SUPPORT VECTOR MACHINE

Shingchern D. You

References

- Easy to read materials (to form this PPT file)
  - http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf
  - http://www.robots.ox.ac.uk/~az/lectures/ml/lect3.pdf
- Math-oriented material
  - http://cs229.stanford.edu/notes/cs229-notes3.pdf

Binary classifiers

- Binary classification problem: want to classify test samples into two classes
- Suppose we have training dataset

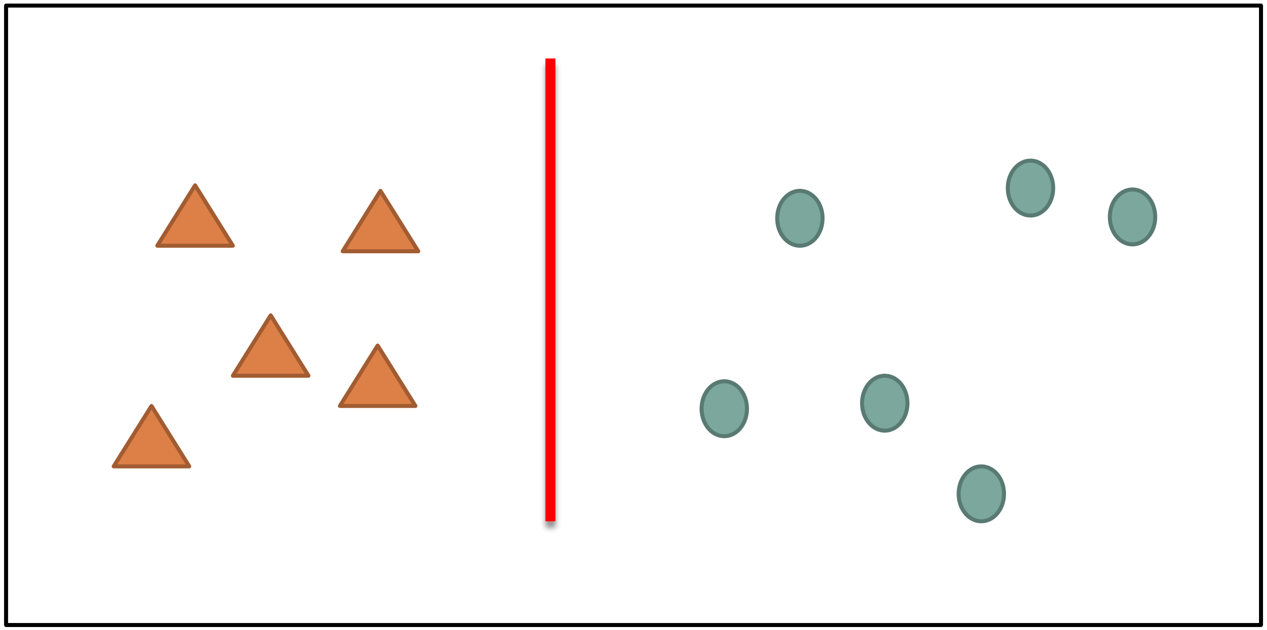
  $$\{(\mathbf{x}_{(k)}, d_{(k)}) | 1 \le k \le M\}, \text{ with } d_{(k)} \in \{1, -1\}$$

- We want to train a classifier $y_{(k)} = f\left(\mathbf{x}_{(k)}\right)$ such that

$$\begin{cases} y_{(k)} > 0 \text{ if } d_{(k)} = +1 \\ y_{(k)} < 0 \text{ if } d_{(k)} = -1 \end{cases}$$

