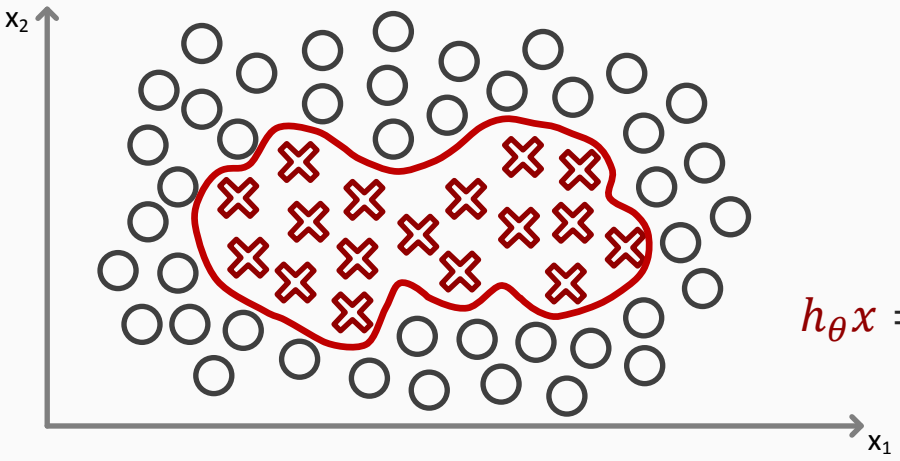
support vector machines kernels

kernels i & ii

nonlinear decision boundary



predict $y = 1$ if

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \cdots \geq 0$$

$$h_\theta x = \begin{cases} 1 \text{ if } \theta_0 + \theta_1 x_1 \ldots \text{ is } \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

the technique to adapt support vector machines to complex nonlinear classifiers is achieved through the use of kernels

an alternative method to write the above notation is as follows:

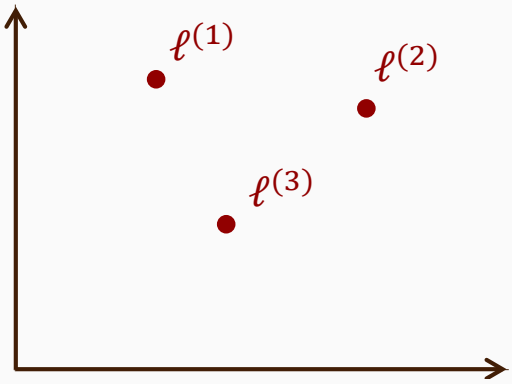
$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \theta_4 f_4 + \theta_5 f_5 + \cdots$$

$$f_1 = x_1,\ f_2 = x_2,\ f_3 = x_1 x_2,\ f_4 = x_1^2,\ f_5 = x_2^2,\ \ldots$$

the question is whether there is a better/alternative choice for features $f_1,\ f_2,\ f_3,\ \ldots$

Kernel example based on three manually chosen points $\ell^{(1)}$, $\ell^{(2)}$, $\ell^{(3)}$

given x, compute new feature depending on proximity to landmarks $\ell^{(1)}$, $\ell^{(2)}$, $\ell^{(3)}$



given x:

$$f_1 = \text{similarity}\left(x, \ell^{(1)}\right) = \exp\left(-\frac{\left\|x - \ell^{(1)}\right\|^2}{2\sigma^2}\right)$$

$$f_2 = \text{similarity}\left(x, \ell^{(1)}\right) = \exp\left(-\frac{\left\|x - \ell^{(1)}\right\|^2}{2\sigma^2}\right)$$

$$f_3 = \text{similarity}\left(x, \ell^{(1)}\right) = \exp(\ldots)$$

the functions above referred to as: **kernels** and **gaussian kernels**, denoted → $\mathrm{k}\left(x, \ell^{(i)}\right)$

kernels and similarity

the similarity between x and $\ell^{(1)}$ is given by the following expression:

$$f_1 = \text{similarity}(x, \ell^{(1)}) = \exp\left(-\frac{\left\|x - \ell^{(1)}\right\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{j=1}^{n}\left(x_j - \ell_j^{(1)}\right)^2}{2\sigma^2}\right)$$

