Lecture 9: Large Margin Classifiers. Linear Support Vector Machines

- Perceptrons
  - Definition
  - Perceptron learning rule
  - Convergence
- Margin & max margin classifiers
- (Linear) support vector machines
  - Formulation as optimization problem
  - Generalized Lagrangian and dual
  - Allowing for noise (soft margins)
  - Solving the dual: SMO

Perceptrons

- Consider a binary classification problem with data $\{\mathbf{x}_i, y_i\}_{i=1}^m$, $y_i \in \{-1, +1\}$.
- A *perceptron* is a classifier of the form:

$$h_{\mathbf{w},w_0}(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + w_0) = \begin{cases} +1 & \text{if } \mathbf{w} \cdot \mathbf{x} + w_0 \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

  Here, $\mathbf{w}$ is a vector of weights, "$\cdot$" denotes the dot product, and w_0 is a constant offset.
- The decision boundary is $\mathbf{w} \cdot \mathbf{x} + w_0 = 0$.
- Perceptrons output a class, not a probability
- An example $\langle \mathbf{x}, y \rangle$ is classified correctly iff:

$$y(\mathbf{w} \cdot \mathbf{x} + w_0) > 0$$

A gradient descent-like learning rule

- Consider the following procedure:
  1. Initialize $\mathbf{w}$ and w_0 randomly
  2. While any training examples remain incorrecty classified
     (a) Loop through all misclassified examples
     (b) For misclassified example i, perform the updates:

$$\mathbf{w} \leftarrow \mathbf{w} + \gamma y_i \mathbf{x}_i, \qquad w_0 \leftarrow w_0 + \gamma y_i$$

     where γ is a step-size parameter.
- The update equation, or sometimes the whole procedure, is called the *perceptron learning rule*.
- Intuition: Yes, for examples misclassified as negative, increase $\mathbf{w} \cdot \mathbf{x}_i + w_0$, for examples misclassified as positive, it decrease it

Gradient descent interpretation

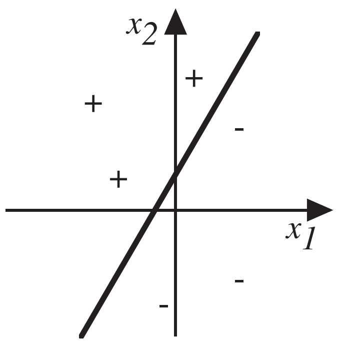
- The perceptron learning rule can be interpreted as a gradient descent procedure, but with the following *perceptron criterion function*

$$J(\mathbf{w}, w_0) = \sum_{i=1}^{m} \begin{cases} 0 & \text{if } y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 0 \\ -y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) & \text{if } y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) < 0 \end{cases}$$

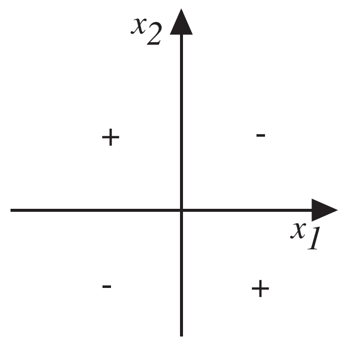
- For correctly classified examples, the error is zero.
- For incorrectly classified examples, the error is by how much $\mathbf{w} \cdot \mathbf{x}_i + w_0$ is on the wrong side of the decision boundary.
- J is piecewise linear, so it has a gradient almost everywhere; the gradient gives the perceptron learning rule.
- J is zero iff all examples are classified correctly – just like the 0-1 loss function.

Linear separability

- The data set is *linearly separable* if and only if there exists $\mathbf{w}$, w_0 such that:
  - For all i, $y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) > 0$.
  - Or equivalently, the 0-1 loss is zero.



(a)