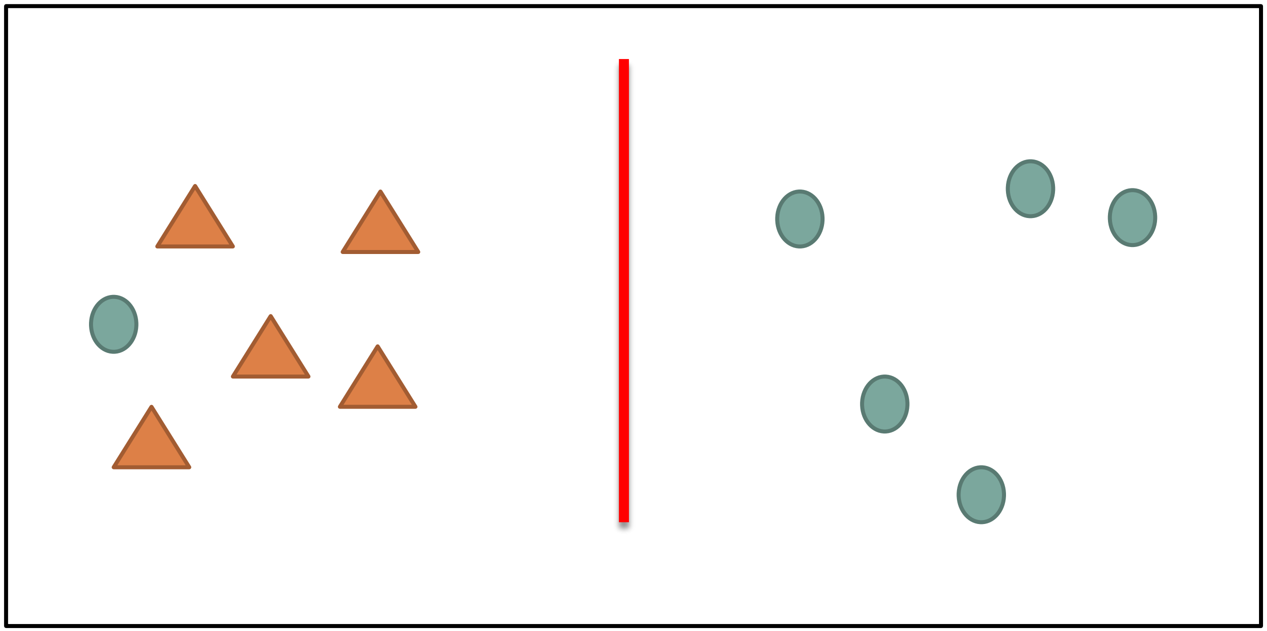
Linear separability

- If we can use a linear function to separate two classes, the data are **linearly separable**



Linear separability

- Not linearly separable case

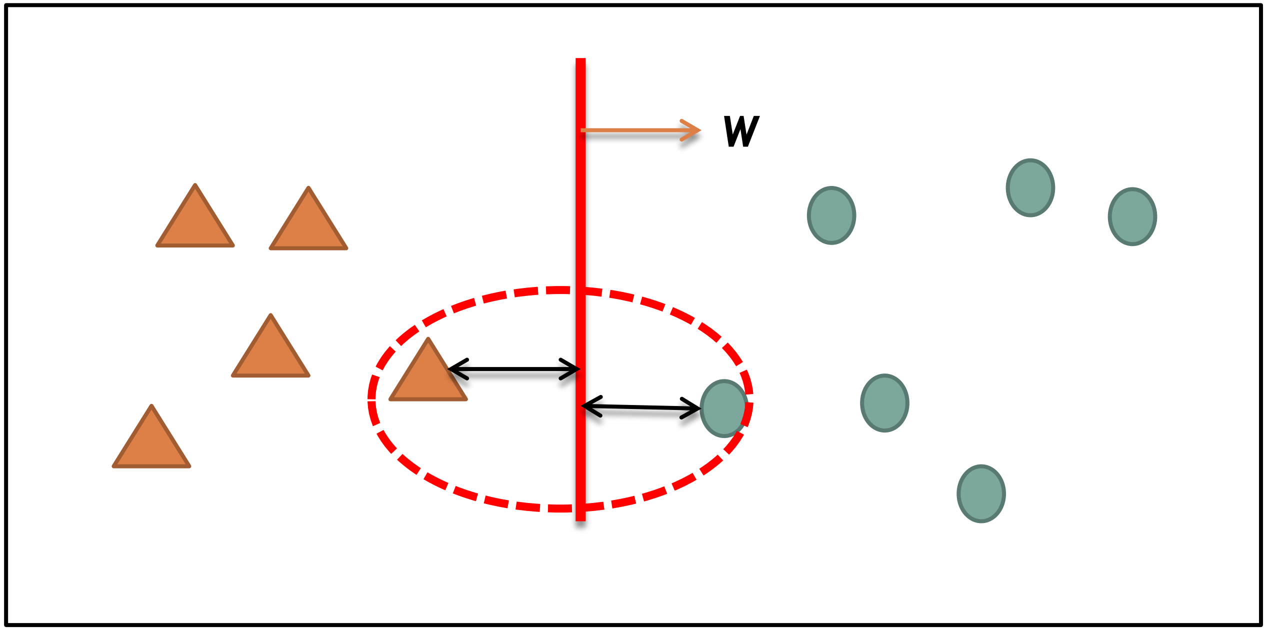


Linear classifier

- A linear classifier has the following math form
  $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ (b is a scalar)
- $\mathbf{w}$ is called weight vector and is **normal** to the separating hyperplane and b is called bias
- If we want, we can also use gradient descent method to find $\mathbf{w}$