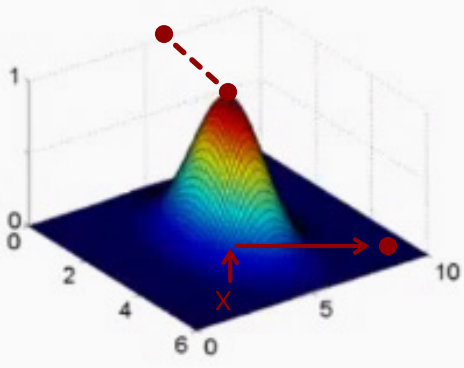
the numerator term is equal to the component wise distance between vector X and I

if $x \approx \ell^{(1)}$: $f_1 = \exp\left(-\frac{(0)^2}{2\sigma^2}\right) \approx 1$

if x is far from $\ell^{(1)}$: $f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0$

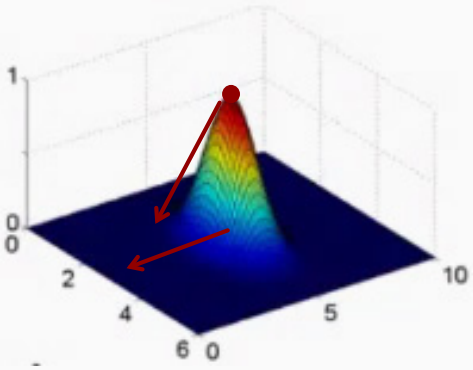
example

$\ell^{(1)} = \begin{bmatrix}3\\5\end{bmatrix}, f_1 = \exp\left(-\frac{\|x-\ell^{(1)}\|^2}{2\sigma^2}\right)$ and setting $\sigma^2 = 1$



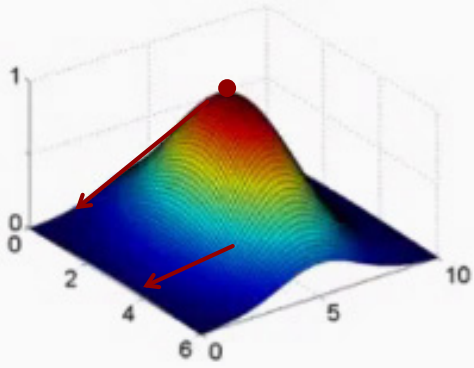
$\sigma^2 = 1$

the of a point x on the contour plot represents the value of f. when x = exactly, f = 1. when x moves farther away from $\ell^{(1)}$, f ≈ 0



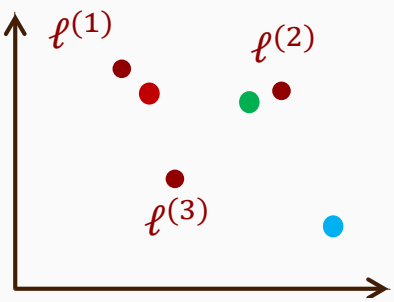
$\sigma^2 = 0.5$

as x moves away from $\ell^{(1)}$, the value of the feature falls away more quickly because the point of the contour is steep



$\sigma^2 = 3$

as x moves away from $\ell^{(1)}$, the value of the feature falls away more slowly because the point of the contour is wider



predict "1" when $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$

supposed outputs of θ: $\theta_0 = -0.5,\ \theta_0 = 1,\ \theta_0 = 1,\ \theta_0 = 3$