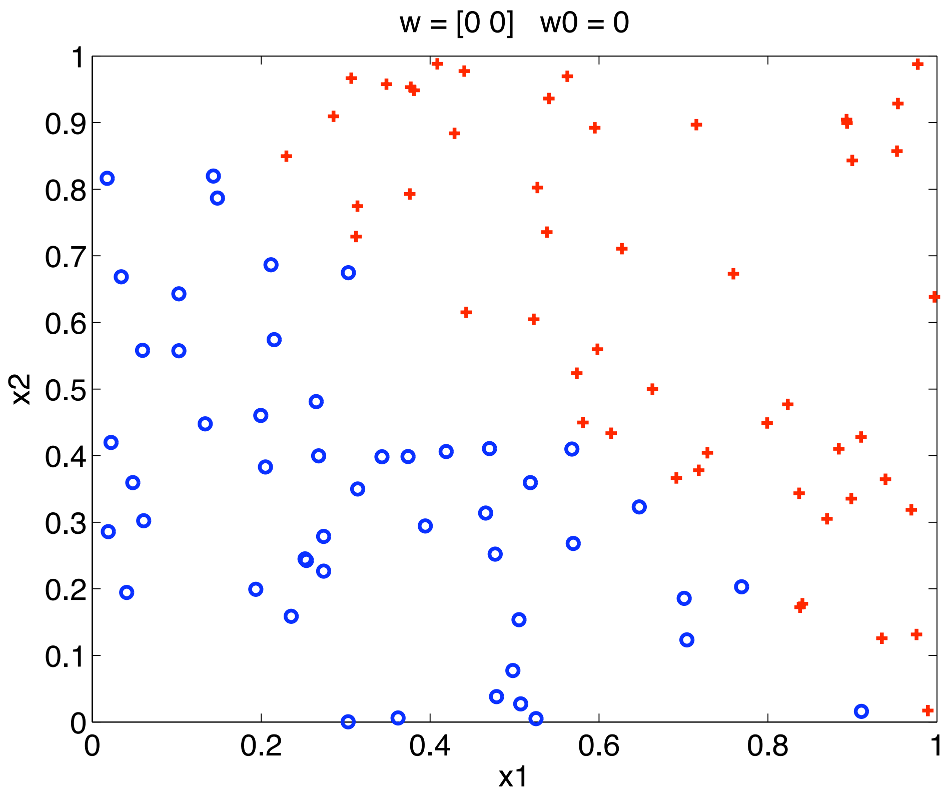


(b)

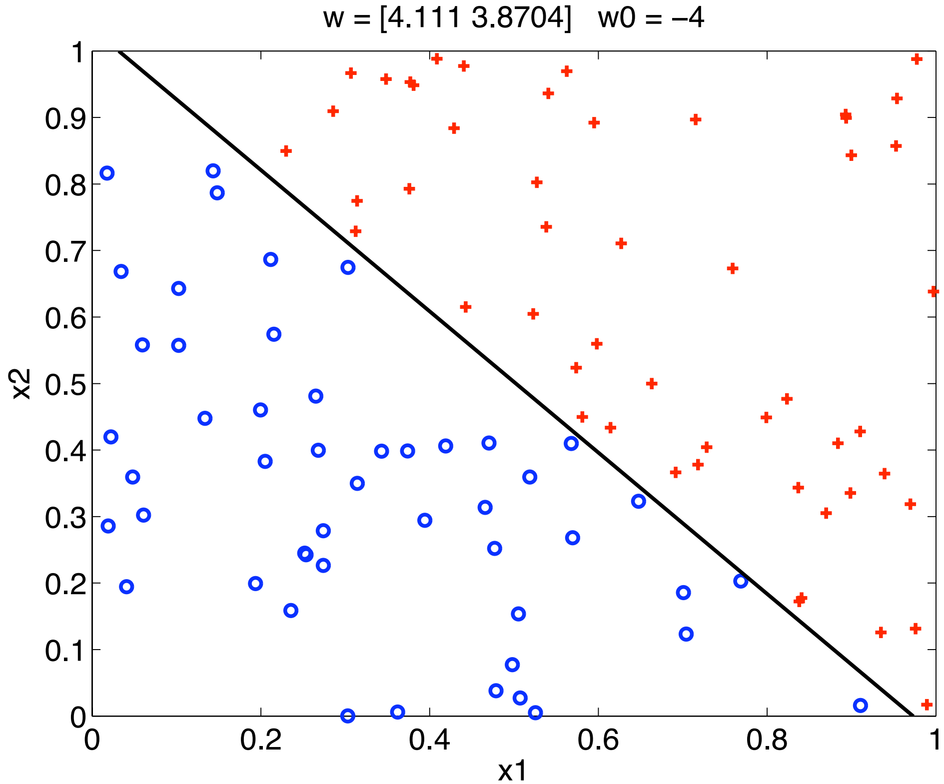
Perceptron convergence theorem

- The *perceptron convergence theorem* states that if the perceptron learning rule is applied to a linearly separable data set, a solution will be found after some finite number of updates.
- The number of updates depends on the data set, and also on the step size parameter.
- If the data is not linearly separable, there will be oscillation (which can be detected automatically).

Perceptron learning example–separable data



Perceptron learning example–separable data



Weight as a combination of input vectors

- Recall percepton learning rule:

$$\mathbf{w} \leftarrow \mathbf{w} + \gamma y_i \mathbf{x}_i, \qquad w_0 \leftarrow w_0 + \gamma y_i$$

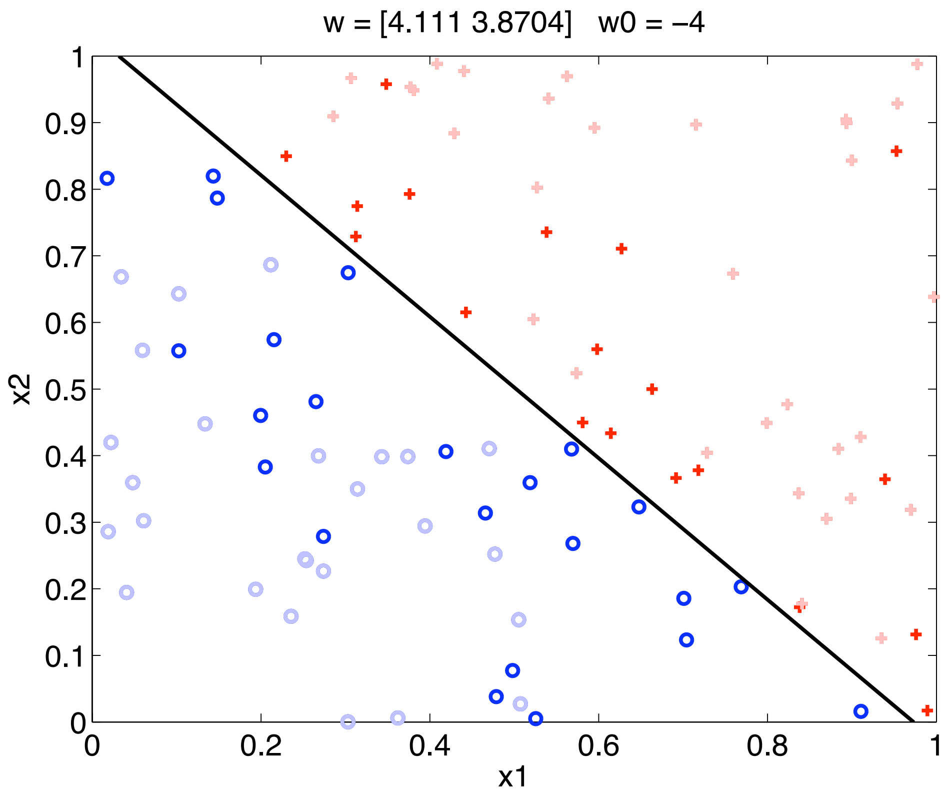
- If initial weights are zero, then at any step, the *weights are a linear combination of feature vectors*:

