Support vector machines (for classification)

Recall: perceptron

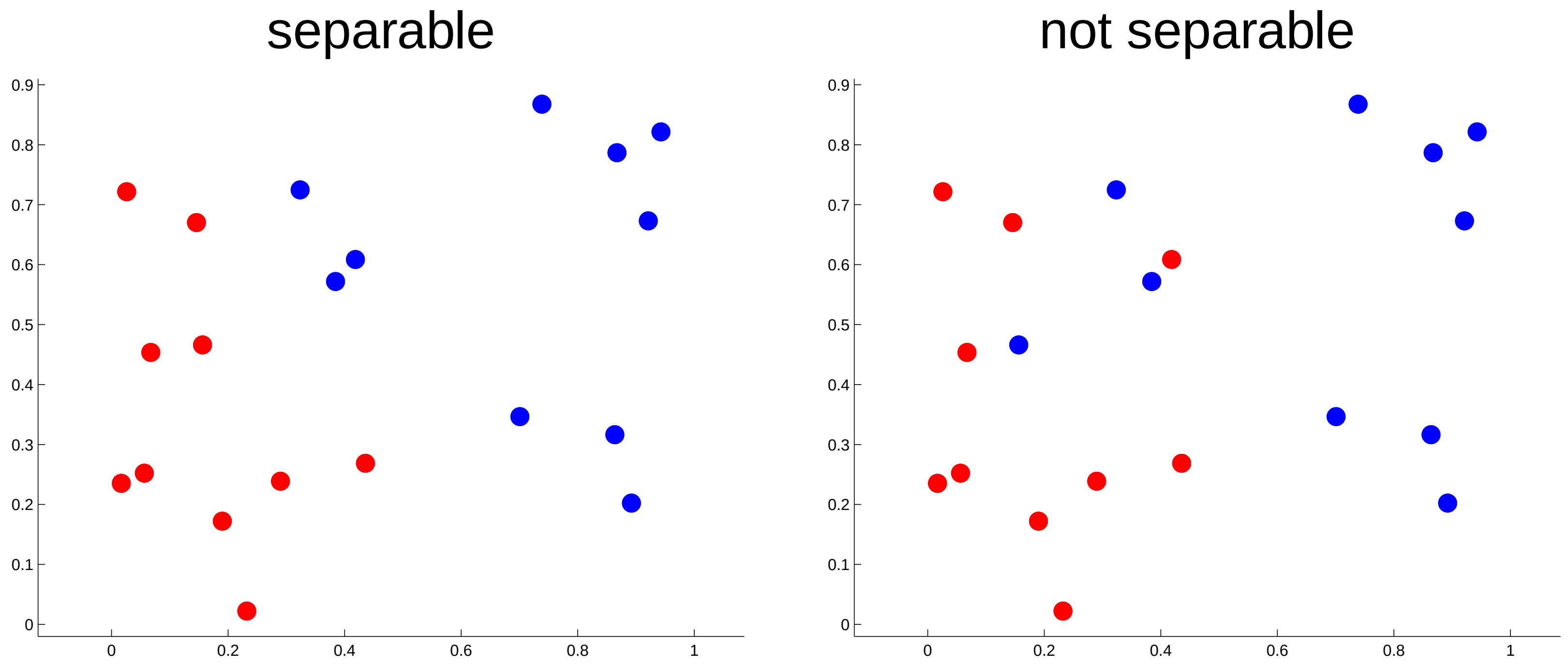
- Data is $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{m}$ where $\mathbf{x}_i \in \Re^n$, $\mathbf{y}_i \in \{-1, +1\}$.
- Classification rule for input $\mathbf{x}$ is

$$\hat{f}(\mathbf{x}) = \text{sgn}(\mathbf{x} \cdot \mathbf{w} + w_0) = \begin{cases} +1 & \text{if } \mathbf{x} \cdot \mathbf{w} + w_0 > 0 \\ ? & \text{if } \mathbf{x} \cdot \mathbf{w} + w_0 = 0 \\ -1 & \text{if } \mathbf{x} \cdot \mathbf{w} + w_0 < 0 \end{cases}$$

  for weights $\mathbf{w}$, w_0.
- The decision boundary, separating the regions of +1 prediction and -1 prediction, is the hyperplane $\mathbf{x} \cdot \mathbf{w} + w_0 = 0$.
- An example $(\mathbf{x}_i, \mathbf{y}_i)$ is correctly classified if $\mathbf{y}_i(\mathbf{x}_i \cdot \mathbf{w} + w_0) > 0$.