$f_1 \approx 1 \quad f_2 \approx 0 \quad f_3 \approx 0$

$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$ = -0.5 + 1 → 0.5 ≥ 0

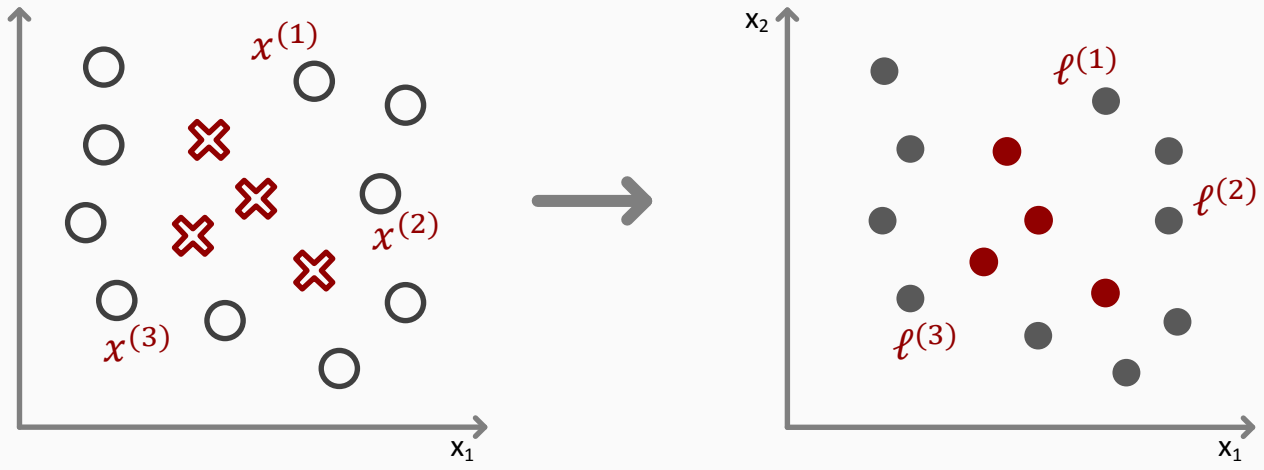
$f_1 \approx 0 \quad f_2 \approx 0 \quad f_3 \approx 0$

$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$ → -0.5 ≤ 0

green point has similar expected output as red training data

choosing the landmarks

initially place landmarks at exactly each point of the training examples in a dataset



support vector machines with kernels

given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

choose $\ell^{(1)} = x^{(1)}, \ell^{(2)} = x^{(2)}, \ldots, \ell^{(m)} = x^{(m)}$

given example x:

$$f_1 = \text{similarity}(x, \ell^{(1)})$$

$$f_1 = \text{similarity}(x, \ell^{(1)})$$

$$\vdots$$

grouped into vector f

$$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \qquad f_0 = 1$$

for training example $(x^{(i)}, y^{(i)})$:

$$f_1^{(i)} = \text{similarity}(x^{(i)}, \ell^{(1)})$$

$$f_2^{(i)} = \text{similarity}(x^{(i)}, \ell^{(2)})$$

$$\vdots \longrightarrow f_i^{(i)} = \text{similarity}(x^{(i)}, \ell^{(i)}) = \exp\left(-\frac{0}{2\sigma^2}\right) = \mathbf{1}$$

$$\vdots$$

$$f_1^{(m)} = \text{similarity}(x^{(i)}, \ell^{(m)})$$

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

$$f^{(i)} = \begin{bmatrix} f_1^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix}$$

$$f_0^{(i)} = 1$$

making predications given an obtained set of parameters of θ

hypothesis: given x, compute features $f \in \mathbb{R}^{m+1}$

predict "y=1" if $\theta^T f \geq 0$

in order to get the set of parameters of θ

training:

$$\min_{\theta} \frac{1}{m} C \sum_{i=1}^{m} y^{(i)} \text{cost}_1(\theta^T f^{(i)}) + \left(1 - y^{(i)}\right) \text{cost}_0\left(\theta^T f^{(i)}\right) + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

~~$\theta^T x^{(i)}$~~

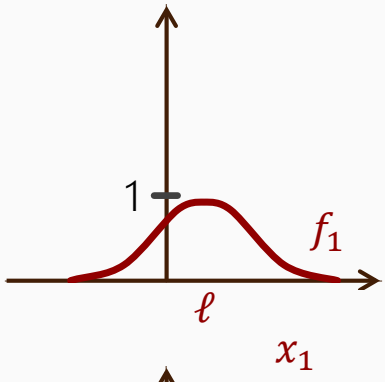
support vector machine parameters

c$\left(=\frac{1}{\lambda}\right)$ large C → **lower bias, higher variance** (small λ)

small C → **higher bias, lower variance** (large λ)

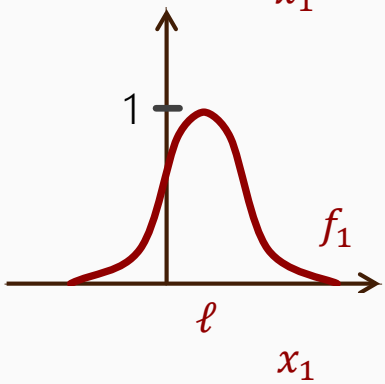
σ^2 large σ^2: features vary more smoothly