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i, \qquad w_0 = \sum_{i=1}^{m} \alpha_i y_i$$

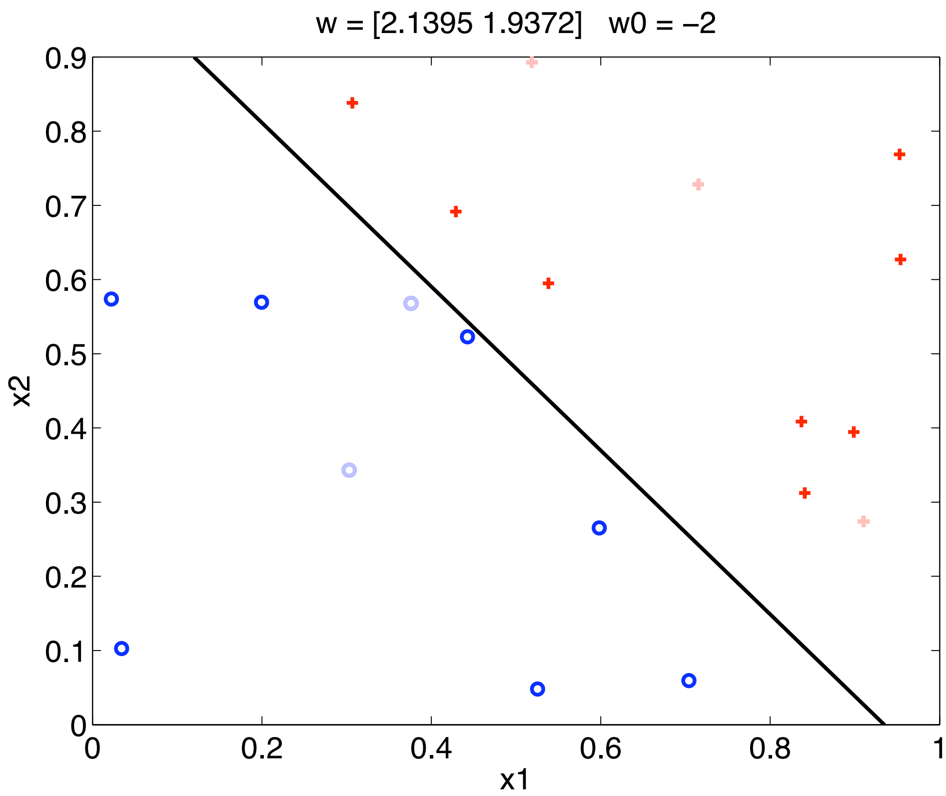
  where α_i is the sum of step sizes used for all updates based on example i.

- This is called the *dual representation* of the classifier.
- Even by the end of training, some example may have never participated in an update.

Example used (bold) and not used (faint) in updates



Comment: Solutions are nonunique



Perceptron summary

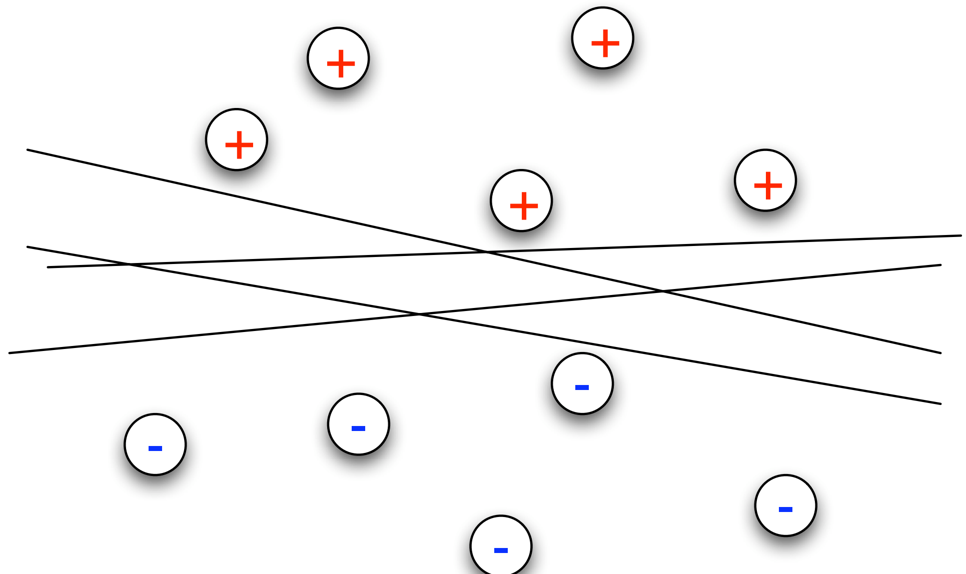
- Perceptrons can be learned to fit linearly separable data, using a gradient descent rule.
- There are other fitting approaches – e.g., formulation as a linear constraint satisfaction problem / linear program.
- Solutions are non-unique.
- Logistic neurons are often thought of as a “smooth” version of a perceptron
- For non-linearly separable data:
  - Perhaps data can be linearly separated in a different feature space?
  - Perhaps we can relax the criterion of separating all the data?