Recall: linear separability

- The data is linearly separable if there exists $\mathbf{w}, w_0$ such that all examples are classified correctly.



Recall: perceptron training

- We saw a gradient-descent based rule for choosing the perceptron weights. If the data is linearly separable, then it finds weights that correctly classify all the data.
- Weights can also be found by minimizing the perceptron criterion,

$$E = \sum_{\substack{\text{misclassified} \\ \text{examples} \\ (\mathbf{x}_i, \mathbf{y}_i)}} -\mathbf{y}_i(\mathbf{x}_i \cdot \mathbf{w} + w_0)$$

  using (e.g.) linear programming.

Linear support vector machines

- Linear SVMs are perceptrons whose weights optimize a slightly different function than the perceptron criterion.
- First, consider the linearly separable case.
  - Typically, there is more than one linear decision boundary which correctly classifies the data.
  - The (geometric) *margin* is two times the distance from the decision boundary to the nearest training example.
    (It is the width of the “strip” around the decision boundary containing no training examples.)
  - SVMs: The best solution is the one with maximum margin!

Computing the margin

- Given $\mathbf{w}, w_0$, that classify the data correctly, what is the distance, δ_i, from the decision boundary to a point $\mathbf{x}_i$?

$$\delta_i = \mathbf{y}_i \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}_i + \frac{w_0}{\|\mathbf{w}\|} \right)$$

- So half the margin is $M = \min_i \delta_i$.

- Alternatively, if the margin is at least $2M$, then for all i

$$\mathbf{y}_i \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}_i + \frac{w_0}{\|\mathbf{w}\|} \right) \geq M$$

Finding the max-margin classifier

- Formulation as an optimization problem:

$$\begin{array}{rl} \text{maximize} & M \\ \text{with respect to} & \mathbf{w}, w_0, M \\ \text{subject to} & \mathbf{y}_i \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}_i + \frac{w_0}{\|\mathbf{w}\|} \right) \geq M \text{ for all } i \end{array}$$