→ **higher bias, lower variance**



large σ^2: features vary less smoothly

→ **lower bias, higher variance**



during training an svm and the model is overfitting the training data. a reasonable step would be to either decrease C and/or decrease σ^2

svms in practice

using support vector machines

svm software packages (e.g. liblinear, libsvm, ...) to solve for parameters θ

specifications with a library package:

choice of parameter C

çhoice of kernel (similiarity function):

no kernel ("linear kernel") $\theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \geq 0$

predict "y=1" if $\theta^T f \geq 0$ n = **large**, m = **small** $x^{(i)} \in \mathbb{R}^{n+1}$

linear kernels are a good option when the dataset is large but only a small amount of training data. fitting a nonlinear function with n = **large,** m = **small** is likely to overfit

gaussian kernel

$f_i = \exp\left(-\frac{\|x-\ell^{(i)}\|^2}{2\sigma^2}\right)$ where $\ell^{(i)} = x^{(i)}$ $x^{(i)} \in \mathbb{R}^{n+1}$

need to choose σ^2 n = **small** and/or m = **large**

gaussian kernels are a good option when the dataset smaller and/or there is a substantial amount of training data. an svm can better fit a complex nonlinear function

kernel (similarity) functions

```
function f = kernel (x1,x2)
```

$$f_i = \exp\left(-\frac{\|x1 - x2\|^2}{2\sigma^2}\right)$$

```
return
```

it is important to perform feature scaling prior to using a gaussian kernel:

$\|x - \ell^{(1)}\|^2$ $v = x - \ell$ $x \in \mathbb{R}^{n+1}$

$$\|v\|^2 = v_1^2 + v_2^2 + \cdots + v_n^2$$

$$= (x_1 - \ell_1)^2 + (x_2 - \ell_2)^2 + \cdots + (x_n - \ell_n)^2$$

multiple scales → 10s 1,000,000s

features that take on vast ranges of values, the computation will suffer by overweighed functions with higher scales than others. this will ensure the svm applies the proper attention to each feature appropriately

other choices of kernel

note: not all similarity functions $\text{similarity}(x, \ell)$ make valid kernels. (need to satisfy technical condition called "mercer's theorem" to ensure svm packages' optimizations run correctly and do not diverge the theorem ensures that support vector machine software packages can use a large class of optimizations and arrive at the parameter θ quickly and efficiently

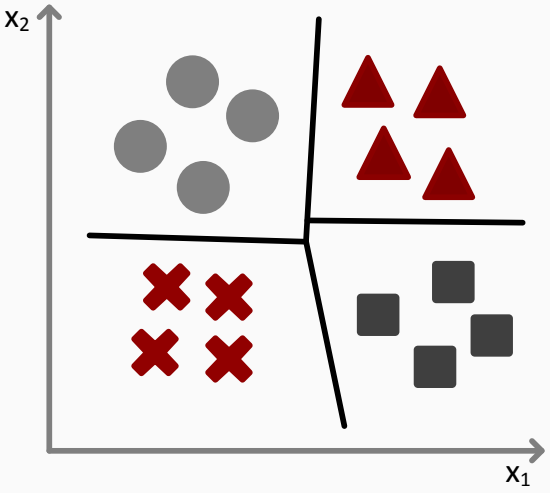
multiple available kernel packages:

polynomial kernel → $k(x, \ell) = (x^T \ell + \mathbf{0})^{\mathbf{2}}$ → $(x^T \ell + \mathbf{constant})^{\text{degree}}$

more esoteric: string kernel, chi-square kernel, histogram, intersection, …

under consideration of a kernel along with parameters such as **C, σ^2**, etc… the kernel choice should be influenced by the best **cross-validation dataset** performance

multiclass classification



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

many svm packages have built-in classification functionality

otherwise, utilize one-vs-all method (train K svms, one to distinguish 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 $\left(\theta^{(i)}\right)^T x$

logistic regression versus support vector machines

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…1000)
→ logistic regression, or svm without a kernel ("linear kernel")

if n is **small**, m is **intermediate**: (e.g. n= 1-1000, m = 10-10,000)
→ svm with a guassian kernel

if n is **small**, m is **large**: (e.g. n= 1-1000, m = 50,000+)
→ create/add more features, then apply logistics regression or svm without a kernel

