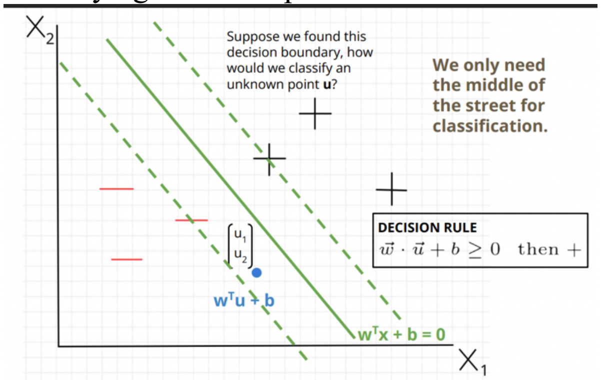
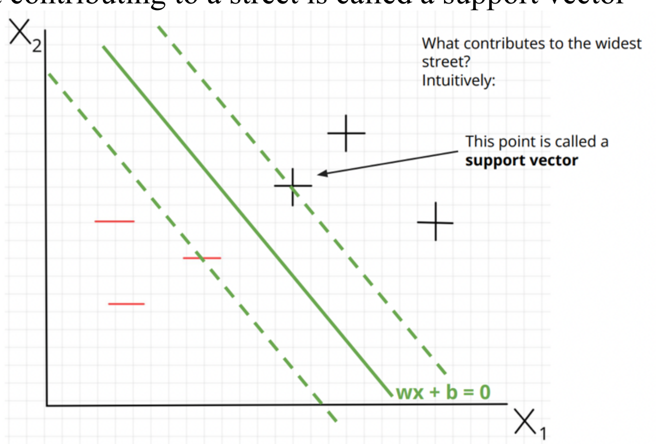
Lecture 13 Support Vector Machines (SVM)

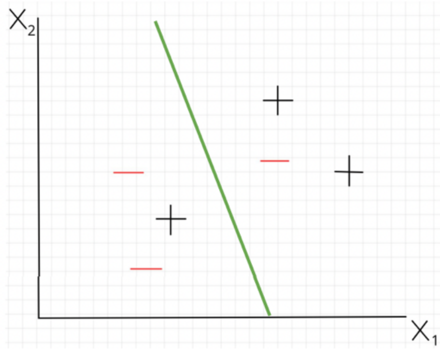
- SVM : find the widest street that separates out classes
- Goal 1: data points should be classified correctly
- Goal 2: Data points should not be in the "street"
  - The dotted line in the middle is the decision boundary
  - How do we find this street?
    - What is the format of the equation of this line / decision boundary
      - w1x1+w2x2+b=0
      - w^Tx+b =0
    - rule for classifying unknown point u
      - 

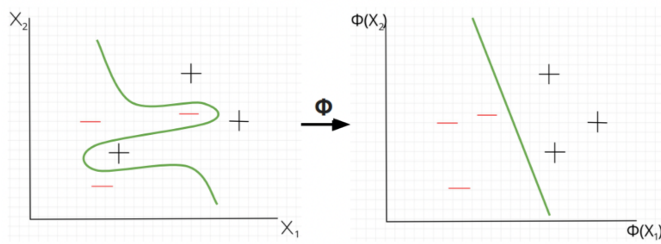
      - Only need to know the middle of the street for classification
    - When there are a significant amount of w's and b's : c*w^T + c*b =0
    - When c>1
      - you're emphasizing correct classification of training examples more than margin maximization
      - The model penalizes misclassification more heavily.
      - The optimization focuses on minimizing the classification error more than maximizing the margin.
      - This leads to a smaller margin and fewer margin violations
      - It may overfit the training data, especially if it's noisy, because it's trying hard to avoid any misclassification.
    - When c=1
      -
    - When 0<c<1
      - The model is more tolerant of classification errors
      - A smaller c places less penalty on misclassified points
      - This can lead to better generalization on unseen data, especially when the training set is noisy or not linearly separable.
      - The model becomes more robust to outliers.
    - When C -> 0
      - Maximize margin, **very tolerant** of errors → risk of **underfitting**

C Value	**Margin**	**Misclassification Tolerance**	**Risk**
C < 1	Wider	More tolerance	Underfitting
C = 1	Balanced	Moderate tolerance	Usually good balance
C > 1	Narrower	Less tolerance	Overfitting

- Picking the street
  - Full Algorithm (Perceptron Algorithm)
    - Start with random line w1 x1 + w2 x2 + b = 0 ●
    - Define:
      - ○ A total number of iterations (ex: 100)
      - ○ A learning rate a (not too big not too small)
      - ○ An expanding rate c (< 1 but not too close to 1)
    - Repeat number of iterations times
      - Pick a point (xi , yi) from the dataset
      - If correctly classified: do nothing
      - If incorrectly classified
        - Adjust w1 by adding (yi * a * x1), w2 by adding (yi * a * x2), and b by adding (yi * a)
        - Expand or retract the width by c (multiply
  - A point contributing to a street is called a support vector
    - 

    - We want samples to lie on the outside of the street so that
      - $$\vec{w} \cdot \vec{x}_+ + b \geq 1$$ $$\vec{w} \cdot \vec{x}_- + b \leq -1$$
    - For an unknown u,
      - $$-1 < \vec{w} \cdot \vec{u} + b < 1$$
  - w is inversely proportional to the width of the street
    - we want to maximize the width
    - $$\max\left(\frac{2}{\|\vec{w}\|}\right)$$
- What if there's no line?

- Option 1: soft margins
    - Can allow for some points in the dataset to be misclassified
    - 

- Option 2 : change perspective
    - Find a transformation phi to make the below possible
    - 

        - How to find phi?

$$\boxed{\sum_i \alpha_i < x_i, x > +b \geq 0 \quad \text{then } +}$$

            -
            - Using this ^ formula we only need to define a Kernal function to retrieve the phi transformation

$$K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$$

Called a Kernel function. This is often referred to as the "kernel trick".

$$\boxed{\sum_i \alpha_i K(x_i, x) + b \geq 0 \quad \text{then } +}$$

            -
    - Kernel Function
        - The inner product of a space describes how close / similar points are
        - Kernel Functions allow for specifying the closeness / similarity of points in a hypothetical transformed space
        - The hope is that with that new notion of closeness, points in the dataset are linearly separable.
        - https://medium.com/@gallettilance/support-vector-machines-16241417ee6d
        -