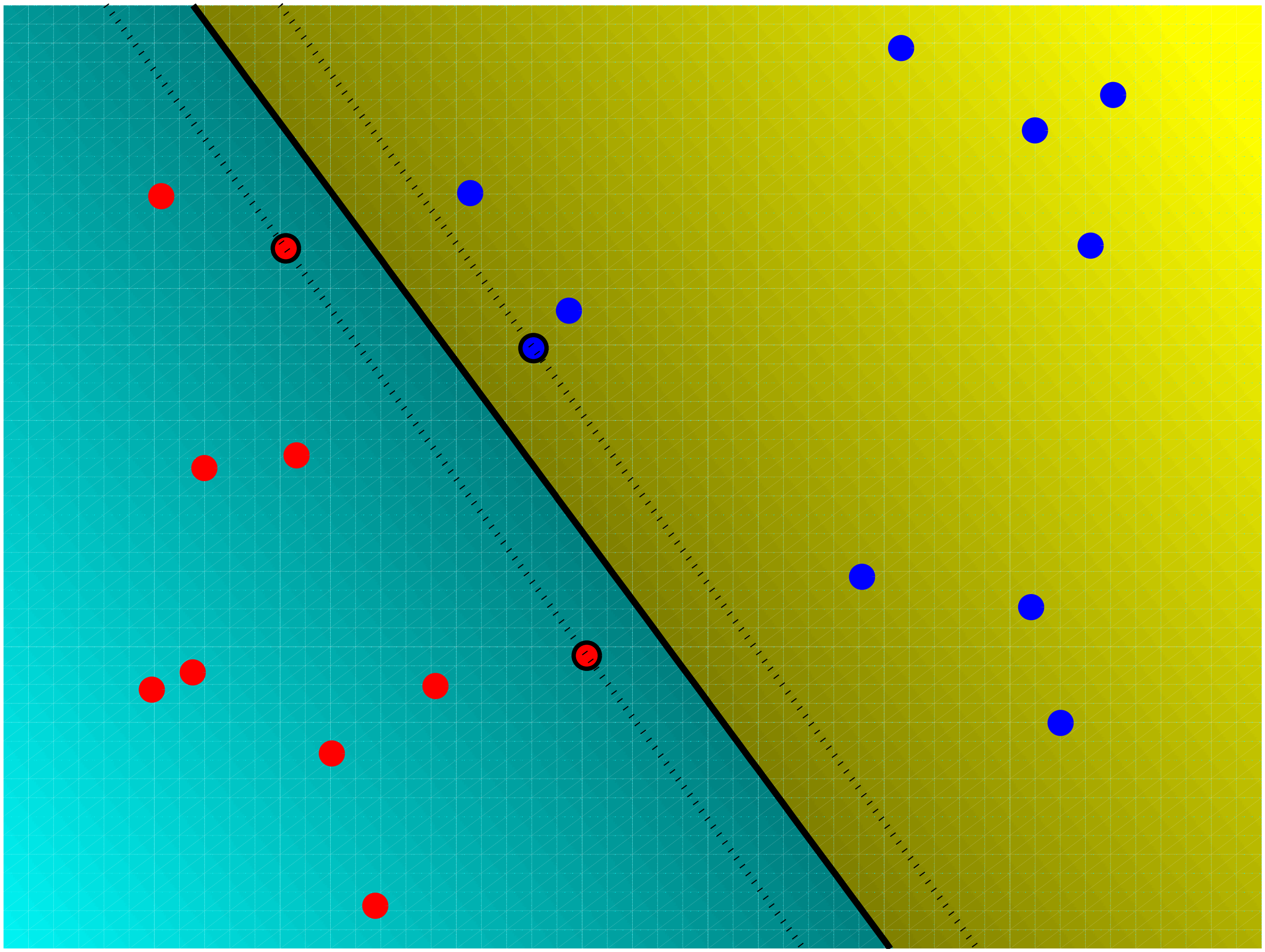
- This problem is underconstrained. If $(\mathbf{w}, w_0, M)$ is an optimal solution, then so is $(\alpha \mathbf{w}, \alpha w_0, M)$ for any $\alpha > 0$.

- Adding the constraint $\|\mathbf{w}\| M = 1$ and maximizing M^2 instead of M yields an equivalent optimization problem:

$$\begin{array}{rl} \text{min} & \|\mathbf{w}\|^2 \\ \text{w.r.t.} & \mathbf{w}, w_0 \\ \text{s.t.} & \mathbf{y}_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1 \end{array}$$

- This can be solved by standard QP software. Margin $= 2/\|\mathbf{w}\|$.

Example



Form of the solution

- It turns out that the solution for $\mathbf{w}$ is always of the form

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{y}_i \mathbf{x}_i$$

where the α_i are non-negative weighting factors. Thus, the SVM output is

$$\hat{f}(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i \mathbf{y}_i \mathbf{x}_i \cdot \mathbf{x} + w_0\right)$$

