7주차 - SVMs in Practice

- SVM software package

아무 kernel을 사용하지 않고 단순히 기존 특성값을 가지고 SVM에 적용하는 것을 'linear kernel'이라고 한다. 즉, 'SVM without kernel'인 것이다.(단순히 말해 가장 기본적인 SVM)
linear kernel이나 logistic regression인 경우 특성값이 데이터에 비해 상대적으로 작기 때문에 parameter optimization이 필요없다. 하지만 gaussian kernel과 그 외 kernel을 이용한 SVM을 사용할 경우에는 그런 고도의 최적화 방법이 필요하고 그런 방법은 이미 선구자들이 패키지로 만들어 놓았다. 그런 것을 응용하는 것이 효율적이다.
단, 그런 패키지를 이용한더라도 C와 분산을 어떻게 정할지와 kernel function(유사도 함수)를 결정해 주어야 한다. 그리고 SVM도 다른 회귀와 마찬가지로 특성값마다 범위가 크게 다른 경우 feature scaling을 적용한 뒤에 이용해야 한다.

Use SVM software package (e.g. liblinear, libsvm, ...) to solve for parameters θ.

Need to specify:

→ Choice of parameter C.

Choice of kernel (similarity function):

E.g. No kernel ("linear kernel")

Predict "y = 1" if $\theta^T x \geq 0$

$\theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \geq 0$

→ n large, m small

$x \in \mathbb{R}^{n+1}$

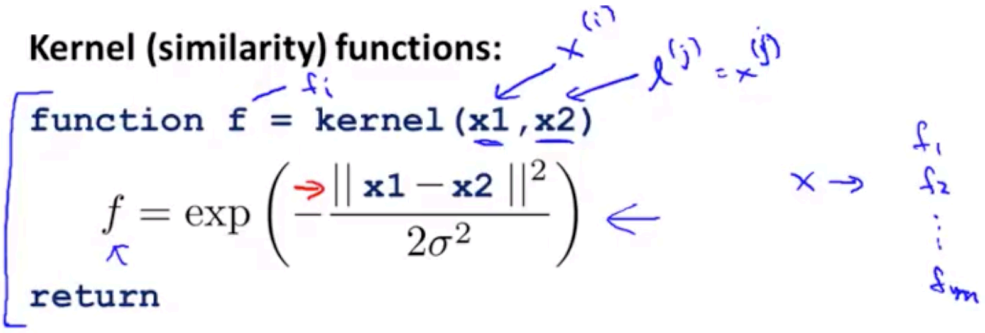
- Gaussian kernel:

$$f_i = \exp\left(-\frac{||x - l^{(i)}||^2}{2\sigma^2}\right), \text{ where } l^{(i)} = x^{(i)}.$$

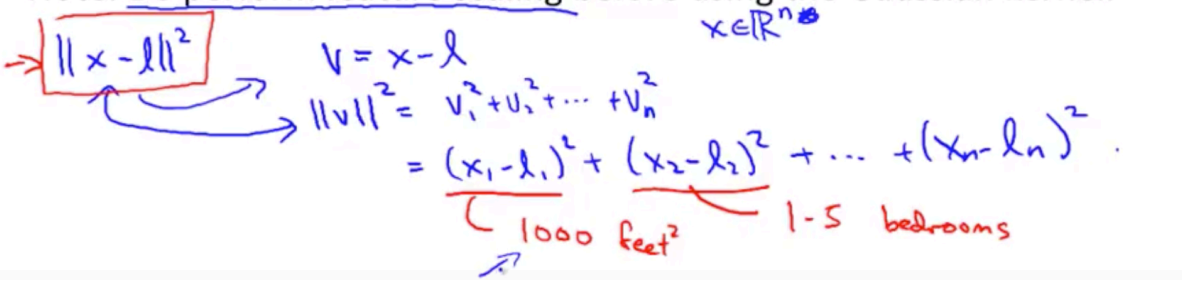
Need to choose σ^2.

$x \in \mathbb{R}^n$, n small and/or m large

Kernel (similarity) functions:



→ Note: Do perform feature scaling before using the Gaussian kernel.



