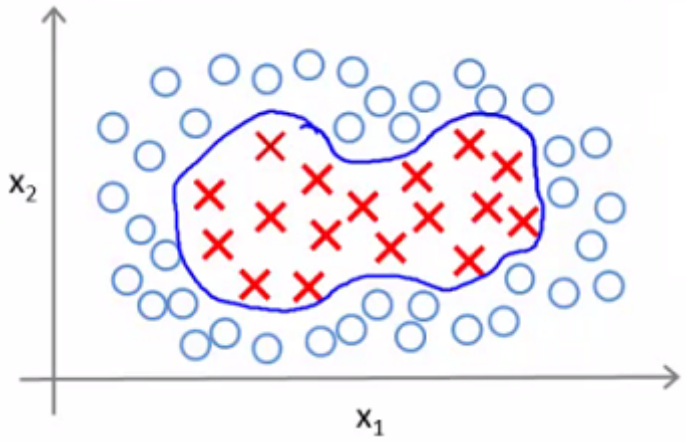
Kernel I: Adapting SVM to develop complex non-linear classifiers

We are going to start adapting support vector machines in order to develop complex nonlinear classifiers, the main technique for doing that is something called **kernels**, let's see what this kernels are and how to use them.

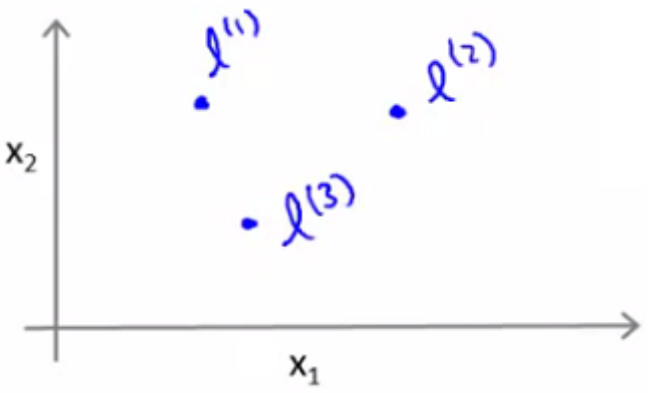
If we have a traninig set and we want to find a non-linear decision boundary, to distinguish the positive and negative examples:



One way to do so is to come up with a set of complex polynomial features

- Predict $y = 1$ if $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \cdots \geq 0$
- So that we end up with a hypothesis:
  - $h_\theta(x) = \begin{cases} 1 & if \quad \theta_0 + \theta_0 x_1 + \cdots \geq 0 \\ 0 & otherwise \end{cases}$
- Another way of writing this (new notation) is that a hypothesis computes a decision boundary by taking the sum of the parameter vector multiplied by a new feature vector f, which simply contains the various high order x terms.
  - $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \ldots$
  - $f_1 = x_1, f_2 = x_2, f_3 = x_1 x_2, f_4 = x_1^2, f_5 = x_2^2, \ldots$
  - i.e. not specific values, but each of the terms from our complex polynomial function
- Is there a different / better choice of the features $f_1, f_2, f_3, \ldots$?
- As we saw with computer vision, when the input is an image with lots of pixels, we also saw how using high order polynomials become very computationally expensive

Here is one idea for how to define features f_1, f_2, f_3, we're going to define only three new features (but for real problems we can get to define much a larger number), ignore the intercept x_0. Have a graph of x_1 vs. x_2 (we don't plot the values, just define the space), and we pick three points in that space:

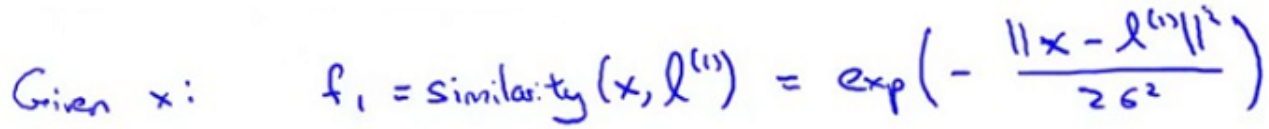


Given x, compute new features depending on proximity to landmarks $l^{(1)}$, $l^{(2)}$, $l^{(3)}$

- These points $l^{(1)}$, $l^{(2)}$, and $l^{(3)}$, were chosen manually and are **called landmarks**.

