A Review of Derivative-based Optimization

Nonlinear Programming

Multivariable Unconstrained Optimization

- For functions with one variable, we use the 1^{st} and 2^{nd} derivatives.
- For functions with multiple variables, we use identical information that is the gradient and the Hessian.
- The gradient is the first derivative with respect to all variables whereas the Hessian is the equivalent of the second derivative

The Gradient

- Review of the gradient (∇):

For a function "f", of variables $x_1, x_2, \ldots, x_n$:

$$\nabla f = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \cdots \quad \frac{\partial f}{\partial x_n} \right]$$

Example: $f = 15x_1 + 2(x_2)^3 - 3x_1(x_3)^2$

$$\nabla f = \left[15 - 3(x_3)^2 \quad 6(x_2)^2 \quad -6x_1x_3 \right]$$

The Hessian

- The Hessian (∇^2) of $f(x_1, x_2, \ldots, x_n)$ is:

$$\nabla^2 f = \begin{bmatrix} \frac{\partial f^2}{\partial x_1^2} & \frac{\partial f^2}{\partial x_1 \partial x_2} & \cdots & \frac{\partial f^2}{\partial x_1 \partial x_n} \\ \frac{\partial f^2}{\partial x_2 \partial x_1} & \frac{\partial f^2}{\partial x_2^2} & \cdots & \frac{\partial f^2}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f^2}{\partial x_n \partial x_1} & \frac{\partial f^2}{\partial x_n \partial x_2} & \cdots & \frac{\partial f^2}{\partial x_n^2} \end{bmatrix}$$

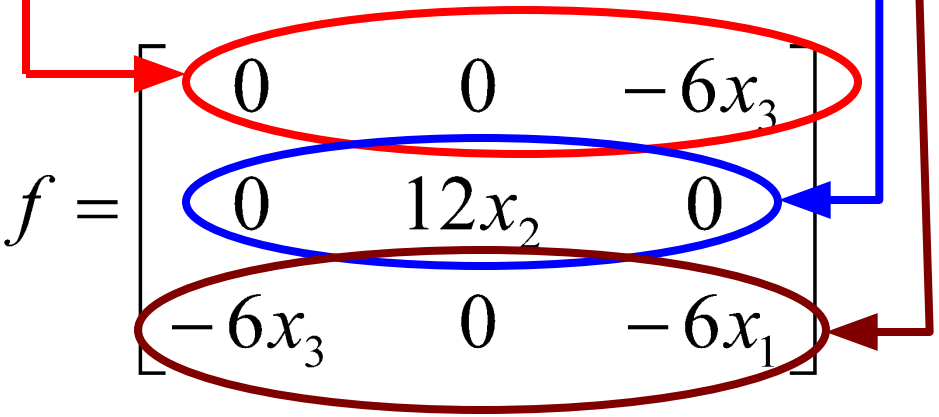
Hessian Example

- Example :

$$f = 15x_1 + 2(x_2)^3 - 3x_1(x_3)^2$$

$$\nabla f = \begin{bmatrix} 15 - 3(x_3)^2 & 6(x_2)^2 & -6x_1x_3 \end{bmatrix}$$

$$\nabla^2 f = \begin{bmatrix} 0 & 0 & -6x_3 \\ 0 & 12x_2 & 0 \\ -6x_3 & 0 & -6x_1 \end{bmatrix}$$



Unconstrained Optimization

The optimization procedure for multivariable functions is:

1. Solve for the gradient of the function equal to zero to obtain candidate points.
2. Obtain the Hessian of the function and evaluate it at each of the candidate points
   - If the result is “positive definite” then the point is a local minimum.
   - If the result is “negative definite” then the point is a local maximum.

Positive/Negative Definite

- A matrix is "positive definite" if all of the eigenvalues of the matrix are positive (> 0)

- A matrix is "negative definite" if all of the eigenvalues of the matrix are negative (< 0)

Positive/Negative Semi-definite

- A matrix is "positive semi-definite" if all of the eigenvalues are non-negative (≥ 0)

- A matrix is "negative semi-definite" if all of the eigenvalues are non-positive (≤ 0)

Example Matrix

Given the matrix A:

$$A = \begin{bmatrix} 2 & 4 & 5 \\ -5 & -7 & -1 \\ 1 & 1 & 2 \end{bmatrix}$$

The eigenvalues of A are:

$$\lambda_1 = -3.702 \qquad \lambda_2 = -2 \qquad \lambda_3 = -2.702$$

This matrix is negative definite

Unconstrained NLP Example

Consider the problem:

Minimize $f(x_1, x_2, x_3) = (x_1)^2 + x_1(1 - x_2) + (x_2)^2 - x_2x_3 + (x_3)^2 + x_3$

First, we find the gradient with respect to x_i:

$$\nabla f = \begin{bmatrix} 2x_1 + 1 - x_2 \\ -x_1 + 2x_2 - x_3 \\ -x_2 + 2x_3 + 1 \end{bmatrix}$$

Unconstrained NLP Example

Next, we set the gradient equal to zero:

$$\nabla f = 0 \qquad \Rightarrow \qquad \begin{bmatrix} 2x_1 + 1 - x_2 \\ -x_1 + 2_2 - x_3 \\ -x_2 + 2x_3 + 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

So, we have a system of 3 equations and 3 unknowns. When we solve, we get:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

Unconstrained NLP Example

So we have only one candidate point to check.

Find the Hessian:

$$\nabla^2 f = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Unconstrained NLP Example

The eigenvalues of this matrix are:

$$\lambda_1 = 3.414 \qquad \lambda_2 = 0.586 \qquad \lambda_3 = 2$$

All of the eigenvalues are > 0, so the Hessian is positive definite.