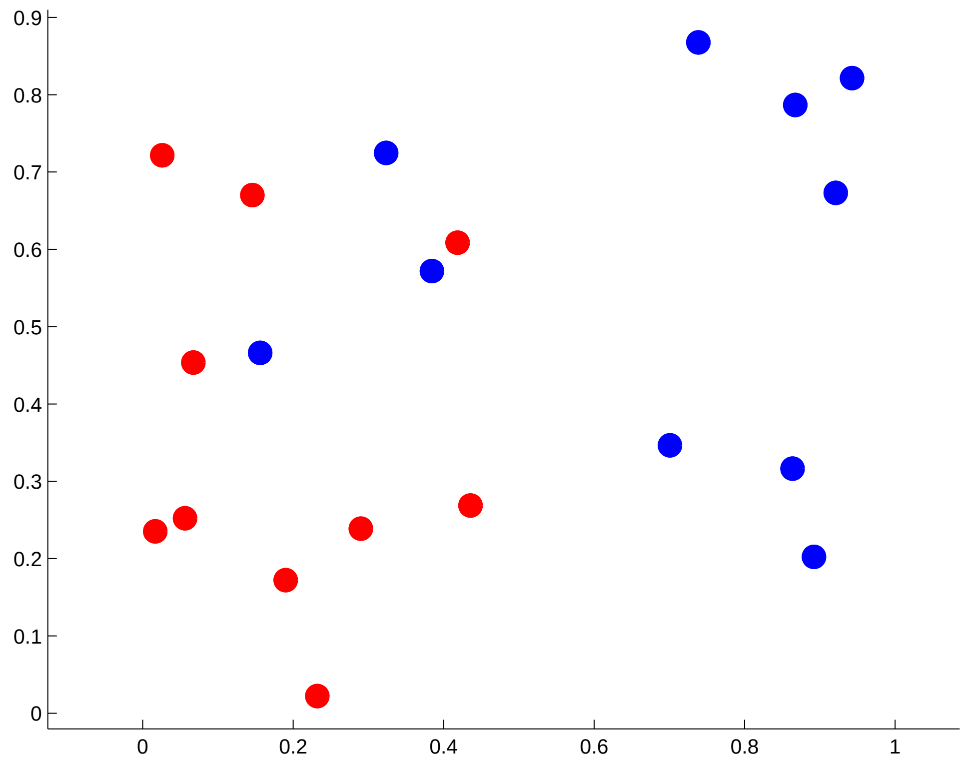
- In the separable case, α_i is non-zero if and only if $\mathbf{x}_i$ lies on the edge of the margin. That is, if and only if

$$\mathbf{y}_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) = 1$$

- Such $\mathbf{x}_i$ are called the *support vectors*. They are the training examples which determine the decision boundary.

Linear SVMs — the non-separable case

If the data is not separable, then no margin is possible (in the previously-discussed sense).



Idea: Find a decision boundary that separates some of the data well (with a large margin), and charge a penalty for the data that is not separated.

Formulating the optimization problem

- Given $\mathbf{w}, w_0, M$, an example $(\mathbf{x}_i, \mathbf{y}_i)$ exceeds the margin if

$$\mathbf{y}_i \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}_i + \frac{w_0}{\|\mathbf{w}\|} \right) \geq M$$

- If $(\mathbf{x}_i, \mathbf{y}_i)$ doesn't exceed the margin, then we can write

$$\mathbf{y}_i \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}_i + \frac{w_0}{\|\mathbf{w}\|} \right) \geq M(1 - \zeta_i)$$

where ζ_i is the distance, in fractions of M, that $(\mathbf{x}_i, \mathbf{y}_i)$ is on the wrong side of the margin.

Formulating the optimization problem

- We want M to be large, but the ζ_i's to be small (zero is best).
- This suggests the optimization problem:

$$\begin{array}{rl} \text{min} & \|\mathbf{w}\|^2 + C\sum_i \zeta_i \\ \text{w.r.t.} & \mathbf{w}, w_0, \zeta_i \\ \text{s.t.} & \mathbf{y}_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq (1-\zeta_i) \\ & \zeta_i \geq 0 \end{array}$$