What we're going to do is define our new features as follows:

- Given x, define f_1 as the similarity between $(x, l^{(1)})$
  - $f_1 = \text{similarity } (x, l^{(1)}) = \exp(-\frac{\|x-l^{(1)}\|^2}{2\sigma^2})$



- $\|x - l^{(1)}\|$ is the euclidean distance between the point x and the landmark $l^{(1)}$ squared
- If we remember our statistics, we know that:
  - σ is the standard deviation
  - σ^2 is commonly called the variance
- Remember, that as discussed:
  - $\|x - l^{(2)}\|^2 = \sum_{j=1}^{n}(x_j - l_j^{(1)})^2$
- $f_2 = \text{similarity } (x, l^{(2)}) = \exp(-\frac{\|x-l^{(2)}\|^2}{2\sigma^2})$
- $f_3 = \text{similarity } (x, l^{(3)}) = \exp(-\frac{\|x-l^{(3)}\|^2}{2\sigma^2})$

This similarity function is called a kernel (this function is a Gaussian Kernel).

- Sometimes, instead of writing similarity between x and l we might write $k(x, l^{(i)})$

So let's see what this kernel actually do and why these sorts of similarity functions might make sense.

Kernels and Similarity (diving deeper into the kernel)

So let's take my first landmark $l^{(1)}$, so the similarity of the kernel between x and $l^{(1)}$ is given by the following expression: $f_1 = \text{similarity } (x, l^{(1)}) = \exp(-\frac{\|x-l^{(1)}\|^2}{2\sigma^2})$.

Just to make sure, the numerator can also be written as: $\exp(-\frac{\sum_{j=1}^{n}(x_j-l_j^{(1)})^2}{2\sigma^2})$, and again for the purpose of this case we're ignoring the intercept term x_0 that is always equals to one, so this is how we compute the kernel with similarity between x and a landmark.

So let's see what this function does:

- If $x \approx l^{(1)}$:
  - We suppose x is close to one of the landmarks $l^{(1)}$. Then the euclidean distance formula and the numerator will be close to 0: $f \approx \exp(-\frac{0}{2\sigma^2}) \approx e^{-0} \approx 1$
- If x if far from $l^{(1)}$
  - We suppose x is for to one of the landmarks $l^{(1)}$. Then the euclidean distance formula and the numerator will be a large number: $f \approx \exp(-\frac{(\text{large number})^2}{2\sigma^2}) \approx 0$

Concretely what these features do is they measure how similar x is from one of our landmarks and the feature f is going to be close to 1 when x is close to our landmark and is going to be 0 or close to 0 when x is far from our landmark:

- If x is close to our landmark $f \approx 1$
- If x is far from our landmark $f \approx 0$

So each landmark defines a new features: $l^{(1)} \rightarrow f_1$, $l^{(2)} \rightarrow f_2$, $l^{(3)} \rightarrow f_3$, given the training example x, we can now compute three new features: f_1, f_2, f_3.

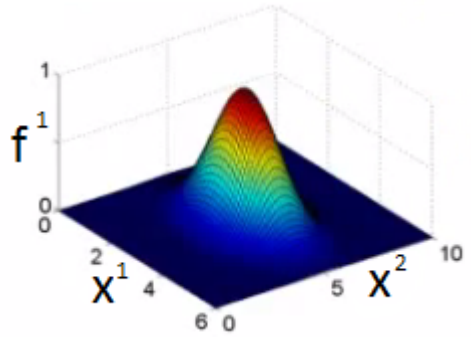
Let's look at this exponentiation function and plot in some figures to understand better what this really looks like.

Example

For this example, let's say I have two features x_1 and x_2, and let's say that our first landmark $l^{(1)}$ is at location 3, 5:

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

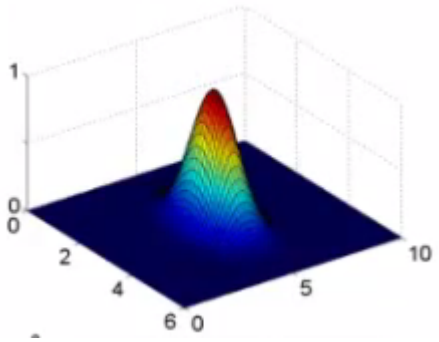
If we plot f_1 vs the kernel function we get a plot like this:



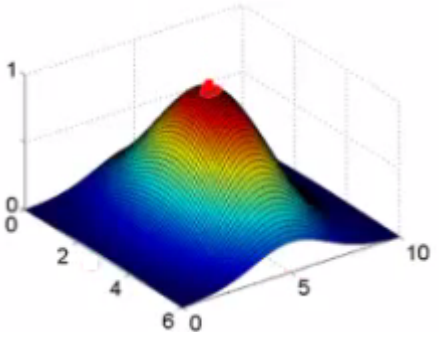
- Notice that when $x = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$ then $f_1 = 1$
- As x moves away from $\begin{bmatrix} 3 \\ 5 \end{bmatrix}$ then the feature takes on values close to zero
- So this measures how close x is to this landmark, where $\sigma^2 = 1$

What are the effects of varying the parameter σ^2?