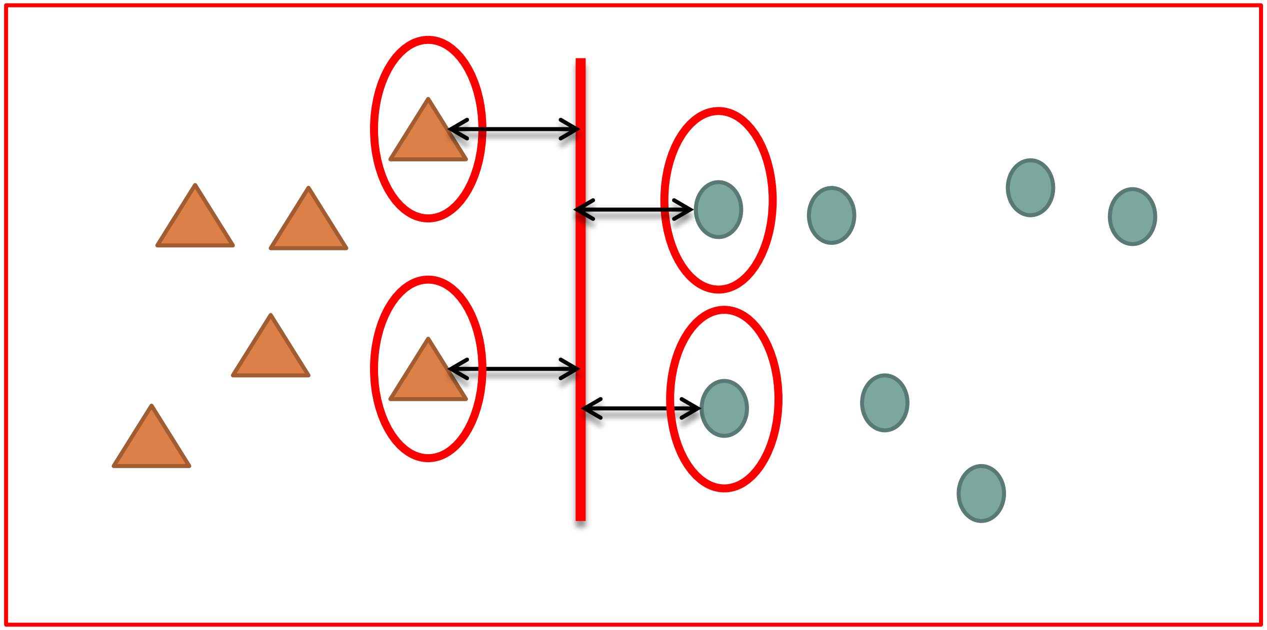
Linear classifier

- Want to find a good **w** for classification
- How to define “good”
- We want maximum margin (shown below)



Support vectors

- Data points closest to the decision boundary are called support vectors (circled below)



Support vectors

- In the above description, we know all support vectors have **equal distance** to the boundary
- We are free to scale $\mathbf{w}$ and b so that the distance is 1
- For supporting vectors $\mathbf{x}_+$ and $\mathbf{x}_-$ (belonging to +1 and -1 classes), we then have

$$\mathbf{w}^T\mathbf{x}_+ + b = 1$$
$$\mathbf{w}^T\mathbf{x}_- + b = -1$$

Support vectors

- Therefore, the margin is $\frac{\mathbf{w}}{||\mathbf{w}||} \cdot (\mathbf{x}_+ - \mathbf{x}_-) = \frac{\mathbf{w}^T\mathbf{x}_+ - \mathbf{w}^T\mathbf{x}_-}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$
- The above is the **inner product** of the unit normal vector and the vector difference

Optimal margin classifier

- With the above reasoning, we have the following optimization problem: **For all k**

$$\max_{\mathbf{w}} \frac{\mathbf{2}}{\|\mathbf{w}\|} \text{ subject to } f\left(\mathbf{x}_{(k)}\right): \begin{cases} \geq 1 \text{ if } d_{(k)} = +1 \\ \leq -1 \text{ if } d_{(k)} = -1 \end{cases}$$

- The above problem is equivalent to

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } d_{(k)}(\mathbf{w}^T \mathbf{x}_{(k)} + b) \geq 1 \quad \text{for all k}$$

- It is a **quadratic programming** problem

Optimal margin classifier

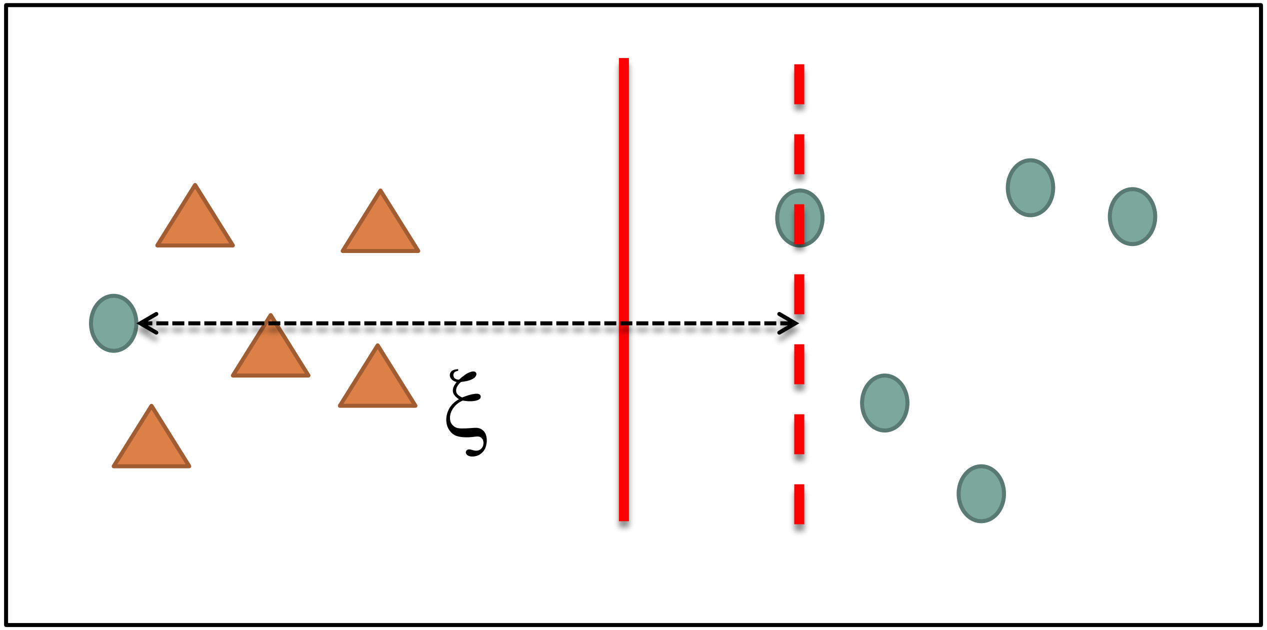
- We can use Lagrange multipliers to solve this problem

$$\mathcal{L}(\mathbf{w}, b, \lambda) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{k=1}^{N} \lambda_{(k)}\left[d_{(k)}\left(\mathbf{w}^T\mathbf{x}_{(k)} + b\right) - 1\right]$$