Support Vector Machines

- Support vector machines (SVMs) for binary classification can be viewed as a way of training perceptrons
- There are three main new ideas:
  - An alternative optimization criterion (the “margin”), which eliminates the non-uniqueness of solutions and has theoretical advantages
  - A way of handling nonseparable data by allowing mistakes
  - An efficient way of operating in expanded feature spaces – the “kernel trick”
- SVMs can also be used for multiclass classification and regression.

Returning to the non-uniqueness issue

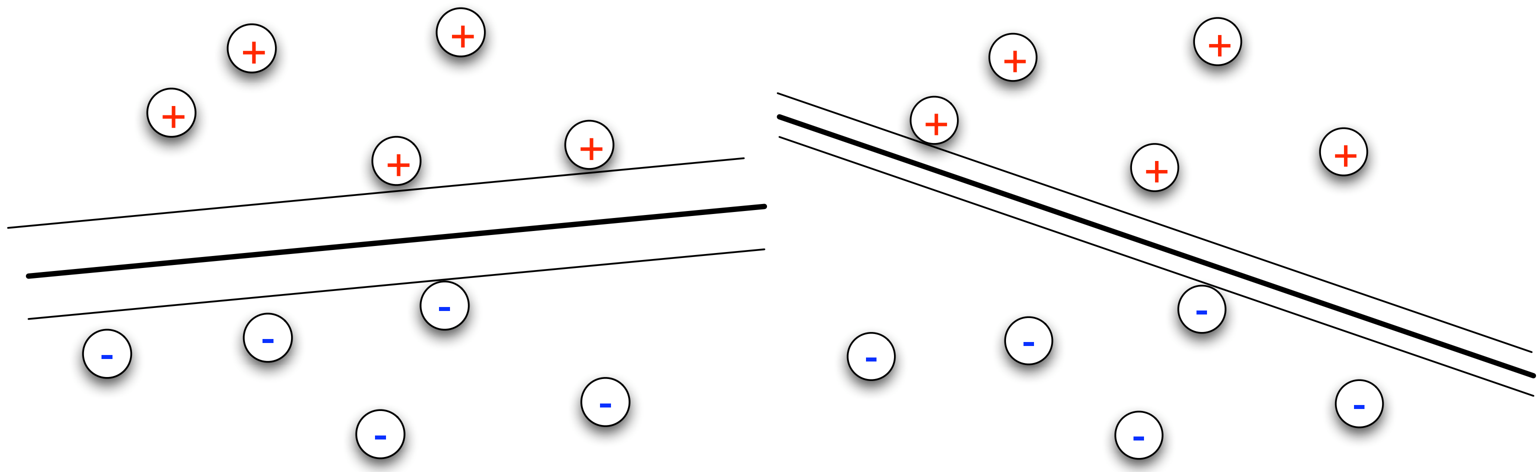
- Consider a linearly separable binary classification data set $\{\mathbf{x}_i, y_i\}_{i=1}^m$.
- There is an infinite number of hyperplanes that separate the classes:



- Which plane is best?
- Relatedly, for a given plane, for which points should we be most confident in the classification?

The margin, and linear SVMs

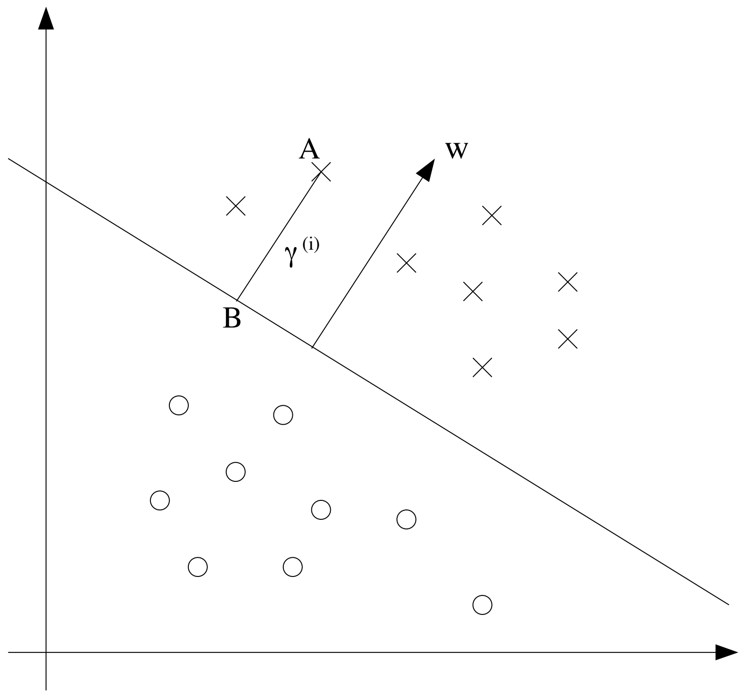
- For a given separating hyperplane, the *margin* is two times the (Euclidean) distance from the hyperplane to the nearest training example.