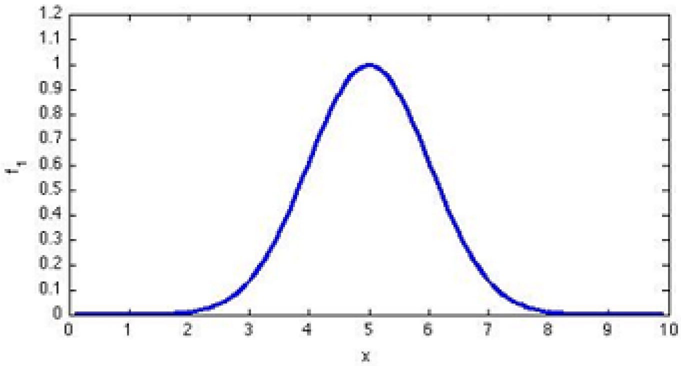
- σ^2 is a parameter of the Gaussian kernel
  - Defines the steepness of the rise around the landmark
- Below example where $\sigma^2 = 0.5$

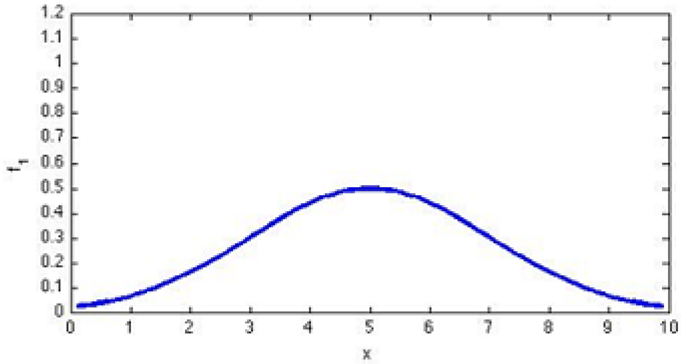


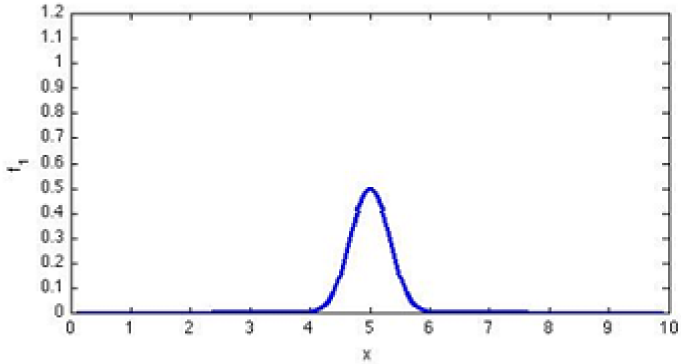
We see here that as we move away from 3,5 the feature f_1 falls to zero much more rapidly. Conversely, if we has increase $\sigma^2 = 3$, if sigma squared is large, then as we move away from $l^{(1)}$, the value of the feature falls away much more slowly.

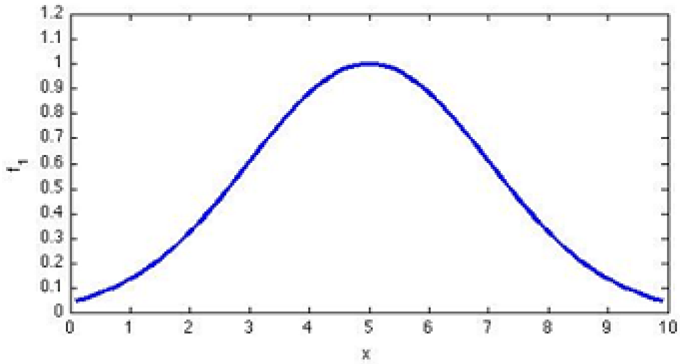


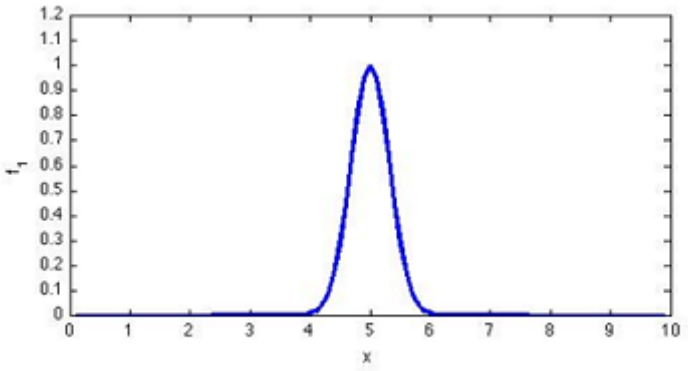
Video Question: Consider a 1-D example with one feature x_1. Suppose $l^{(1)} = 5$. To the right is a plot of $f_1 = \exp(-\frac{\|x_1 - l^{(1)}\|}{2\sigma^2})$ when $\sigma^2 = 1$. Suppose we now change $\sigma^2 = 4$. Which of the following is a plot of f_1 with the new value of σ^2?