- Stationary points satisfy the following
  - If $d_{(k)}(\mathbf{w}^T\mathbf{x}_{(k)} + b - 1) = 0$ then $\lambda_{(k)} \geq 0$ (boundary case, opposite gradient direction)
  - If $d_{(k)}(\mathbf{w}^T\mathbf{x}_{(k)} + b - 1) > 0$ then $\lambda_{(k)} = 0$ (constraint inactive)
  - Therefore, $\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}, b, \lambda) = 0$; $\nabla_b\mathcal{L}(\mathbf{w}, b, \lambda) = 0$; $\nabla_\lambda\mathcal{L}(\mathbf{w}, b, \lambda) = 0$ if $\lambda \neq 0$

Soft margin solution

- What if we encounter linearly non-separable dataset



Soft margin solution

- We may allow for some data points not having margin greater than 1
- Define the distance between support vector and the violating data $\mathbf{x}_{(k)}$ is $\xi_{(k)}$
- We know $\xi_{(k)} \geq 0$
- Want to add penalty term in the cost function

Soft margin solution

- Thus, the new optimization problem becomes

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_{(k)}$$

subject to $d_{(k)}(\mathbf{w}^T\mathbf{x}_{(k)} + b) \geq 1 - \xi_{(k)}$ for all k

- In the equation, ***C* is a regularization parameter**
- Large C implies stronger “margin” requirement
- Small C allows soft “margin”

Soft margin solution

- The term $\xi_{(k)}$ appears only if data point $\mathbf{x}_{(k)}$ does not have enough marine
- We may formulate it in another form

$$\xi_{(k)} = \max\left(0,1 - d_{(k)}y_{(k)}\right)$$
$$= \max\left(0, 1 - \ d_{(k)}(\mathbf{w}^T\mathbf{x}_{(k)} + b)\ \right)$$

- This function is called hinge (loss) function

Hinge function

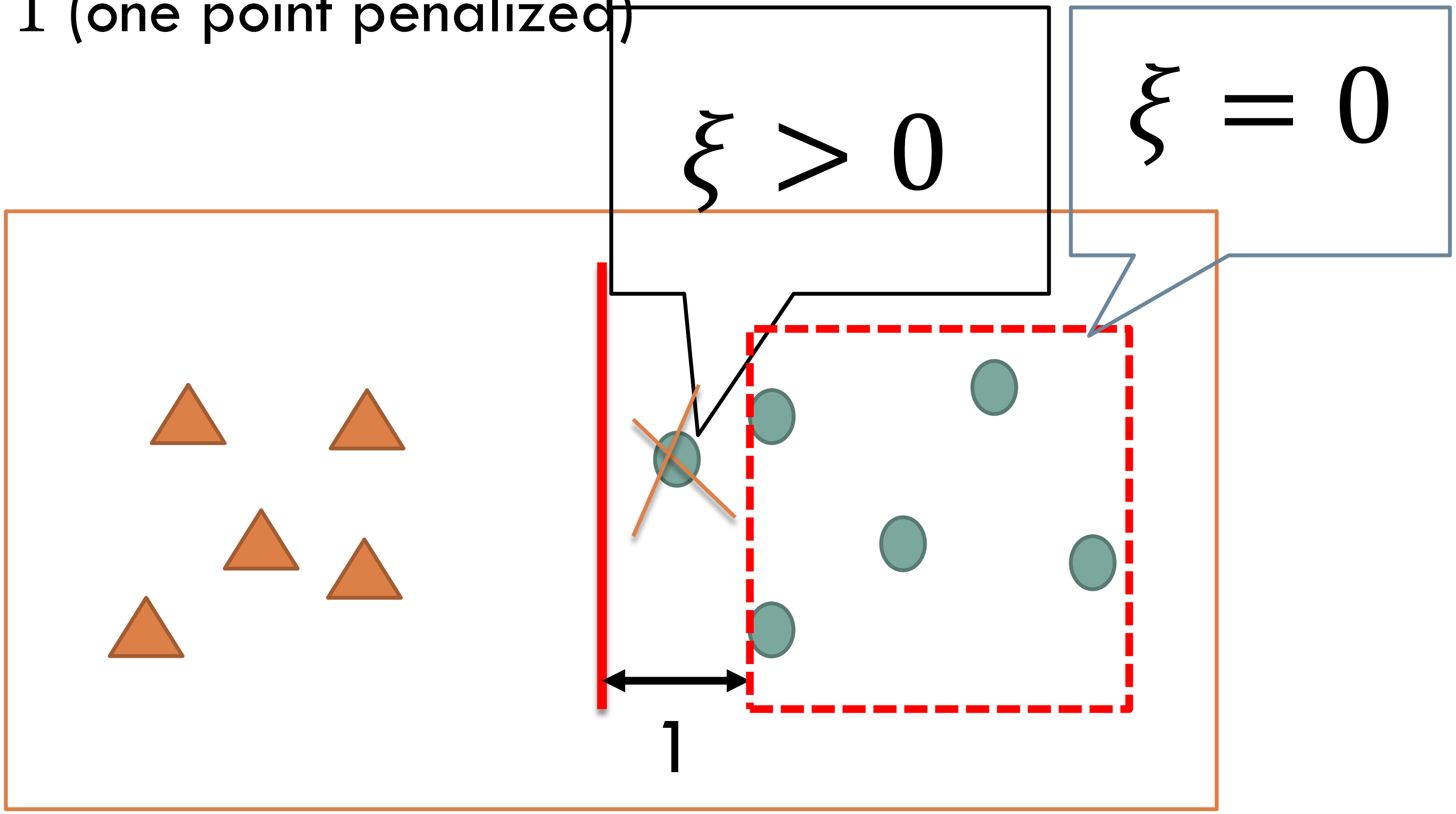
- If we modify the max function as
  $\xi_{(k)} = \max\left(0, \Delta - d_{(k)} y_{(k)}\right)$

