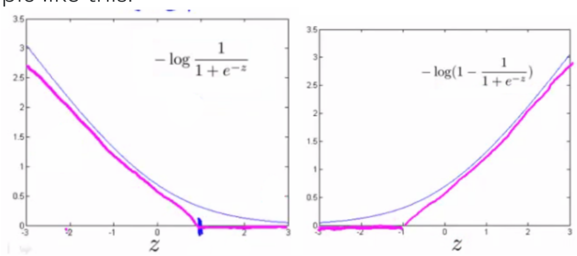
machine learning_week7

Created: 18/11/23 15:02 **Updated:** 18/11/24 08:57

Support Vector Machine

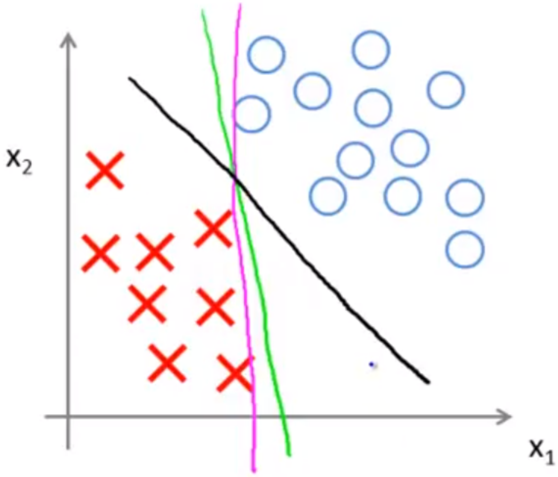
Large Margin Classification

- Optimization Objective
    - recall logistic regression. we have the cost function
        - $$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} -y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))$$
          $$= \frac{1}{m}\sum_{i=1}^{m} -y^{(i)}\log\left(\frac{1}{1+e^{-\theta^T x^{(i)}}}\right) - (1-y^{(i)})\log\left(1-\frac{1}{1+e^{-\theta^T x^{(i)}}}\right)$$
        - what we want are:
        - if y=1, then $h_\theta(x) \approx 1$ and $\Theta^T x \gg 0$

          if y=0, then $h_\theta(x) \approx 0$ and $\Theta^T x \ll 0$
    - make a **support vector machine**
        - use a straight decreasing line instead of the sigmoid curve
            - in place log() with :
            - $$z = \theta^T x$$
              $$\mathrm{cost}_0(z) = \max(0, k(1+z))$$
              $$\mathrm{cost}_1(z) = \max(0, k(1-z))$$
            - k is an constant defining the magnitude
            - pic like this:
            - 

        - use factor C instead of λ
            - $$J(\theta) = \boxed{C}\sum_{i=1}^{m} y^{(i)}\ \mathrm{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\ \mathrm{cost}_0(\theta^T x^{(i)}) + \frac{1}{2}\sum_{j=1}^{n}\Theta_j^2$$
        - in SVM, we don't output probability but output prediction in only 1 or 0.
            - $$h_\theta(x) = \begin{cases} 1 & \text{if } \Theta^T x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$
- **Large Margin Classifier**
    - see here. By let ⩾1 be condition, intuitively if could let decision boundary has a large margin.
        - If y=1, we want $\Theta^T x \geq 1$ (not just ≥0)

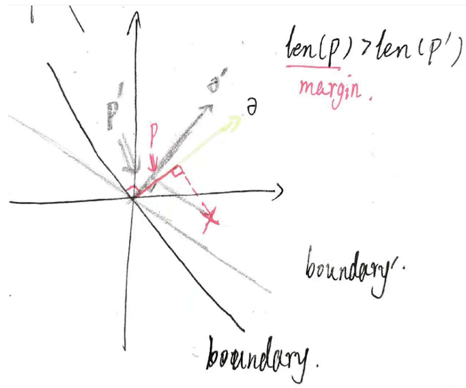
          If y=0, we want $\Theta^T x \leq -1$ (not just <0)
    - it is **as far away as possible** from both the positive and the negative examples.