- It is the width of the "strip" around the decision boundary containing no training examples.
- A linear SVM is a perceptron for which we choose $\mathbf{w}, w_0$ so that margin is maximized

Distance to the decision boundary

- Suppose we have a decision boundary that separates the data.



- Let γ_i be the distance from instance $\mathbf{x}_i$ to the decision boundary.
- How can we write γ_i in term of $\mathbf{x}_i, y_i, \mathbf{w}, w_0$?

Distance to the decision boundary (II)

- The vector $\mathbf{w}$ is normal to the decision boundary. Thus, $\frac{\mathbf{w}}{||\mathbf{w}||}$ is the unit normal.
- The vector from the B to A is $\gamma_i \frac{\mathbf{w}}{||\mathbf{w}||}$.
- B, the point on the decision boundary nearest $\mathbf{x}_i$, is $\mathbf{x}_i - \gamma_i \frac{\mathbf{w}}{||\mathbf{w}||}$.
- As B is on the decision boundary,

$$\mathbf{w} \cdot \left(\mathbf{x}_i - \gamma_i \frac{\mathbf{w}}{||\mathbf{w}||} \right) + w_0 = 0$$

- Solving for γ_i yields, for a positive example:

$$\gamma_i = \frac{\mathbf{w}}{||\mathbf{w}||} \cdot \mathbf{x}_i + \frac{w_0}{||\mathbf{w}||}$$

The margin

- The *margin of the hyperplane* is $2M$, where $M = \min_i \gamma_i$
- The most direct statement of the problem of finding a maximum margin separating hyperplane is thus

$$\begin{aligned} & \max_{\mathbf{w},w_0} \min_i \gamma_i \\ \equiv \quad & \max_{\mathbf{w},w_0} \min_i y_i \left(\frac{\mathbf{w}}{||\mathbf{w}||} \cdot \mathbf{x}_i + \frac{w_0}{||\mathbf{w}||} \right) \end{aligned}$$

- This turns out to be inconvenient for optimization, however...

Treating the γ_i as constraints

- From the definition of margin, we have:

$$M \leq \gamma_i = y_i \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x_i} + \frac{w_0}{\|\mathbf{w}\|} \right) \quad \forall i$$

- This suggests:

maximize	M
with respect to	$\mathbf{w}, w_0$
subject to	$y_i \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}_i + \frac{w_0}{\|\mathbf{w}\|} \right) \geq M$ for all i

Treating the γ_i as constraints

- From the definition of margin, we have:

$$M \leq \gamma_i = y_i \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x_i} + \frac{w_0}{\|\mathbf{w}\|} \right) \quad \forall i$$

- This suggests:

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

- Problems:
  - $\mathbf{w}$ appears nonlinearly in the constraints.
  - This problem is underconstrained. If $(\mathbf{w}, w_0, M)$ is an optimal solution, then so is $(\beta \mathbf{w}, \beta w_0, M)$ for any $\beta > 0$.

Adding a constraint

- Let's try adding the constraint that $\|\mathbf{w}\|M = 1$.
- This allows us to rewrite the objective function and constraints as:

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

- This is really nice because the constraints are linear.
- The objective function $\|\mathbf{w}\|$ is still a bit awkward.

Final formulation

- Let's maximize $\|\mathbf{w}\|^2$ instead of $\|\mathbf{w}\|$.
  (Taking the square is a monotone transformation, as $\|\mathbf{w}\|$ is postive, so this doesn't change the optimal solution.)
- This gets us to:

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

- This we can solve! How?