we can determine when penalty starts

- We consider some cases
  - $\Delta = 1$
  - $\Delta = 0$
- You can check other cases

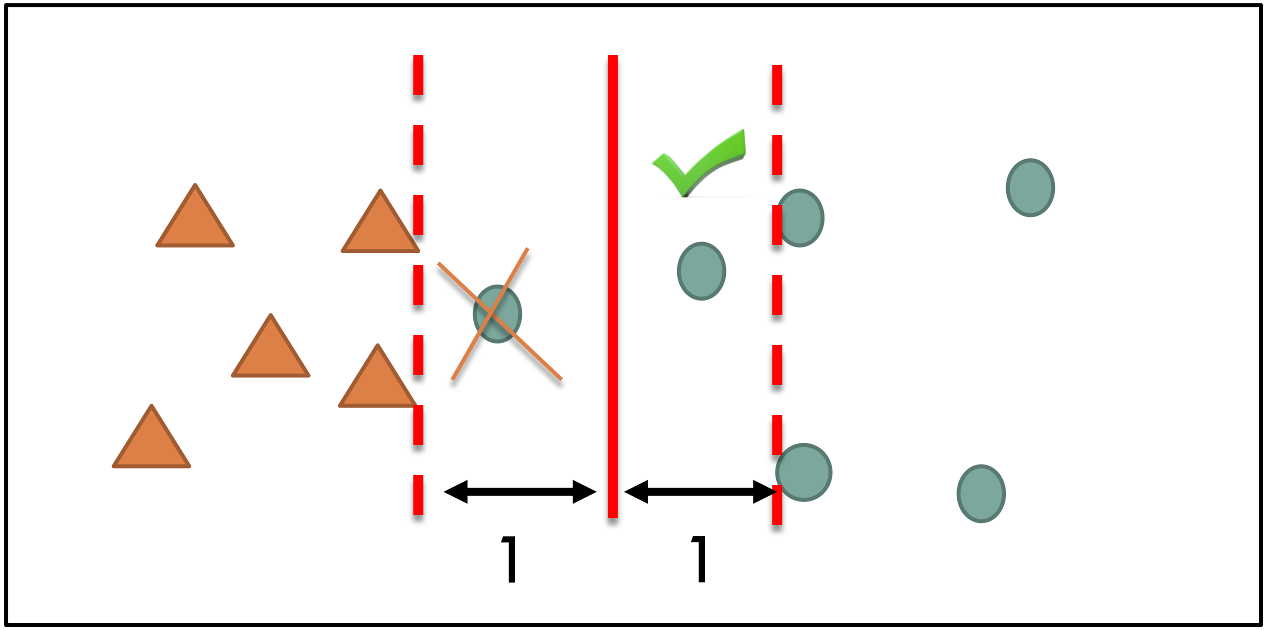
Hinge function

- $\Delta = 1$ (one point penalized)