- pic
  - 

  - black boundary is given by SVM.
  - noted that this intuition just happen when C is setting really large.
- **mathematical explanation**
  - **mathematical concept**
    1. the inner product of vector [X,Y]
       - $$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$
         $$[\boldsymbol{x}, \boldsymbol{y}] = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$
    2. The **projection** of vector
    3. The length of vector x, which is denoted ||x||.
       - $$\|\boldsymbol{x}\| = \sqrt{[\boldsymbol{x}, \boldsymbol{x}]} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$
  - explanation:
    - • p= length of projection of v onto the vector u.
      • $u^T v = p \cdot ||u||$

      Note that $u^T v = ||u|| \cdot ||v|| \cos\theta$ where θ is the angle between u and v. Also, $p = ||v|| \cos\theta$. If you substitute p for $||v|| \cos\theta$, you get $\boxed{u^T v = p \cdot ||u||.}$
    - use θ and X replace u and v, we get
    - We can use the same rules to rewrite $\Theta^T x^{(i)}$:

      $$\Theta^T x^{(i)} = p^{(i)} \cdot ||\Theta|| = \Theta_1 x_1^{(i)} + \Theta_2 x_2^{(i)} + \cdots + \Theta_n x_n^{(i)}$$
    - further, see the regularized part:
    - $$\begin{aligned} & \min_{\Theta} \frac{1}{2} \sum_{j=1}^{n} \Theta_j^2 \\ & = \frac{1}{2} (\Theta_1^2 + \Theta_2^2 + \cdots + \Theta_n^2) \\ & = \frac{1}{2} (\sqrt{\Theta_1^2 + \Theta_2^2 + \cdots + \Theta_n^2})^2 \\ & = \frac{1}{2} ||\Theta||^2 \end{aligned}$$
    - so for each $\Theta^T x^{(i)} = p^{(i)} \cdot ||\Theta||$



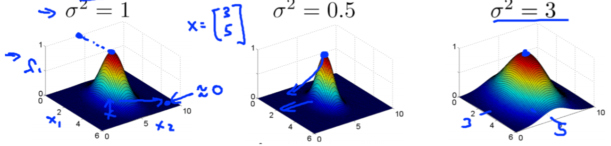
  - because we want ||Θ|| as small as possible, so SVM try to find a larger len(P), which led to large margin.

Kernels

- **understanding:**
  - sign some **landmarks**
  - find the **similarity** of landmark(i) and examples.
  - to find it, we need to "map" them into a different dimension
- $$f_i = similarity(x, l^{(i)}) = \exp\left(-\frac{||x - l^{(i)}||^2}{2\sigma^2}\right)$$
  - Gaussian Kernel, can be re-write below:
  - $$f_i = similarity(x, l^{(i)}) = \exp\left(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(i)})^2}{2\sigma^2}\right)$$
  - δ here have the same meaning in normal distribution.
  - **Example:**

    $l^{(1)} = \boxed{\begin{bmatrix} 3 \\ 5 \end{bmatrix}}$ $\quad f_1 = \exp\left(-\frac{||x - l^{(1)}||^2}{2\boxed{\sigma^2}}\right)$

    $\sigma^2 = 1$ $\quad x = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$ $\quad \sigma^2 = 0.5$ $\quad \sigma^2 = 3$

- how to measure similarity?
  - If $x \approx l^{(i)}$, then $f_i = \exp(-\frac{\approx 0^2}{2\sigma^2}) \approx 1$

    If x is far from $l^{(i)}$, then $f_i = \exp(-\frac{(large\ number)^2}{2\sigma^2}) \approx 0$
- **build feature vector**
  - get landmark: put them in the **exact same** locations as all the training examples.
  - vector here:
    - $$x^{(i)} \rightarrow \begin{bmatrix} f_1^{(i)} = similarity(x^{(i)}, l^{(1)}) \\ f_2^{(i)} = similarity(x^{(i)}, l^{(2)}) \\ \vdots \\ f_m^{(i)} = similarity(x^{(i)}, l^{(m)}) \end{bmatrix}$$
- replace X with f:
  - $$\min_\Theta C \sum_{i=1}^{m} y^{(i)} \text{cost}_1(\Theta^T \boxed{f^{(i)}}) + (1 - y^{(i)}) \text{cost}_0(\theta^T \boxed{f^{(i)}}) + \frac{1}{2} \sum_{j=1}^{n} \Theta_j^2$$

SVM Parameters

- **C:**
    - Choosing C (recall that $C = \frac{1}{\lambda}$
        - If C is large, then we get higher variance/lower bias
        - If C is small, then we get lower variance/higher bias
- **δ:**
    - large **:** high bias, low variance
    - small : high variance, low bias

Implementing SVM:

- use good SVM libraries
- need to specify:
    1. parameter **C**
    2. kernels:
        1. **No kernel** (linear kernel)
            - do not "map" into other dimension
            - perform like logistic regression, but use cost(Z) instead of log(h)
        2. **Gaussian** kernel
            - choose δ
            - suit for little features but large examples set.
        3. more
- remember:
    - perform **feature scaling**
    - kernel must satisfy "Mercer's Theorem"

---

what model?

- large n: logistic regression, or SVM without a kernel
- small n, intermediate m: SVM
- small n, large m: SVM may be slow. Manually create/add more features, then use logistic regression or SVM without a kernel.

neural network perform well in all the cases above.