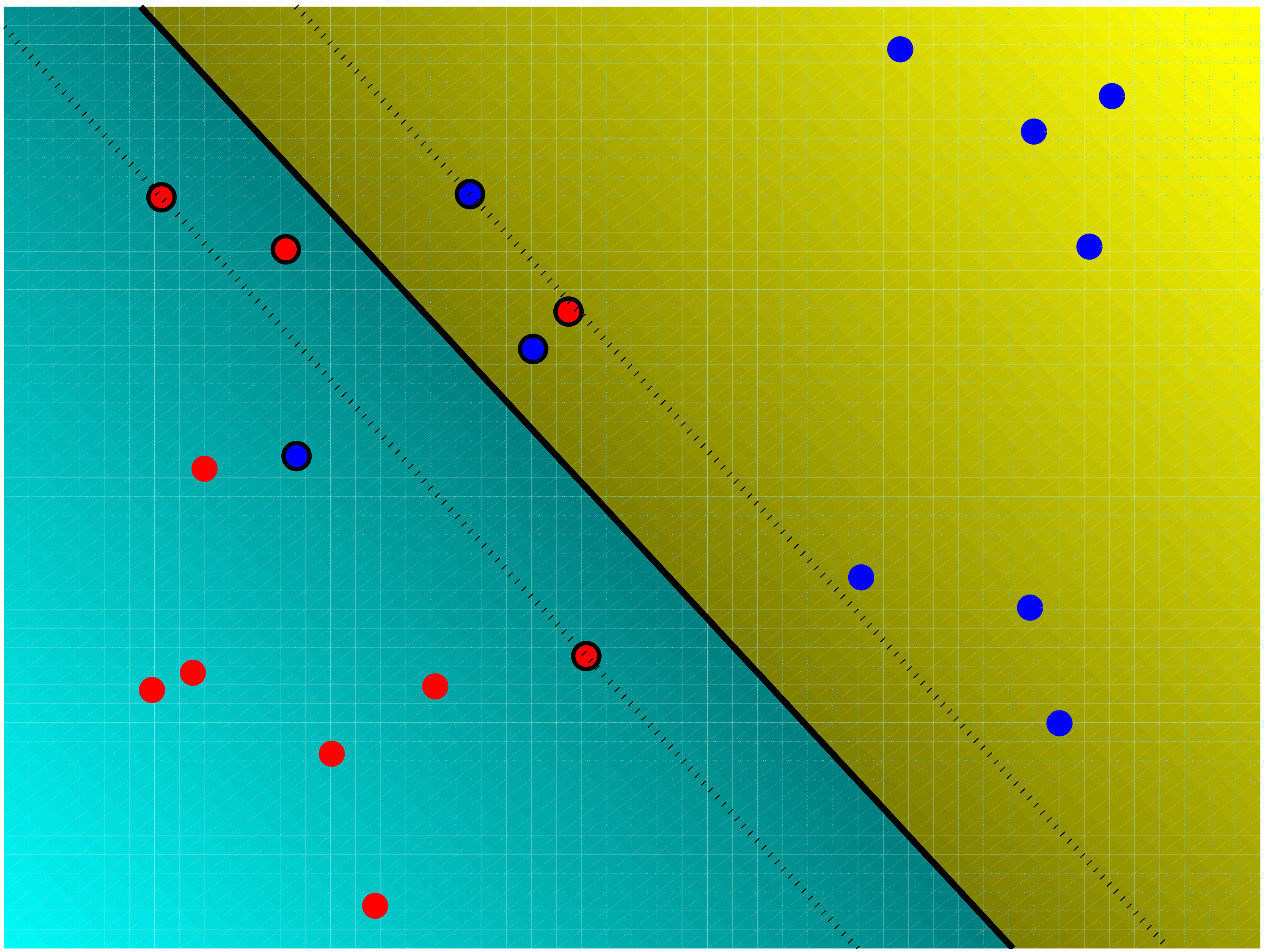
  (Recall, $M = 1/\|\mathbf{w}\|$. We still call $2M$ the margin.)

- Compare with the formulation for linearly-separable data:

$$\begin{array}{rl} \text{min} & \|\mathbf{w}\|^2 \\ \text{w.r.t.} & \mathbf{w}, w_0 \\ \text{s.t.} & \mathbf{y}_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq 1 \end{array}$$

- Either can also be solved by quadratic programming.

Example



Solution

- As in the separable case, the solution for $\mathbf{w}$ is of the form

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{y}_i \mathbf{x}_i$$

where the α_i are non-negative weighting factors. Thus,

$$\hat{f}(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i \mathbf{y}_i \mathbf{x}_i \cdot \mathbf{x} + w_0 \right)$$

- α_i is positive if and only if $\mathbf{x}_i$ lies on the edge of the margin *or* on the wrong side of the margin. That is, if $\mathbf{y}_i(\mathbf{x}_i \cdot \mathbf{w} + w_0) \leq 1$.
- Such an $\mathbf{x}_i$ is a *support vector*.