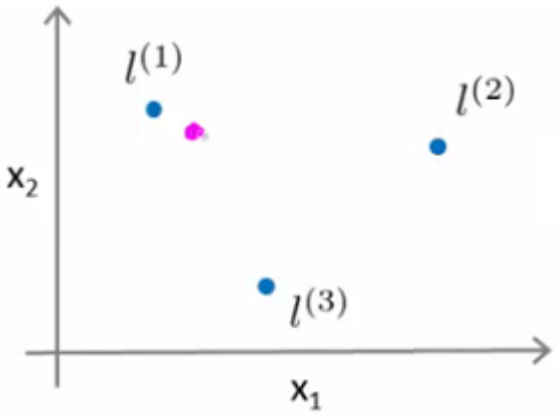




Given this definition, what kinds of hypotheses can we learn?

- With training examples x we predict "1" when:
  - $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$
- For our example, lets say we've already run an algorithm and got the and somehow we ended up with these values of the parameter
  - $\theta_0 = -0.5, \theta_1 = 1, \theta_2 = 1, \theta_3 = 0$

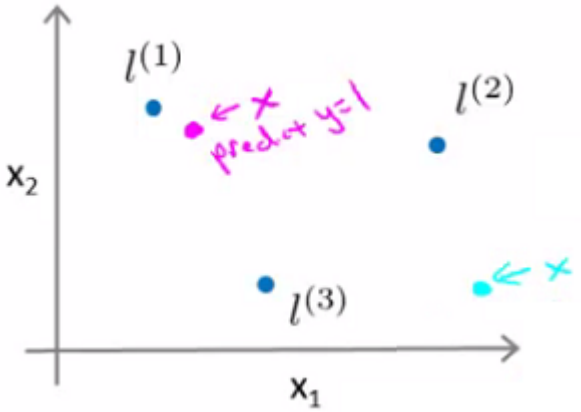
Given our placement of three examples, what happens if we evaluate an example at the magenta dot below?



Looking at our formula, we know f_1 will be close to 1, but f_2 and f_3 will be close to 0.

- So if we look at the formula we have ($f_1 \approx 1, f_2 \approx 0, f_3 \approx 0$)
  - $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$
  - $\theta_0 + \theta_1 \times 1 + \theta_2 \times 0 + \theta_3 \times 0$
  - $-0.5 + 1 = 0.5 \geq 0$
- So at that point ($l^{(1)}$) we're going to predict $y = 1$

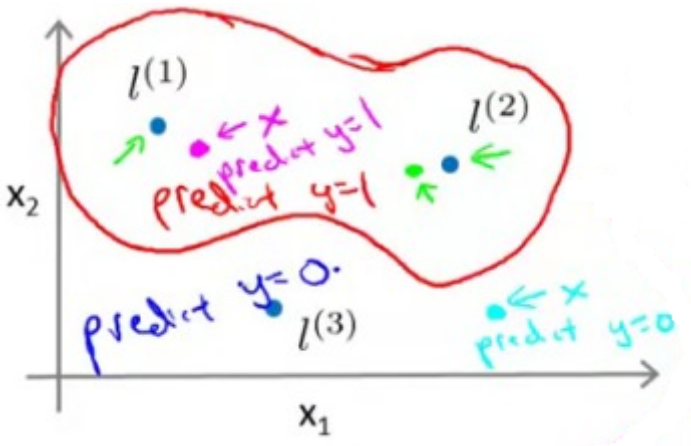
If we had another point (cian) far away from all three



If that point were our training example x, then if we make a similar computation we find that $f_1, f_2, f_3 \approx 0$

- $\theta_0 + \theta_1 f_1 + \cdots \approx -0.5 < 0$
- And so, at this point out there, we're going to predict $y = 0$

Considering our parameter, for points near $l^{(1)}$ and $l^{(2)}$ we predict 1, but for points near $l^{(3)}$ we predict 0. Which means we create a non-linear decision boundary that goes something like this:



- Inside we predict $y = 1$
- Outside we predict $y = 0$

So this show how we can create a non-linear boundary with landmarks and the kernel function in the support vector machine but:

- How do we get/chose the landmarks?
- What other kernels can we use (other than the Gaussian kernel)?