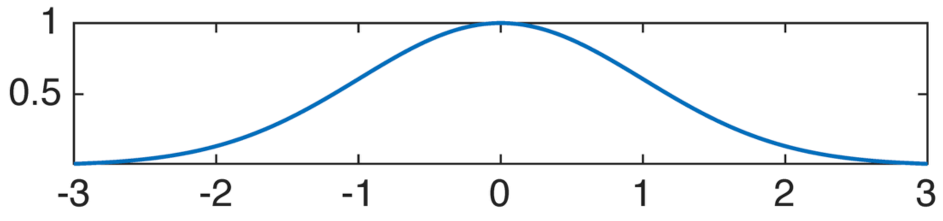
Example: radial basis functions

Since function spaces can be counterintuitive, it's good to keep some examples in mind; these examples can help us think about their properties. One good example is the grid-sampled functions we described above: these can be thought of either as vectors in $\mathbb{R}^n$, or as functions from the finite set $\{1 \ldots n\}$ to $\mathbb{R}$.

A second good example is a space spanned by Gaussian radial basis functions. A Gaussian radial basis function (also called a squared-exponential function) is a function of the form

$$q_{y,\sigma}(x) = \exp(\|y - x\|^2 / 2\sigma^2)$$

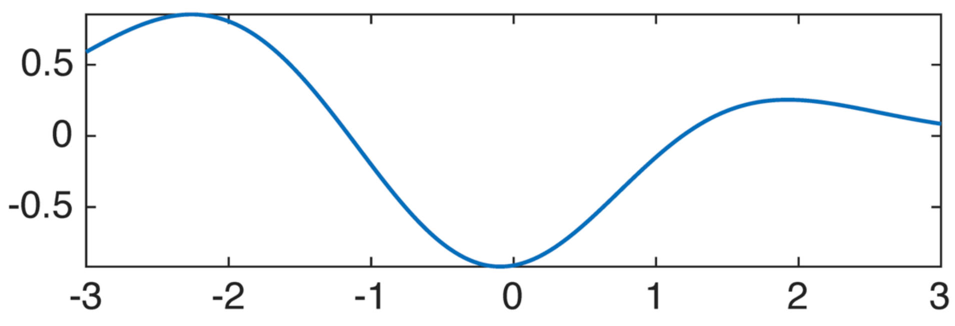
Here y and σ are parameters (part of the function definition), while x is the single real-valued argument. We call y the *center* and σ the *width* of the radial basis function. By convention, if we leave out σ, it is 1: that is, $q_y = q_{y,1}$. Here is q_0:



We can define a vector space H by taking the finite span of all Gaussian radial basis functions of a given width, say $\sigma = 1$. That is, a vector in H is a function of the form

$$\sum_{i=1}^{n} u_i q_{y_i}$$

where y_i is the center for the ith term in the sum. It looks like this:



(Each element of H can use a different set of centers; when we add together two vectors, we take the union of their sets of centers.)

We can upgrade H to an inner product space by defining an inner product for individual radial basis functions

