mathrm{sim}(\mathbf{x}, \mathbf{l}_1) = \exp\left(-\frac{\|\mathbf{x}\text{-}\,\mathbf{l}_1\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{i=1}^{m}(x_{i1}-l_{i1})^2}{2\sigma^2}\right)$$

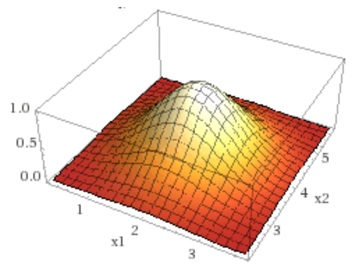
If $\mathbf{x}$ is close to $\mathbf{l}_1$: $f_1 \approx \exp\left(-\frac{0}{2\sigma^2}\right) = 1$

If $\mathbf{x}$ is far from $\mathbf{l}_1$: $f_1 = \exp\left(-\frac{\text{big number}}{2\sigma^2}\right) \approx 0$

Each landmark gives a new feature:

$$\mathbf{l}_1 \rightarrow f_1$$
$$\mathbf{l}_2 \rightarrow f_2$$
$$\mathbf{l}_3 \rightarrow f_3$$

Sigma: The smaller the sigma the narrower the “bump”.



Choosing landmarks

- **All** the **training data** points become **landmarks**.

- Then compute $f_1, \ldots, f_n$ for each training point.

- Finally find the **maximum margin classifier** (hyperplane) using these new features.

Challenge: Find the best C and sigma

- One the best SVM implementations: libsvm
  - Available in scikit learn.

- To find C and sigma perform a grid search
  - i.e. try a range of C values and sigma values, and find the best pair
  - See: “A practical guide to SVM classification” in the above site.

Grid Search for Parameters

```
from sklearn import svm, datasets
from sklearn.model_selection import GridSearchCV

iris = datasets.load_iris()

parameters = {'gamma':[0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30, 100],
'C':[0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30, 100]}

svc = svm.SVC()

clf = GridSearchCV(svc, parameters)

clf.fit(iris.data, iris.target)

# examine the best model
print(clf.best_score_)
print(clf.best_params_)
```

gamma=1/sigma^2