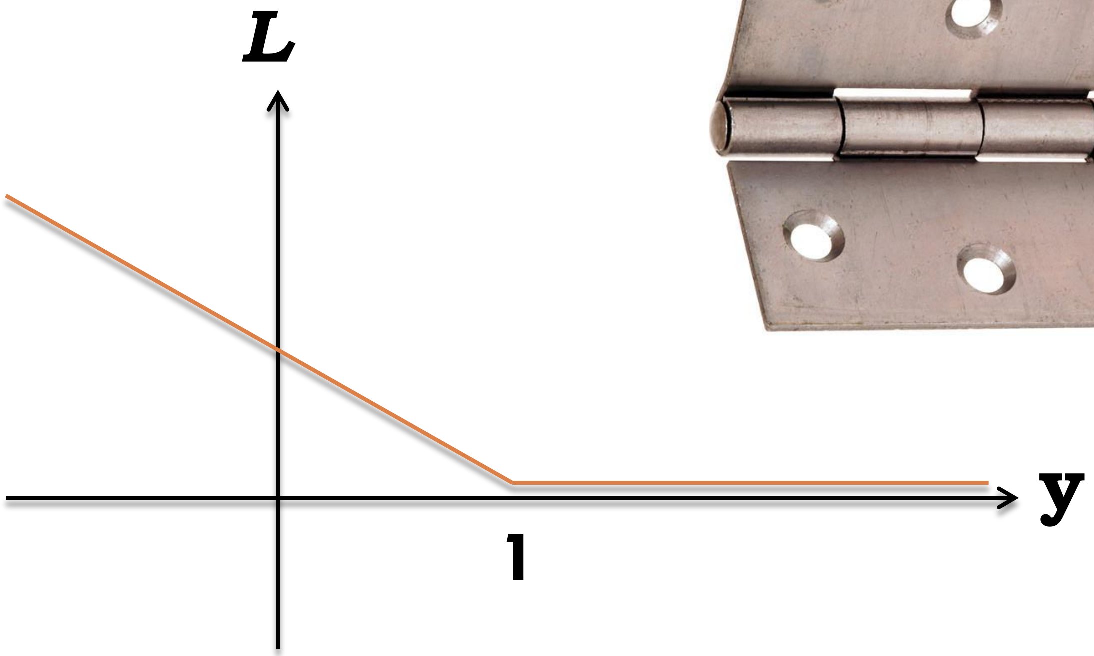
Hinge function

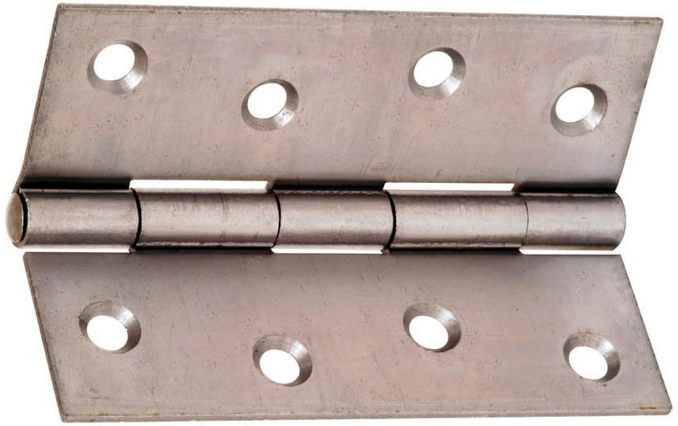
- $\Delta = 0$ (x means ξ>0, V means $\xi = 0$)



Hinge function

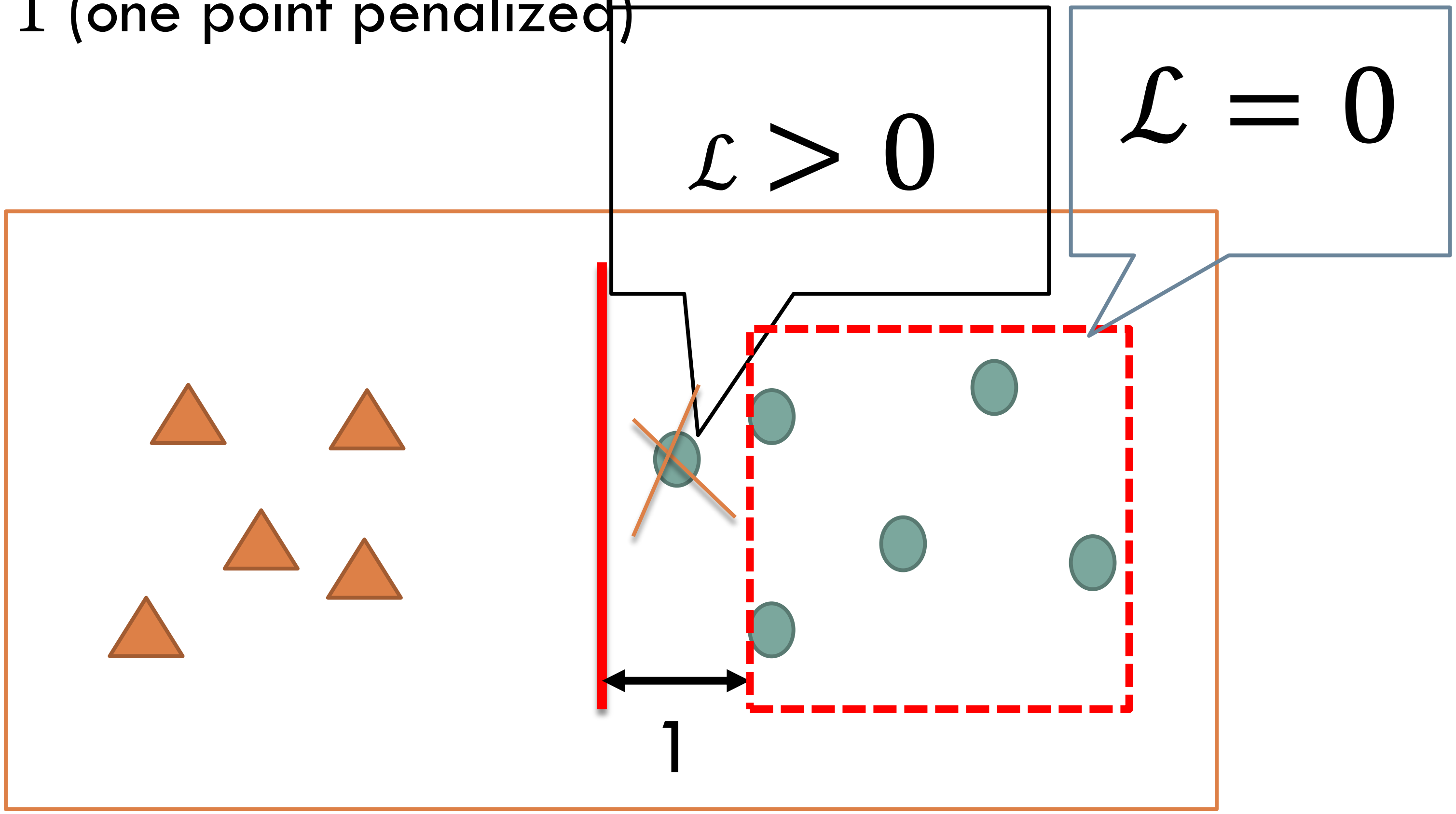
- Why called hinge function
- Let $\mathcal{L}(y) = \max(0, 1 - y)$