$$\langle q_y, q_{y'} \rangle = q_y(y') = q_{y'}(y)$$

and then extending to functions in the finite span by linearity. For example,

$$\langle q_1, q_1 \rangle = \exp(0) = 1$$

$$\langle q_1, q_5 \rangle = \exp(-4^2/2) = e^{-8}$$

$$\langle q_1, q_1 + 3q_5 \rangle = 1 + 3e^{-8}$$

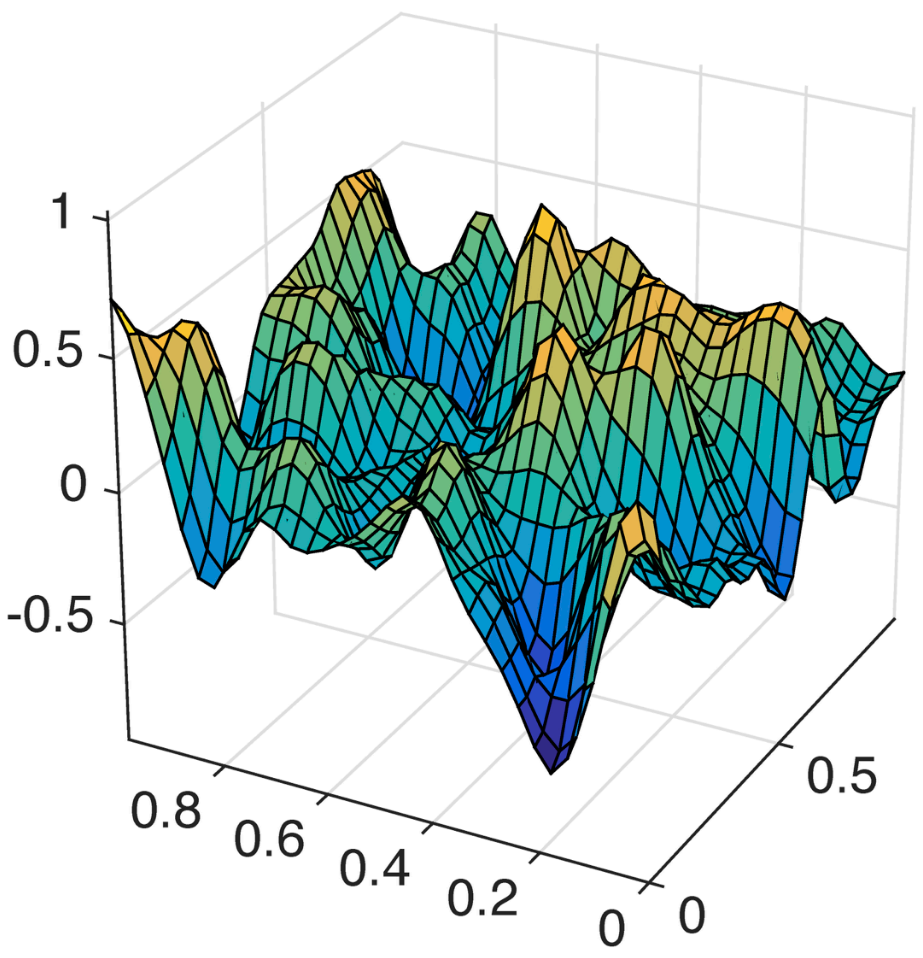
We won't do it here, but it's possible to show that this definition yields a valid inner product.

> *One route is to use Bochner's theorem together with the fact that the Fourier transform of a Gaussian is another Gaussian.*

The resulting inner product space is not complete, but we can complete it to get a new space $\bar{H}$. Again we won't give the details of how this happens, but the elements of $\bar{H}$ wind up being countable-length linear combinations of radial basis functions, with coefficient sequences that are square-summable.

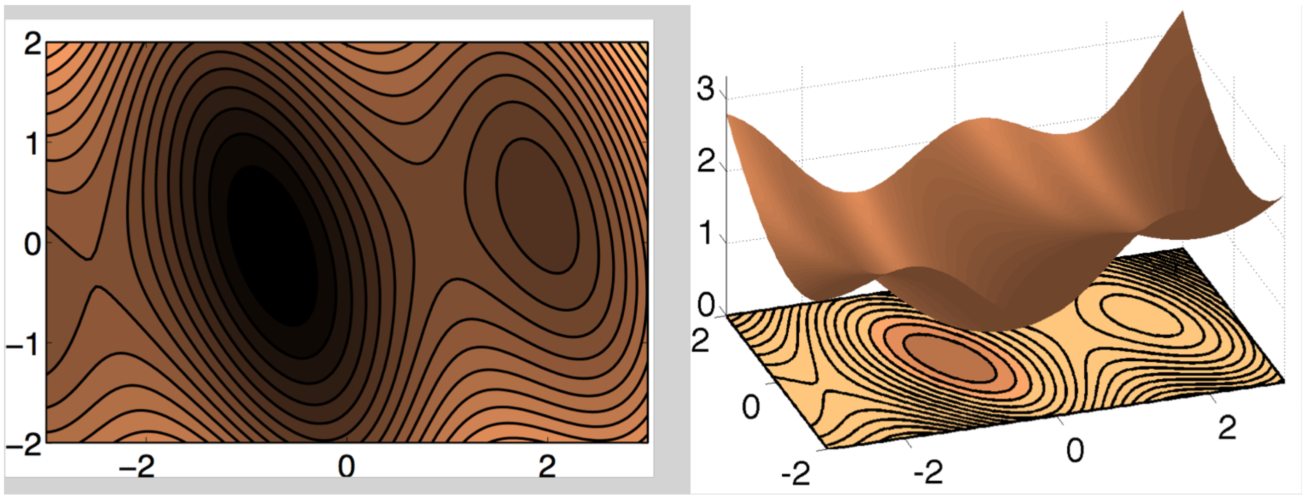
This new space $\bar{H}$ is very useful: like $\mathbb{R}^n$ it is a complete inner product space, but unlike $\mathbb{R}^n$ it is infinite-dimensional. It contains an arbitrarily-good approximation to any continuous function on $\mathbb{R}$, so we can use it inside highly-expressive ML algorithms. And it turns out (though this is not at all obvious) that there are efficient algorithms to solve simple ML problems on $\bar{H}$, such as finding a function in $\bar{H}$ that interpolates some given training data, or finding a function in $\bar{H}$ that passes close to some training data while remaining as smooth as possible.