Kernels

Feature expansions and nonlinear decision boundaries

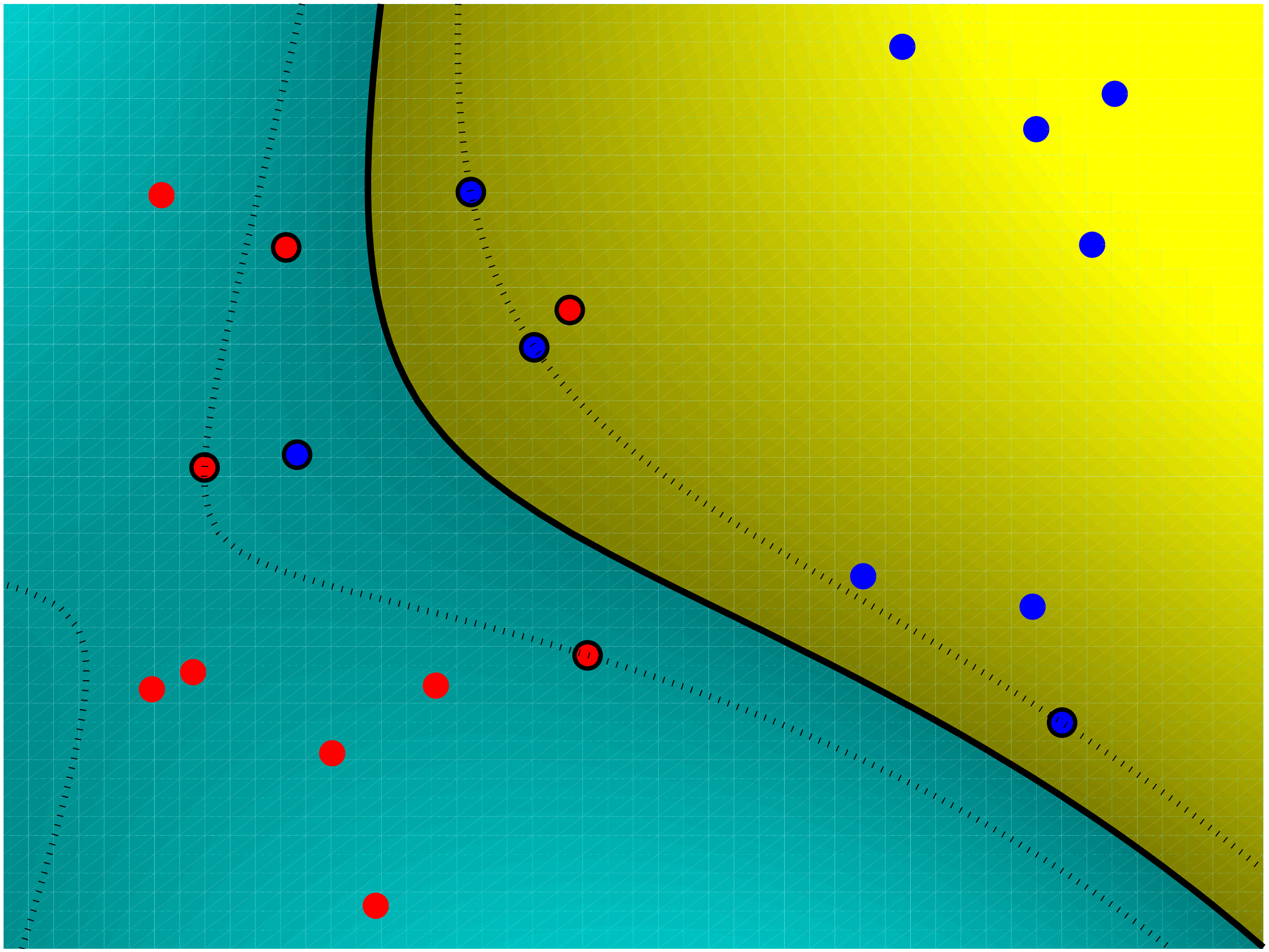
- Linear SVMs always produce a decision boundary that is a hyperplane. For some data this is not appropriate.