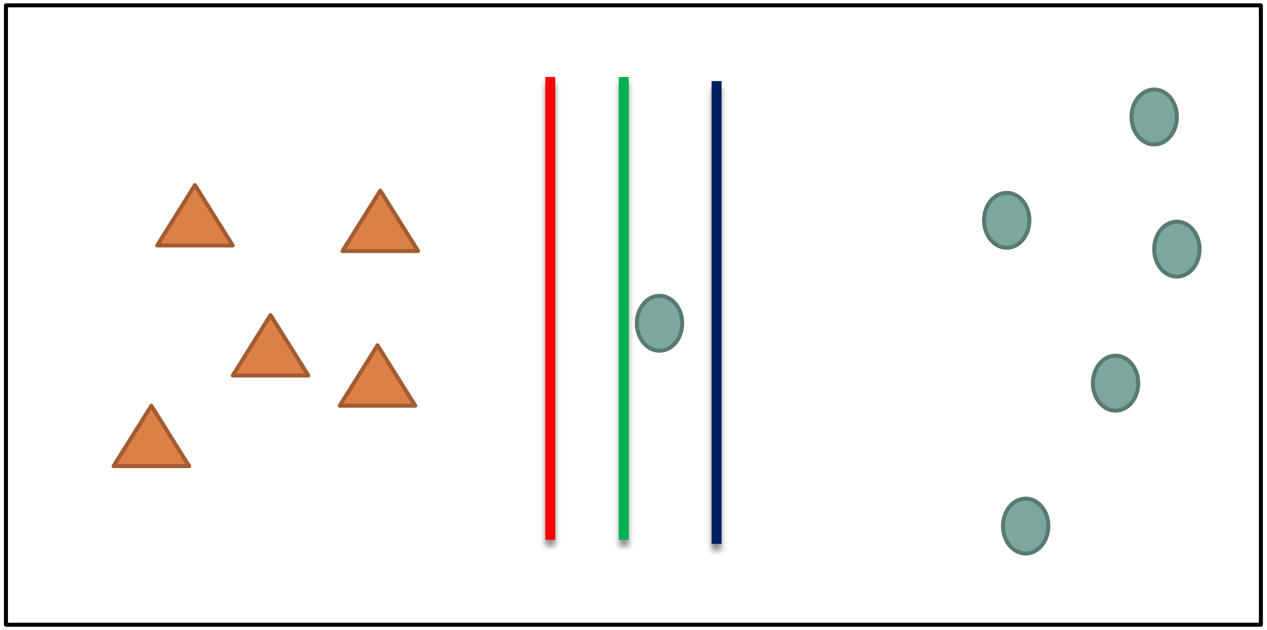
Hinge function used in SVM

- $\Delta = 1$ (one point penalized)



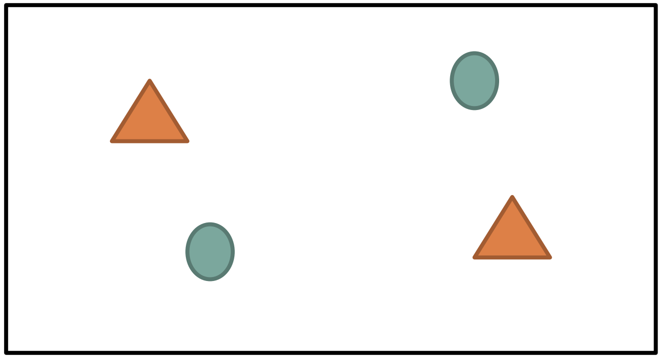
Intuition of hinge function

- Which boundary line is better (R,G, or B)?
  - Without hinge function, we choose R
  - With hinge function, we choose B



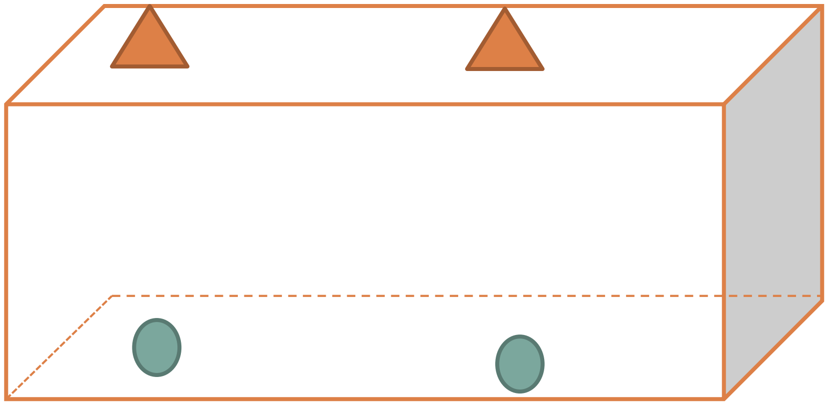
Kernel tricks

- To deal with non-linearly separable problem, we may map data points to higher dimensional space
- Data points below are not linearly separable