We can do this in any dimension, not just $\mathbb{R}$. Here's an example of a function in the space we get from radial basis functions on $\mathbb{R}^2$:



Visualizing functions

In order to work in a vector space of functions, it helps to be able to visualize the individual vectors (the functions themselves). One good tool is a *contour plot*. Contour plots look like topographical maps:



The curves on this plot are called *contours*, *isolines*, or *level sets*. Each one is a set of the form $\{x \mid f(x) = k\}$ for some $k \in \mathbb{R}$; that is, the height of f is the same anywhere along a given contour. The color scale shows what this height is: in this case the darkest areas are lowest and the lightest areas are highest.

When $k = 0$ we get the decision boundary of our classifier. Other values of k show what the decision boundary would be if we shifted our function up or down by adding a constant. Where contours are close together, the function is changing rapidly; where they are far apart, the function is changing more gradually.

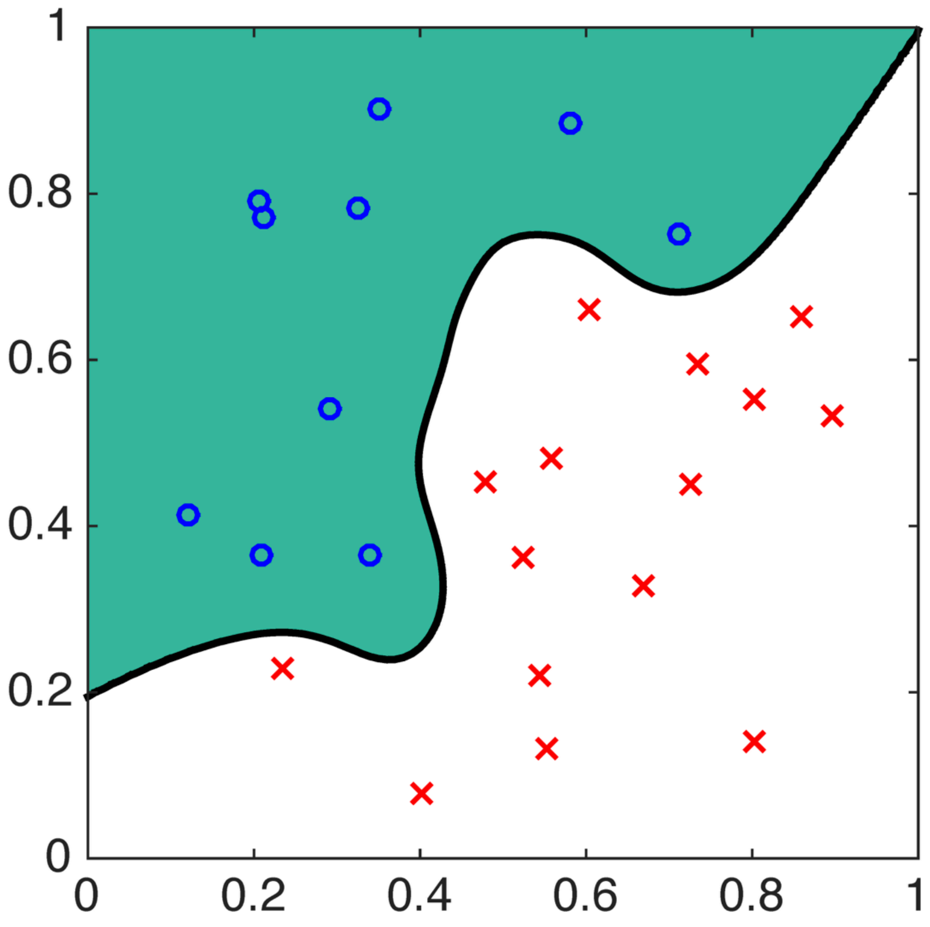
As with a linear function, the normal to a contour shows the direction of steepest increase of f (i.e., the gradient ∇f).

Linear classifiers in function space

We already saw how to learn a simple class of nonlinear decision boundaries: we use a nonlinear feature transform to map our original space to a higher-dimensional feature space, then we compute a linear classifier such as the Fisher linear discriminant in the feature space.