- other choices of kernel

linear와 gaussian외에도 polynomial이나 복잡한 kernel들이 있으나 대부분 사용하는 kernel은 앞에 두 function이다. 문자열과 같은 특수한 데이터를 다룰 때에만 난해한 kernel을 접할 것이고 인생에 한두번 쓸 것이다.

Other choices of kernel

Note: Not all similarity functions $\text{similarity}(x, l)$ make valid kernels.
→ (Need to satisfy technical condition called "Mercer's Theorem" to make sure SVM packages' optimizations run correctly, and do not diverge).

Many off-the-shelf kernels available:

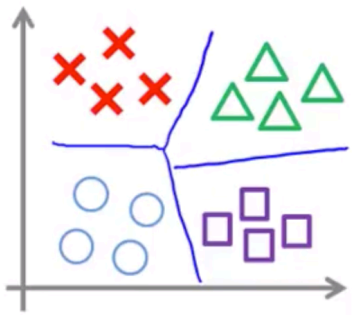
- Polynomial kernel: $k(x, l) = (x^T l)^2$, $(x^T l)^3$, $(x^T l + 1)^3$, $(x^T l + 5)^4$, $(x^T l + \text{constant})^{\text{degree}}$
- More esoteric: String kernel, chi-square kernel, histogram intersection kernel, ...

$sim(x, l)$

- Multi-class classification

SVM package를 사용하는 경우 대부분 다범주 분류를 지원한다. package를 사용하지 않을 경우는 이전에 logistic regression처럼 one-vs-all 방법을 사용하면 된다.

Multi-class classification



$y \in \{1, 2, 3, \ldots, K\}$

Many SVM packages already have built-in multi-class classification functionality.

→ Otherwise, use one-vs.-all method. (Train K SVMs, one to distinguish $y = i$ from the rest, for $i = 1, 2, \ldots, K$), get $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(K)}$

Pick class i with largest $(\theta^{(i)})^T x$

- 분류기로 무엇을 사용해야할까? (로지스틱 회귀 vs SVM vs NN)

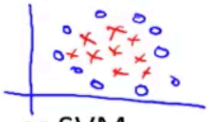
특징값을 n, 훈련 데이터 개수를 m이라고 했을 때,
아래와 같이 두 변수의 크기에 따라 잘맞는 알고리즘이 있다.
특성값 데이터에 비해 엄청 많은 경우(ex 스팸 분류기), 단순히 로지스틱 회귀나 linear kernel을 사용하는 것이 낫다. 왜냐하면 그외 kernel을 사용하게 되면 특성값의 개수가 m에 따라가기 때문에 오히려 줄기 때문이다.
데이터가 생각보다 있다하면 가우시안 커널을 사용한다.
한편, 데이터가 너무 많아지면 parameter 최적화(Theta ^ 2을 구하는 것)가 잘 작동하지 않는다.(한계에 부딪힌다) 따라서 이때는 그냥 다시 처음 경우처럼 로지스틱 회귀나 linear kernel을 사용하는 것이 바람직하다.
그리고 NN의 경우에는 단순히 unit과 층위를 조절하면 되기 때문에 어떤 경우에도 잘 작동한다. 다만 전체적으로 학습이 느리다는 점이있다.

하지만 주로 실생활에서 모델을 정확히 정하는 것은 그리 중요하지 않다고 했다. 오히려 데이터량 자체가 절대적으로 부족한 것이 문제가 될 수도 있다고 했다. 따라서 이것은 참고일 뿐 너무 엄격하게 받아들일 필요는 없다.

Logistic regression vs. SVMs

n = number of features ($x \in \mathbb{R}^{n+1}$), m = number of training examples

- If n is large (relative to m): (e.g. $n \geq m$, $n = 10{,}000$, $m = 10 \cdots 1000$)
  - Use logistic regression, or SVM without a kernel ("linear kernel")
- If n is small, m is intermediate: ($n = 1\text{-}1000$, $m = 10\text{-}10{,}000$)
  - Use SVM with Gaussian kernel



- If n is small, m is large: ($n = 1\text{-}1000$, $m = 50{,}000+$)
  - Create/add more features, then use logistic regression or SVM without a kernel
- Neural network likely to work well for most of these settings, but may be slower to train.