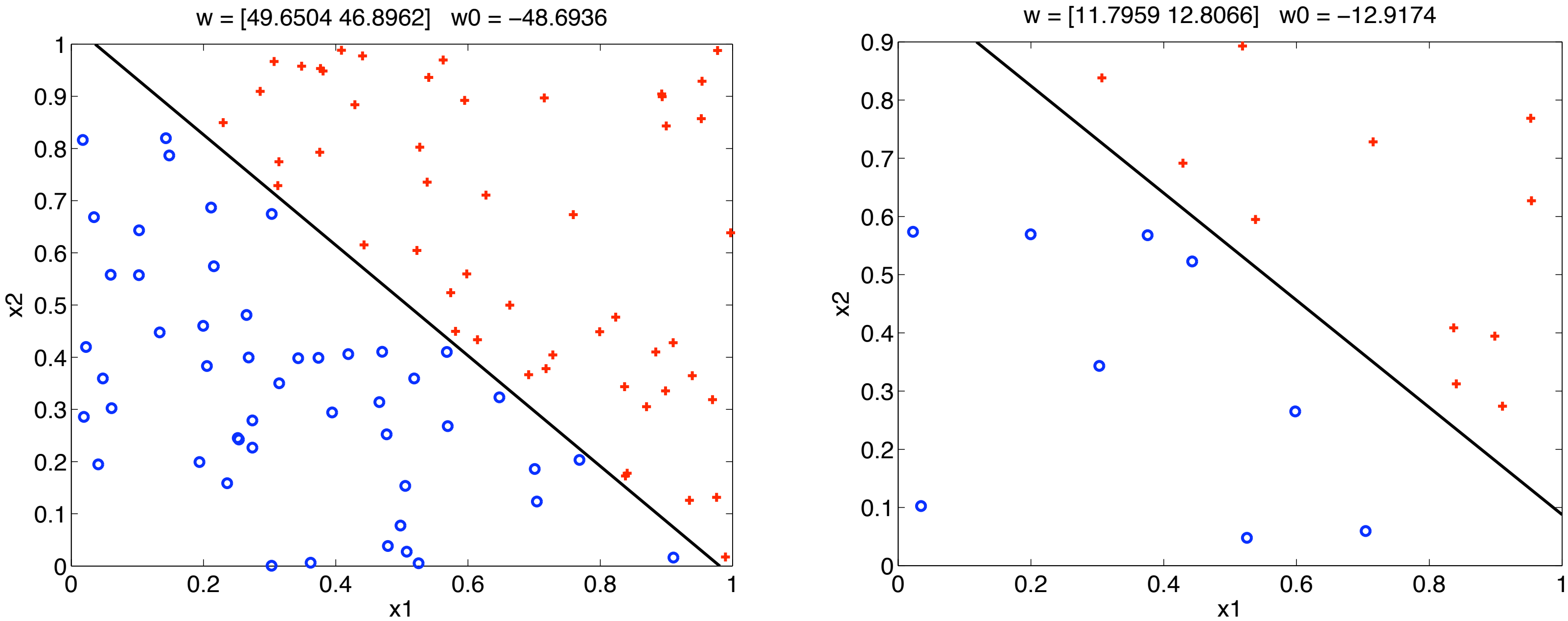
Final formulation

- Let's maximize $\|\mathbf{w}\|^2$ instead of $\|\mathbf{w}\|$.
  (Taking the square is a monotone transformation, as $\|\mathbf{w}\|$ is postive, so this doesn't change the optimal solution.)
- This gets us to:

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

- This we can solve! How?
  - It is a *quadratic programming* (QP) problem—a standard type of optimization problem for which many efficient packages are available.
  - Better yet, it's a convex (positive semidefinite) QP

Example



We have a solution, but no support vectors yet...

Lagrange multipliers for inequality constraints (revisited)

- Suppose we have the following optimization problem, called *primal*:

$$\begin{aligned}\min_{\mathbf{w}} f(\mathbf{w})& \\ \text{such that } g_i(\mathbf{w}) &\leq 0,\ i = 1 \dots k\end{aligned}$$

- We define the *generalized Lagrangian*:

$$L(\mathbf{w}, \alpha) = f(\mathbf{w}) + \sum_{i=1}^{k} \alpha_i g_i(\mathbf{w}), \tag{1}$$

where $\alpha_i,\ i = 1 \dots k$ are the Lagrange multipliers.

A different optimization problem

- Consider $\mathcal{P}(\mathbf{w}) = \max_{\alpha:\alpha_i \geq 0} L(\mathbf{w}, \alpha)$
- Observe that the follow is true. Why?

$$\mathcal{P}(\mathbf{w}) = \begin{cases} f(\mathbf{w}) & \text{if all constraints are satisfied} \\ +\infty & \text{otherwise} \end{cases}$$

- Hence, instead of computing $\min_{\mathbf{w}} f(\mathbf{w})$ subject to the original constraints, we can compute:

$$p^* = \min_{\mathbf{w}} \mathcal{P}(\mathbf{w}) = \min_{\mathbf{w}} \max_{\alpha:\alpha_i \geq 0} L(\mathbf{w}, \alpha)$$

Dual optimization problem

- Let $d^* = \max_{\alpha:\alpha_i \geq 0} \min_{\mathbf{w}} L(\mathbf{w}, \alpha)$ (max and min are reversed)
- We can show that $d^* \leq p^*$.
  - Let $p^* = L(w^p, \alpha^p)$
  - Let $d^* = L(w^d, \alpha^d)$
  - Then $d^* = L(w^d, \alpha^d) \leq L(w^p, \alpha^d) \leq L(w^p, \alpha^p) = p^*$.)

Dual optimization problem

- If f, g_i are convex and the g_i can all be satisfied simultaneously for some $\mathbf{w}$, then we have equality: $d^* = p^* = L(\mathbf{w}^*, \alpha^*)$
- Moreover $\mathbf{w}^*, \alpha^*$ solve the primal and dual if and only if they satisfy the following conditions (called Karush-Kuhn-Tucker):

$$\begin{aligned} \frac{\partial}{\partial w_i} L(\mathbf{w}^*, \alpha^*) &= 0,\ i = 1 \dots n && (2) \\ \alpha_i^* g_i(\mathbf{w}^*) &= 0,\ i = 1 \dots k && (3) \\ g_i(\mathbf{w}^*) &\leq 0,\ i = 1 \dots k && (4) \\ \alpha_i^* &\geq 0,\ i = 1 \dots k && (5) \end{aligned}$$

Back to maximum margin perceptron

- We wanted to solve (rewritten slightly):

  min $\frac{1}{2}\|\mathbf{w}\|^2$

  w.r.t. $\mathbf{w}, w_0$

  s.t. $1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) \leq 0$

- The Lagrangian is:

$$L(\mathbf{w}, w_0, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_i \alpha_i(1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0))$$

- The primal problem is: $\min_{\mathbf{w}, w_0} \max_{\alpha:\alpha_i \geq 0} L(\mathbf{w}, w_0, \alpha)$
- We will solve the dual problem: $\max_{\alpha:\alpha_i \geq 0} \min_{\mathbf{w}, w_0} L(\mathbf{w}, w_0, \alpha)$
- In this case, the optimal solutions coincide, because we have a quadratic objective and linear constraints (both of which are convex).

