More Classification

- In **nearest-neighbor** classification, each data point is simply classified as having the same label as the training point that it is nearest to. The definition of distance is important because it depends on the scale of the features; one option is Euclidean distance, and sometimes it helps to scale the distance (for example using z-scores). A less noisy modification of nearest-neighbor is to classify each point as having the same label as the majority of its k nearest neighbors.

- Recall that we can define a hyperplane as $\mathbf{w} \cdot \mathbf{x} + b = 0$.
  Then we can define a linear classifier $h(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$.
  The **margin** of a data point $\mathbf{x}_i$ with label y_i is $\gamma_i = y_i(\mathbf{w} \cdot \mathbf{x}_i + b)$. The margin is positive for points on the correct side of the separator and negative for points on the incorrect side.
  For simplicity, we will often get rid of b by adding an extra dimension to all the $\mathbf{x}_i$'s and $\mathbf{w}$ (for the remainder of this handout we will do this).

- The **perceptron** algorithm learns $\mathbf{w}$ by iteratively updating it based on misclassified points. For each point $\mathbf{x}_i$, if $\gamma_i < 0$, the weights $\mathbf{w}$ are updated based on gradient descent, and the new weights vector becomes $\mathbf{w} + \eta y_i \mathbf{x}_i$, where η is called the learning rate. If a linear separator exists, repeating this process will eventually find one.

- We can write the weights as a linear combination of the training data points:
  $\mathbf{w} = \eta \sum_i \alpha_i y_i \mathbf{x}_i$, where α_i is the number of times that $\mathbf{x}_i$ was misclassified.
  This is called the **dual form** of the perceptron.

- A **support vector machine (SVM)** finds the maximum-margin linear separator, the unique linear separator that maximizes the distance to the closest points. If we rescale everything by $\|\mathbf{w}\|$, then the **geometric margin** $\gamma_i/\|\mathbf{w}\|$ will be 1 for the points closest to the margin. To maximize the margin, we need to minimize $\|\mathbf{w}\|$, or equivalently, minimize $\frac{1}{2}\|\mathbf{w}\|^2$, subject to the constraints that $y_i(\mathbf{w} \cdot \mathbf{x}_i) \geq 1$ (the distance from any point $\mathbf{x}_i$ to the separator has to be at least 1).

- To express an SVM in the dual form, we use Lagrange multipliers: we want to find the α_i's that maximize
  $L(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_k \alpha_i \alpha_k y_i y_k (\mathbf{x}_i \cdot \mathbf{x}_k)$,
  subject to the constraints that $\sum_i \alpha_i y_i = 0$ and $\alpha_i \geq 0$.
  The optimal weights vector is $\mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$.
  The points $\mathbf{x}_i$ for which $\alpha_i > 0$ are called **support vectors**; they are the points closest to the separator. Note that $\mathbf{w}^*$ depends only on the support vectors, the other points don't matter at all!

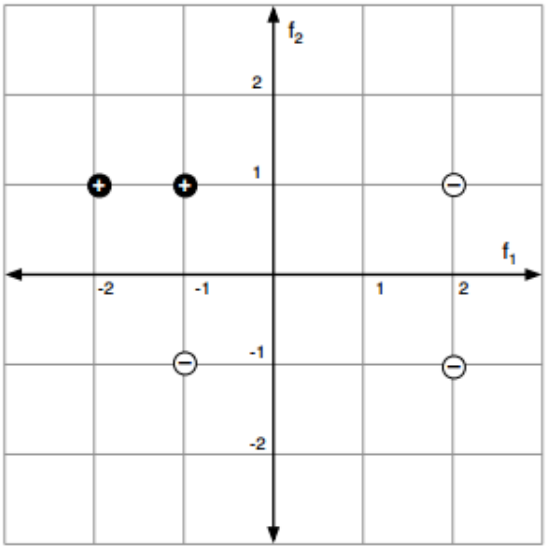
- In some cases the data are not linearly separable but we want to find a separator that does a reasonable job of separating them. A **soft-margin** SVM has an additional parameter C which is an upper bound on the α_i's. Large values of C favor fewer misclassifications, whereas small values of C favor larger margins.

- Even if the data are not linearly separable, we can define a transformation function ϕ that transforms them into a space (usually higher-dimensional) where they are linearly separable. In fact, we don't actually need to compute ϕ, because we don't need the points themselves, we only need the dot products of pairs of points. So we can define a **kernel** function $K(\mathbf{x}_i, \mathbf{x}_k) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_k)$ that gives us the dot product in the transformed space. Some typical kernels are
  polynomial kernels: $K(\mathbf{x}_i, \mathbf{x}_k) = (1 + \mathbf{x}_i \cdot \mathbf{x}_k)^n$
  and radial basis (Gaussian) kernels: $K(\mathbf{x}_i, \mathbf{x}_k) = \exp\left(\frac{-1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_k\|^2\right)$.

Exercises

1. (AIMA 18.17) Construct a support vector machine that computes the `XOR` function. Use values of +1 and −1 (instead of 1 and 0) for both inputs and outputs, so that an example looks like ($[-1, 1]$, 1) or ($[-1, -1]$, −1). Map the input $[x_1, x_2]$ into a space consisting of x_1 and $x_1 x_2$. Draw the four input points in this space, and the maximal margin separator. What is the margin? Now draw the separating line back in the original Euclidean space.

2. Suppose that you want to classify the following data with a perceptron:



Recall that the perceptron algorithm uses the extended form of the data points in which a 1 is added as the 0th component.

(a) Assume that the initial value of the weight vector for the perceptron is [0, 0, 1]; that the data points are examined in the order (-1, -1), (2, 1), (2, -1), (-2, 1), (-1, 1); and that the learning rate is 1.0. Give the weight vector after one iteration of the algorithm (one pass through all the data points):

(b) Draw the separator corresponding to the weights after this iteration on the graph.

(c) Would the algorithm stop after this iteration or keep going? Explain.

(d) If we add a new positive point at (1, -1) to the other points and retrain the perceptron, what would the perceptron algorithm do? Explain.