Kernel tricks

- Transform into higher dimensional space, and they are linearly separable



Kernel tricks

- With this idea, what we can do is to use a function $\phi(\cdot)$ to perform the mapping
- Therefore, we use the following equation instead

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

XOR problem

- Truth table of exclusive OR (XOR)

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

XOR problem

- Not possible for linear separation in original 2-D domain (obvious)
- If we map the data points to higher dim space, linear separation becomes possible
- Want to do it from 2-D to 3-D

Simple kernel trick

- Let $\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 x_2 \\ x_1 \\ x_2 \end{bmatrix}$
- We then have four data points as

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

- We can have a linear function to separate samples

Simple kernel trick

- Let $f(\mathbf{x}) = \begin{bmatrix} -2 & 1 & 1 \end{bmatrix} \cdot \phi(\mathbf{x}) - 0.5$
- We then have
  - $f\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) = 0.5, f\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) = 0.5 > 0$
  - $f\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = -0.5, f\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) = -0.5 < 0$
- As $f(\mathbf{x})$ is a linear function, it is linearly separable

Kernel tricks

- In our previous example, we use the following equation to perform classification

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

- However, we want
  - Not to computing $\phi(\mathbf{x})$ and $\mathbf{w}^T$ explicitly
    - It can be solved by **dual form**
  - A systematic way to find kernels (Mercer's theorem, omitted)

Dual form

- Recall the original problem (**primal form**)

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum \max\left(0, 1 - d_{(k)} y_{(k)}\right)$$

- This is equivalent to the following **dual form**

$$\max_{\lambda} \sum \lambda_{(k)} - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_{(i)} \lambda_{(j)} d_{(i)} d_{(j)} \left(\mathbf{x}_{(i)}^T \mathbf{x}_{(j)}\right)$$

Subject to $0 \leq \lambda_{(k)} \leq C$ and $\sum_k \lambda_{(k)} d_{(k)} = 0$

- The math details of dual form is omitted (KKT condition)

Dual form

- When using kernels, we use $\phi(\mathbf{x})$ in place of $\mathbf{x}$
- Therefore, in dual form we need to deal with the term $\left(\phi\left(\mathbf{x}_{(i)}\right)^T \phi\left(\mathbf{x}_{(j)}\right)\right)$
- Some functions have special properties

$$\phi\left(\mathbf{x}_{(i)}\right)^T \phi\left(\mathbf{x}_{(j)}\right) = \kappa\left(\mathbf{x}_{(i)}, \mathbf{x}_{(j)}\right)$$

- Thus, the computation is simplified even if $\phi(\mathbf{x})$ is in a very high dimensional space

Radical basis function

- One widely used kernel is **radical basis function**

$$\kappa\left(\mathbf{x}_{(i)},\mathbf{x}_{(j)}\right)=\exp\left(-\frac{\left\|\mathbf{x}_{(i)}-\mathbf{x}_{(j)}\right\|^2}{2\sigma^2}\right)$$

- Theoretically, $\phi\left(\mathbf{x}_{(j)}\right)$ with exponential function has **infinitely many** dimensions (**Taylor expansion** for exponential function, or more mathematically Hilbert space)
- Recall $\phi\left(\mathbf{x}_{(j)}\right)^T\phi\left(\mathbf{x}_{(j)}\right)$ is computing inner product, and thus it is a scalar

Support vector machine

- Recall in dual form we do not compute the weights
- How to use SVM if we find the optimal solution in the dual form

$$f(\mathbf{x}) = \sum_{j=1}^{N} \lambda_{(j)} d_{(j)} \kappa(\mathbf{x}_{(j)}, \mathbf{x}) + b$$

Could be zero

Only need to consider support vectors

Computing SVM

- It is not too difficult to find the optimal solution with linear kernel using Lagrange multipliers (for small problems)
- For complicated problems, we can find numerical solution by **SMO** method
- You can read the following paper to have some ideas about how it is done

https://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf

Using SVM

- One widely used SVM library is Libsvm developed by an NTU professor
- To use SVM with RBF (radical basis function) kernel, we need to determine following hyperparameters
  - σ^2 (sometimes use $\gamma = \frac{1}{2\sigma^2}$)
  - C in the cost function
- Therefore, again, use validation to fine-tune parameters

Using SVM

- One possible method to find hyperparameters is through grid search
  - To reduce the search space, use exponential increment in parameter values
  - For example: $C = 2^{-5}, 2^{-3}, \dots, 2^{15}$,

$$\gamma = 2^{-15}, 2^{-13}, \dots, 2^{3}$$

Using SVM

- If training data are class imbalanced, need to adjust other parameters to reduce its impact
- One more thing: sometimes it is useful to perform normalization on training (and then testing) data

Using SVM

- The original SVM is a binary classifier
  - Able to classify two classes
- We may extend SVM to Multiclass classification (used in sklearn SVM tool)
  - One vs all
  - One vs one
- Don't have time to cover the detail
  - Refer to supplementary material (multiclass classification)

What is not covered

- SVM for regression
- SVM for multiple classes
- SMO algorithm
- Variations of SVM (read textbook or other reference materials)