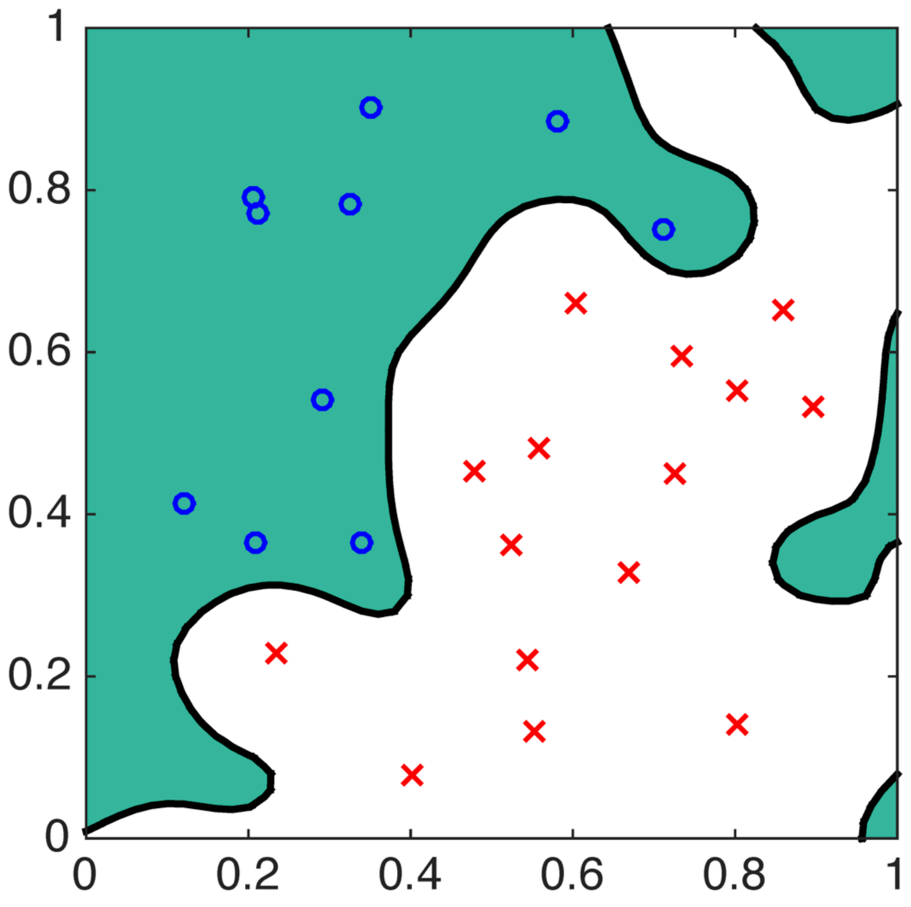
What if we want to choose our discriminant from a much larger class — say the space $\bar{H}$ from above? This seems extremely expressive: we can get arbitrarily close to any continuous decision boundary. But it also seems computationally difficult.

This is where it helps to think in function space directly. It turns out we can use the exact same recipe as before: use a feature transform to map our original data points into $\bar{H}$, and find a Fisher linear discriminant in $\bar{H}$. Here's what that looks like:



Note that the decision boundary is now much more complicated than a plain line or ellipse.

By choosing a different function space we can get a different classifier. For example, if we reduce the width of our radial basis functions, we get a wigglier decision boundary:



In more detail, to learn a Fisher linear discriminant directly in a function space, we have to pick a feature transform that maps our input vectors into the function space. In our example, we might choose

$$\phi(x) = q_x$$

so that $\phi \in \mathbb{R}^2 \to \bar{H}$. That is, we represent a point in the plane by a radial basis function centered at that point.

Given this choice, the class means use exactly the same formula as before:

$$\mu_\times = \frac{1}{n_\times} \sum_{i=1}^{n_\times} \phi(x_i^\times)$$

$$\mu_\circ = \frac{1}{n_\circ} \sum_{j=1}^{n_\circ} \phi(x_j^\circ)$$

Unlike before, we can't simplify the sums above, unless the same exact input vector x appears twice: we just have to represent the sums as a list of terms.

The formulas for w and b are also the same as before. So, the weight vector w is a linear function of $\mu_\times$ and $\mu_\circ$. Again we can't simplify w, and have to just represent it as a list of terms.

Now when we get a new point x to classify, we have to compute $\langle w, \phi(x)\rangle$. We can do this by linearity: we compute the inner product of $\phi(x)$ with each term in the representation of w. Since each term in w is based on a single training point, the key step is to compute $\langle \phi(x_i^\times), \phi(x)\rangle$ or $\langle \phi(x_j^\circ), \phi(x)\rangle$. To do this, we can use our definition of inner product from above: for example,

$$\langle \phi(x_i^\times), \phi(x)\rangle = q_x(x_i^\times)$$

If we evaluate our discriminant at every point in a grid in $\mathbb{R}^2$, we can make a plot like the ones above: these show the zero contour of the discriminant function, which is our decision boundary.