Solving the dual

- From KKT (2), the derivatives of $L(\mathbf{w}, w_0, \alpha)$ wrt $\mathbf{w}, w_0$ should be 0
- The condition on the derivative wrt w_0 gives $\sum_i \alpha_i y_i = 0$
- The condition on the derivative wrt $\mathbf{w}$ gives:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x_i}$$

⇒ Just like for the perceptron with zero initial weights, the optimal solution for $\mathbf{w}$ is a linear combination of the $\mathbf{x}_i$, and likewise for w_0.

- The output is

$$h_{\mathbf{w},w_0}(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + w_0\right)$$

⇒ Output depends on weighted dot product of input vector with training examples

Solving the dual (II)

- By plugging these back into the expression for L, we get:

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\mathbf{x_i} \cdot \mathbf{x_j})$$

with constraints: $\alpha_i \geq 0$ and $\sum_i \alpha_i y_i = 0$

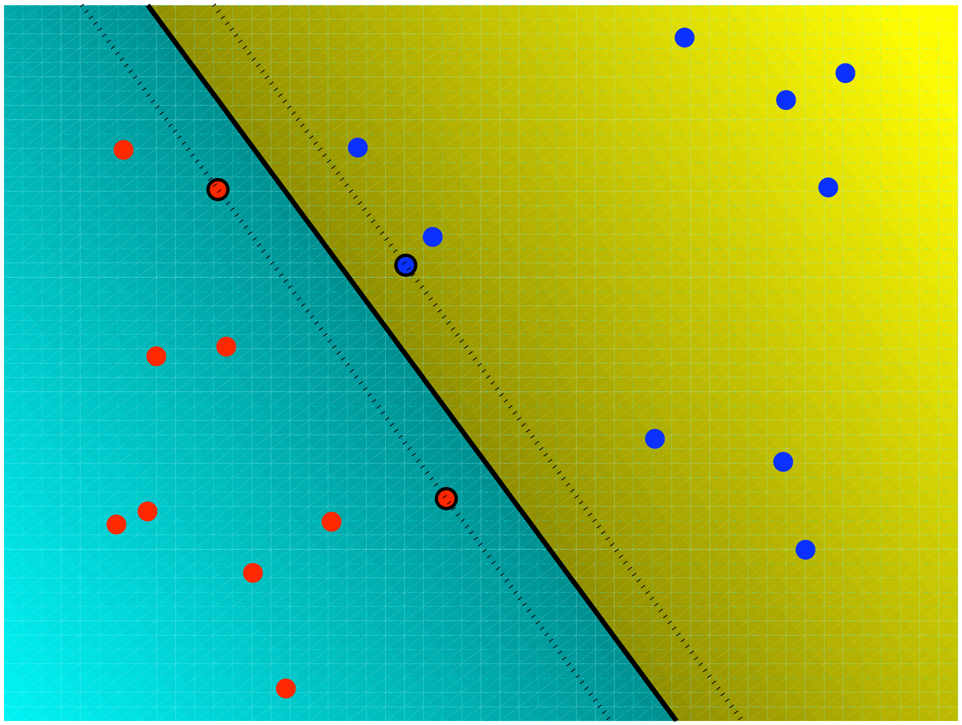
The support vectors

- Suppose we find optimal αs (e.g., using a standard QP package)
- The α_i will be > 0 only for the points for which $1 - y_i(\mathbf{w} \cdot \mathbf{x_i} + w_0) = 0$
- These are the points lying on the edge of the margin, and they are called *support vectors*, because they define the decision boundary
- The output of the classifier for query point $\mathbf{x}$ is computed as:

$$\text{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i (\mathbf{x_i} \cdot \mathbf{x}) + w_0\right)$$

Hence, the output is determined by computing the *dot product of the point with the support vectors*!

Example



Support vectors are in bold

Soft margin classifiers

- Recall that in the linearly separable case, we compute the solution to the following optimization problem:

min	$\frac{1}{2}\|\mathbf{w}\|^2$
w.r.t.	$\mathbf{w}, w_0$
s.t.	$y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1$

- If we want to allow misclassifications, we can relax the constraints to:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + w_0) \geq 1 - \xi_i$$

- If $\xi_i \in (0, 1)$, the data point is within the margin
- If $\xi_i \geq 1$, then the data point is misclassified
- We define the *soft error* as $\sum_i \xi_i$
- We will have to change the criterion to reflect the soft errors

New problem formulation with soft errors

- Instead of:

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

  we want to solve:

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

- Note that soft errors include points that are misclassified, as well as points within the margin
- There is a linear penalty for both categories
- The choice of the *constant C controls overfitting*

A built-in overfitting knob

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

- If C is 0, there is no penalty for soft errors, so the focus is on maximizing the margin, even if this means more mistakes
- If C is very large, the emphasis on the soft errors will cause decreasing the margin, if this helps to classify more examples correctly.