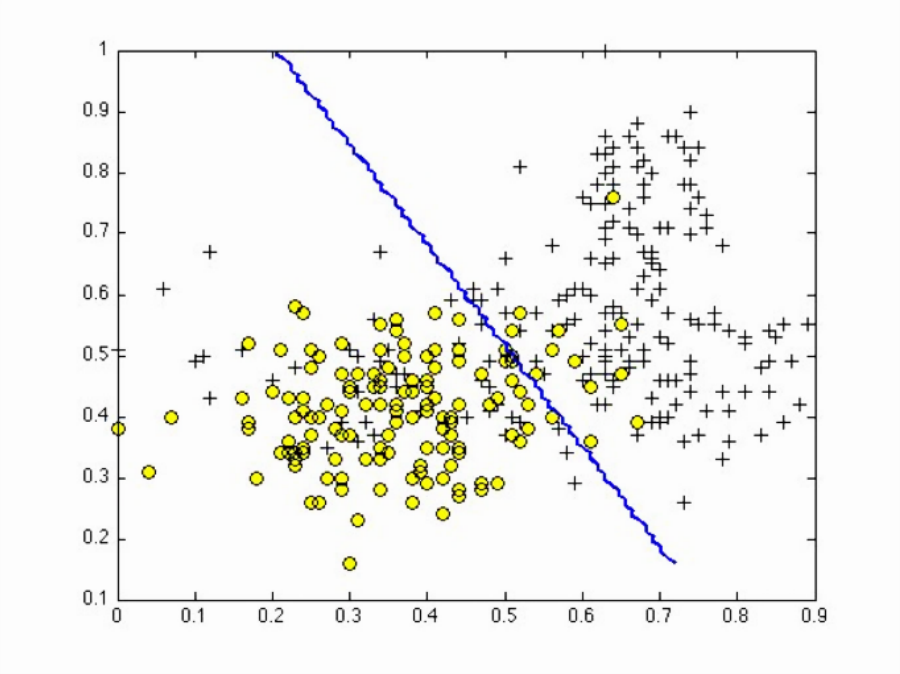
support vector machines and logistic regression are similar in effect but sometimes one will be more efficient or optimized for the data than the other

nueral networks will likely functions with most of the above settings but slower to train

the optimization problem that an svm has is a convex optimization problem; the software packages will consequentially always find the global minimum (or close to)

support vector machines

1. Suppose you have trained an SVM classifier with a Gaussian kernel, and it learned the following decision boundary on the training set:



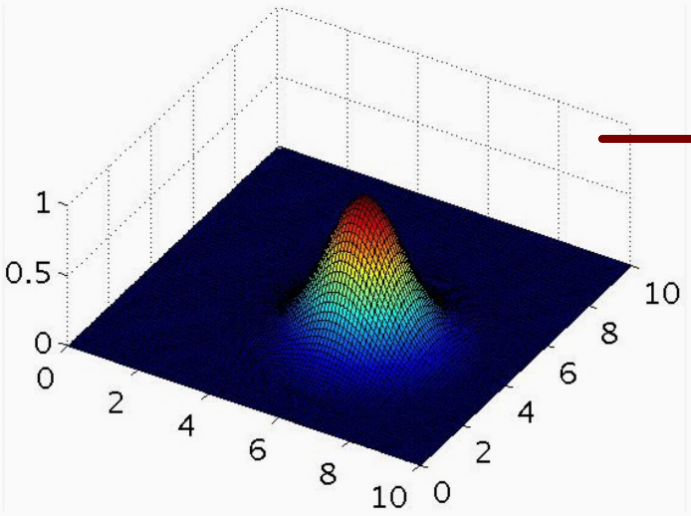
You suspect that the SVM is underfitting your dataset. Should you try increasing or decreasing C? Increasing or decreasing σ^2?