- One way of getting a nonlinear decision boundary in the input space is to find a linear decision boundary in an expanded space. (Similar to polynomial regression.)

- That is $\mathbf{x}_i$ is replaced by $\phi(\mathbf{x}_i)$, where $\phi : \Re^n \mapsto \Re^p$, where $p \in \{1, 2, 3, \ldots\}$ or possibly even $p = \infty$. Also, $\mathbf{w} \in \Re^p$.

- If p is finite, then the same SVM algorithm can be applied to find $\mathbf{w}$.

Example



Dot-products

- In a linear SVM, dot-products in the input space are used to make a prediction:

$$\hat{f}(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i \mathbf{y}_i \mathbf{x}_i \cdot \mathbf{x} + w_0\right)$$

- Recall, $\mathbf{x}_i \cdot \mathbf{x} = \cos z \|\mathbf{x}_i\| \|\mathbf{x}\|$, where z is the angle between the two vectors.

- This is partly a measure of "similarity" between $\mathbf{x}_i$ and $\mathbf{x}$ — specifically how much they point in the same direction — though it also reflect the lengths of the vectors.

Dot-products (2)

- The optimization problem to find $\mathbf{w}$ and w_0 can also be formulated in terms of dot-products in the input space.

$$\begin{array}{rl} \text{min} & \|\mathbf{w}\|^2 + C\sum_i \zeta_i \\ \text{w.r.t.} & \mathbf{w}, w_0, \zeta_i \\ \text{s.t.} & \mathbf{y}_i(\mathbf{w}\cdot\mathbf{x}_i + w_0) \geq (1-\zeta_i) \\ & \zeta_i \geq 0 \end{array}$$

can be solved by instead solving

$$\begin{array}{rl} \text{max} & \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m \mathbf{y}_i\mathbf{y}_j\alpha_i\alpha_j\mathbf{x}_i\cdot\mathbf{x}_j \\ \text{w.r.t.} & \alpha_i \\ \text{s.t.} & 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^m \alpha_i\mathbf{y}_i = 0 \end{array}$$

and using $\mathbf{w} = \sum_{i=1}^m \alpha_i\mathbf{y}_i\mathbf{x}_i$. w_0 can be found in several ways.

Kernels

- Whenever a learning algorithm (such as SVMs) can be written in terms of dot-products, it can be generalized to kernels.
- A *kernel* is any function $K : \Re^n \times \Re^n \mapsto \Re$ which corresponds to a dot product for some feature mapping. That is, $K(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2)$ for some $\phi : \Re^n \mapsto \Re^p$, $p \in \{1, 2, 3, \ldots, \infty\}$.
- Example Kernels:
  - Degree d polynomial: $K(\mathbf{x}_1, \mathbf{x}_2) = (1 + \mathbf{x}_1 \cdot \mathbf{x}_2)^d$
  - Radial basis/Gaussian: $K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2|^2/s)$
  - Neural network: $K(\mathbf{x}_1, \mathbf{x}_2) = \tanh(c_1\mathbf{x}_1 \cdot \mathbf{x})2 + c_2)$