Lagrangian for the new problem

- Like before, we can write a Lagrangian for the problem and then use the dual formulation to find the optimal parameters:

$$
\begin{aligned}
L(\mathbf{w}, w_0, \alpha, \xi, \mu) &= \frac{1}{2}||w||^2 + C\sum_i \xi_i \\
&+ \sum_i \alpha_i \left(1 - \xi_i - y_i(\mathbf{w}_i \cdot \mathbf{x}_i + w_0)\right) + \sum_i \mu_i \xi_i
\end{aligned}
$$

- All the previously described machinery can be used to solve this problem
- Note that in addition to α_i we have coefficients μ_i, which ensure that the errors are positive, but do not participate in the decision boundary
- Next time: an even better way of dealing with non-linearly separable data

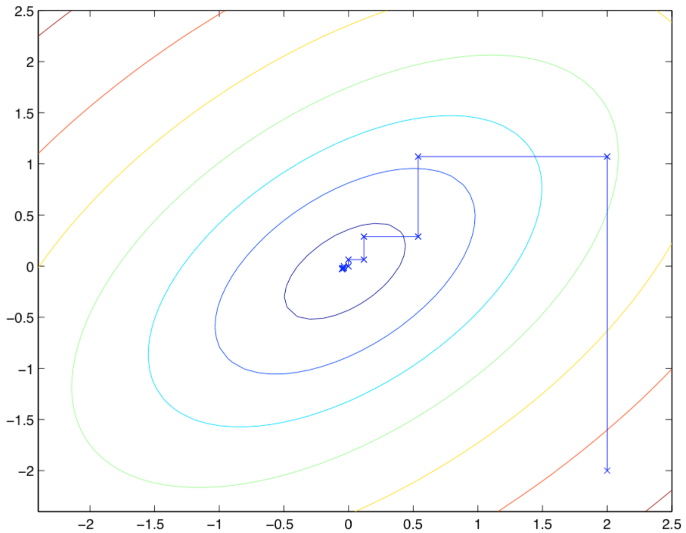
Solving the quadratic optimization problem

- Many approaches exist
- Because we have constraints, gradient descent does not apply directly (the optimum might be outside of the feasible region)
- Platt's algorithm is the fastest current approach, based on <u>*coordinate ascent*</u>

Coordinate ascent

- Suppose you want to find the maximum of some function $F(\alpha_1, \ldots \alpha_n)$
- Coordinate ascent optimizes the function by repeatedly picking an α_i and optimizing it, while all other parameters are fixed
- There are different ways of looping through the parameters:
  - Round-robin
  - Repeatedly pick a parameter at random
  - Choose next the variable expected to make the largest improvement
  - ...

Example



Our optimization problem

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j K(\mathbf{x_i}, \mathbf{x_j})$$

with constraints: $0 \leq \alpha_i \leq C$ and $\sum_i \alpha_i y_i = 0$

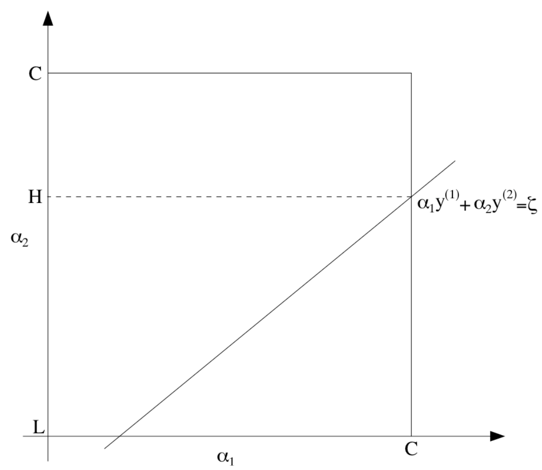
- Suppose we want to optimize for α_1 while $\alpha_2, \ldots \alpha_n$ are fixed
- We cannot do it because α_1 will be completely determined by the last constraint: $\alpha_1 = -y_1 \sum_{i=2}^{m} \alpha_i y_i$
- Instead, we have to optimize *pairs* of α_i, α_j parameters together

SMO

- Suppose that we want to optimize α_1 and α_2 together, while all other parameters are fixed.
- We know that $y_1\alpha_1 + y_2\alpha_2 = -\sum_{i=1}^{m} y_i\alpha_i = \xi$, where ξ is a constant
- So $\alpha_1 = y_1(\xi - y_2\alpha_2)$ (because y_1 is either $+1$ or -1 so $y_1^2 = 1$)
- This defines a line, and any pair α_1, α_2 which is a solution has to be on the line

SMO (II)

- We also know that $0 \leq \alpha_1 \leq C$ and $0 \leq \alpha_2 \leq C$, so the solution has to be on the line segment inside the rectangle below



SMO(III)

- By plugging α_1 back in the optimization criterion, we obtain a quadratic function of α_2, whose optimum we can find exactly
- If the optimum is inside the rectangle, we take it.
- If not, we pick the closest intersection point of the line and the rectangle
- This procedure is very fast because all these are simple computations.