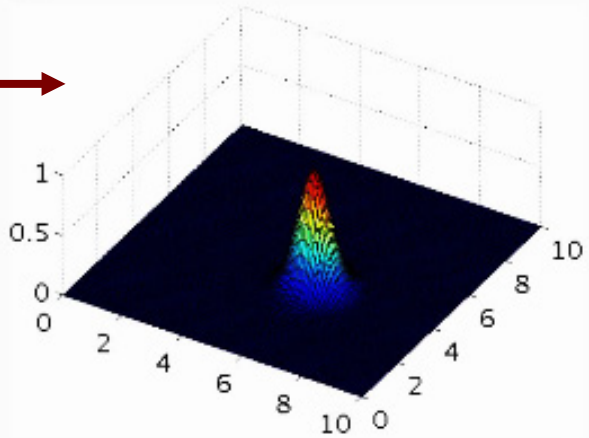
- It would be reasonable to try **increasing** C. It would also be reasonable to try **decreasing** σ^2.

The figure shows a decision boundary that is underfit to the training set, so we'd like to lower the bias / increase the variance of the SVM. We can do so by either increasing the parameter C or decreasing σ^2.

2. The formula for the Gaussian kernel is given by $\text{similarity}(x, l^{(1)}) = \exp\left(-\frac{||x-l^{(1)}||^2}{2\sigma^2}\right)$.

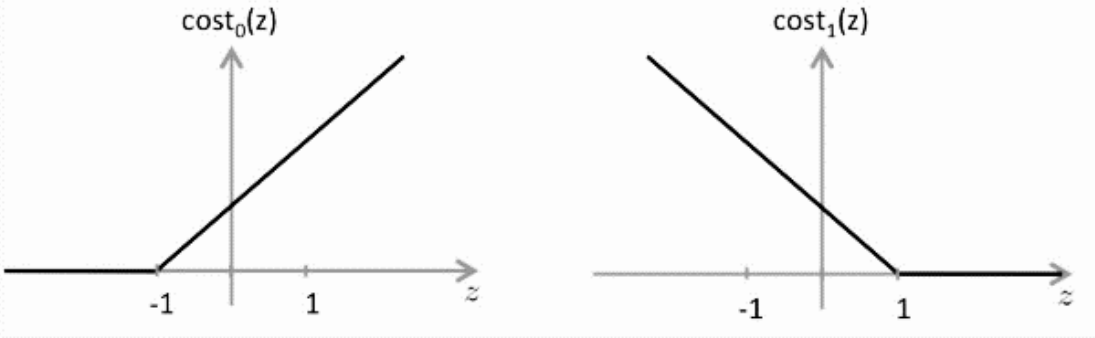
The figure below shows a plot of $f_1 = \text{similarity}(x, l^{(1)})$ when $\sigma^2 = 1$.





This figure shows a "narrower" Gaussian kernel centered at the same location which is the effect of decreasing σ^2.

The SVM solves $\min_\theta \; C\sum_{i=1}^{m} y^{(i)}\text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\text{cost}_0(\theta^T x^{(i)}) + \sum_{j=1}^{n}\theta_j^2$ where the functions $\text{cost}_0(z)$ and $\text{cost}_1(z)$ look like this:



The first term in the objective is: $C\sum_{i=1}^{m} y^{(i)}\text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\text{cost}_0(\theta^T x^{(i)})$.

This first term will be zero if two of the following four conditions hold true. Which are the two conditions that would guarantee that this term equals zero?

3.

for every example where $y^{(i)} = 1$, $\theta^T x^{(i)} \geq 1$
for every example where $y^{(i)} = 0$, $\theta^T x^{(i)} \leq -1$

4. Suppose you have a dataset with n = 10 features and m = 5000 examples.

After training your logistic regression classifier with gradient descent, you find that it has underfit the training set and does not achieve the desired performance on the training or cross validation sets.

try using a neural network with a large number of hidden units
create/add new polynomial features
reduce the number of examples in the training data

5. Which of the following statements are true? Check all that apply.

it is important to perform feature normalization before using a gaussian kernel
the maximum value of the gaussian kernel ($\text{similarity}(x, \ell^{(1)})$) is 1
2D input examples (i.e. $x^{(i)} \in \mathbb{R}^2$) plots a linear decision boundary as a straight line (linear kernel)