Training SVMs with Kernels

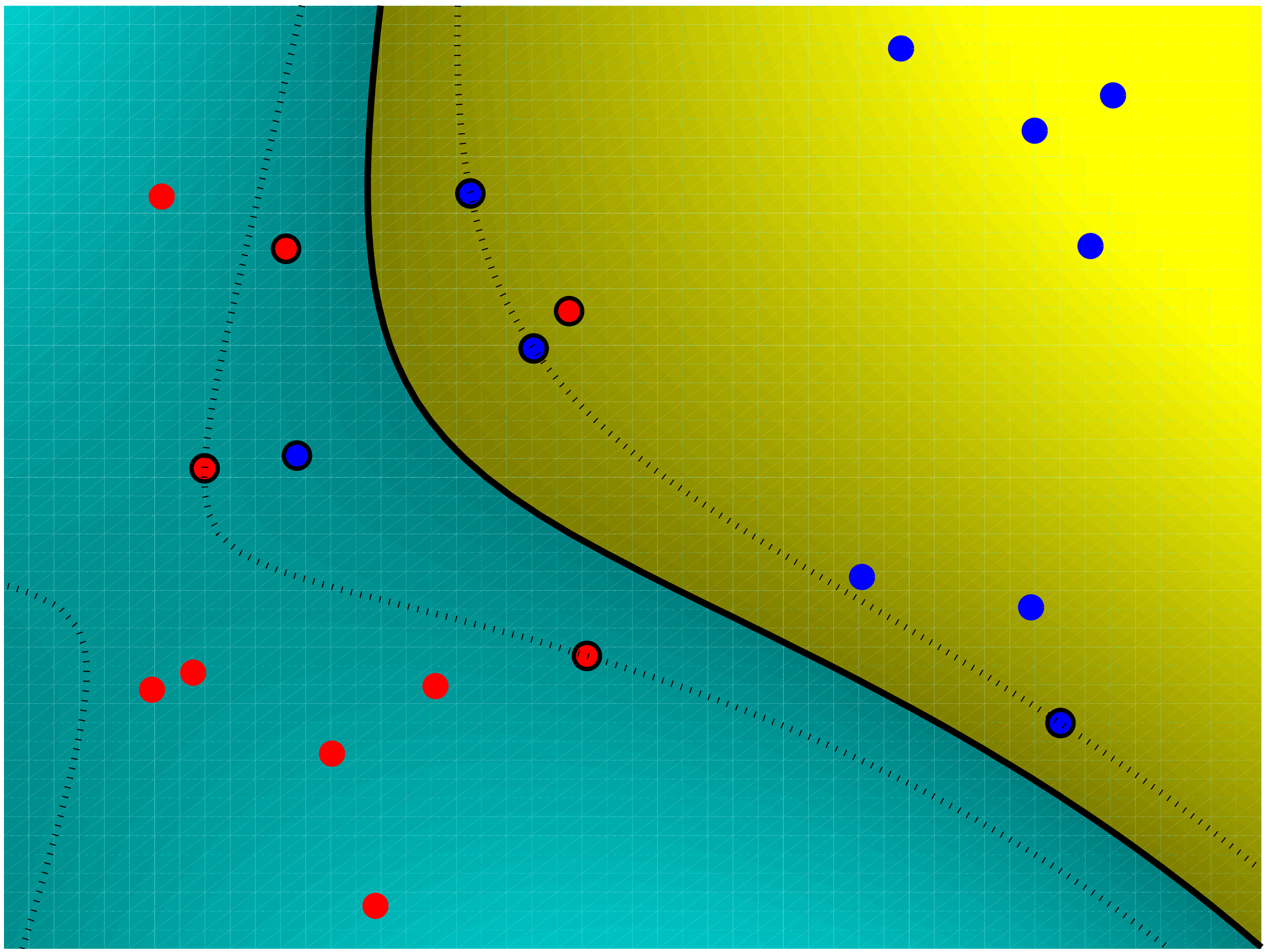
- We solve the optimization problem

$$\begin{array}{rl} \text{max} & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \mathbf{y}_i \mathbf{y}_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{w.r.t.} & \alpha_i \\ \text{s.t.} & 0 \le \alpha_i \le C \\ & \sum_{i=1}^m \alpha_i \mathbf{y}_i = 0 \end{array}$$

- We evaluate the predictor as

$$\hat{f}(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i \mathbf{y}_i K(\mathbf{x}_i, \mathbf{x}) + w_0 \right)$$

Example



SVM summary

- SVMs are perceptrons which optimize a different criterion than the usual perceptron.
  - The separable case, they maximize the margin between the $+$'s and the $-$'s.
  - In the nonseparable case, they seek a large margin but a low penalty for points on the wrong side of the margin.
- Kernels (and explicit feature expansions) allow for decision boundaries that are nonlinear in the original input features.
- Standard optimization software can be used to compute optimal parameters for the classifier.
- To evaluate the classified, one computes a weighted sum of dot-products (or Kernels) evaluated between a subset of the training points (the support vectors) and the input point.