So, the point $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$ is a minimum

Method of Solution

- In the previous example, when we set the gradient equal to zero, we had a system of 3 linear equations & 3 unknowns.
- For other problems, these equations could be nonlinear.
- Thus, the problem can become trying to solve a system of nonlinear equations, which can be very difficult.

Method of Solution

- To avoid this difficulty, NLP problems are usually solved numerically.

- We will now look at examples of numerical methods used to find the optimum point for single-variable NLP problems. These and other methods may be found in any numerical methods reference.

Multivariable Optimization

- Now we will consider unconstrained multivariable optimization
- Nearly all multivariable optimization methods do the following:
  1. Choose a search direction $\mathbf{d}^k$
  2. Minimize along that direction to find a new point:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$$

where k is the current iteration number and α^k is a positive scalar called the step size.

The Step Size

- The step size, α^{k}, is calculated in the following way:
- We want to minimize the function $f(\mathbf{x}^{k+1}) = f(\mathbf{x}^{k} + \alpha^{k}\mathbf{d}^{k})$ where the only variable is α^{k} because $\mathbf{x}^{k}$ & $\mathbf{d}^{k}$ are known.
- We set $\frac{\mathrm{d}f\left(\mathbf{x}^{k} + \alpha^{k}\mathbf{d}^{k}\right)}{\mathrm{d}\alpha^{k}} = 0$ and solve for α^{k} using a single-variable solution method such as the ones shown previously.

Steepest Descent Method

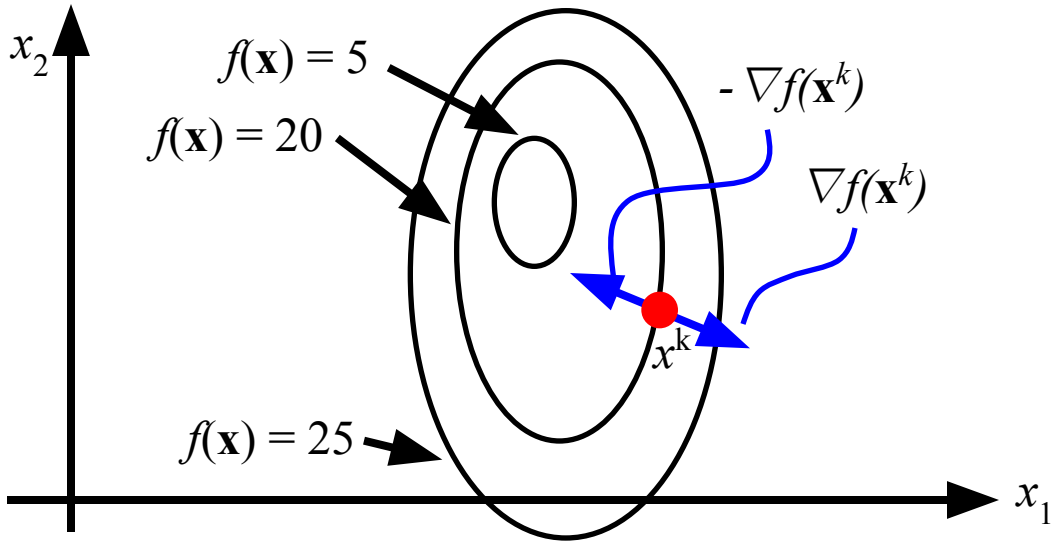
- This method is very simple – it uses the gradient (for maximization) or the negative gradient (for minimization) as the search direction:

$$\mathbf{d}^k = \begin{Bmatrix} + \\ - \end{Bmatrix} \nabla f(\mathbf{x}^k) \text{ for } \begin{Bmatrix} \max \\ \min \end{Bmatrix}$$

So, $$\mathbf{x}^{k+1} = \mathbf{x}^k \begin{Bmatrix} + \\ - \end{Bmatrix} \alpha^k \nabla f(\mathbf{x}^k)$$

Steepest Descent Method

- Because the gradient is the rate of change of the function at that point, using the gradient (or negative gradient) as the search direction helps reduce the number of iterations needed



Steepest Descent Method Steps

So the steps of the Steepest Descent Method are:

1. Choose an initial point $\mathbf{x}^0$
2. Calculate the gradient $\nabla f(\mathbf{x}^k)$ where k is the iteration number
3. Calculate the search vector: $\mathbf{d}^k = \pm\nabla f(\mathbf{x}^k)$
4. Calculate the next $\mathbf{x}$: $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$ Use a single-variable optimization method to determine α^{k}.

Steepest Descent Method Steps

5. To determine convergence, either use some given tolerance ε_1 and evaluate:

$$\left| f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) \right| < \varepsilon_1$$

for convergence

Or, use another tolerance ε_2 and evaluate:

$$\left\| \nabla f(\mathbf{x}^k) \right\| < \varepsilon_2$$

for convergence

Convergence

- These two criteria can be used for any of the multivariable optimization methods discussed here

Recall: The norm of a vector, $\|\mathbf{x}\|$ is given by:

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \cdot \mathbf{x}} = \sqrt{(x_1)^2 + (x_2)^2 + \square \quad + (x_n)^2}$$

Steepest Descent Example

Let's solve the earlier problem with the Steepest Descent Method:

Minimize $f(x_1, x_2, x_3) = (x_1)^2 + x_1(1 - x_2) + (x_2)^2 - x_2x_3 + (x_3)^2 + x_3$

Let's pick $\mathbf{x}^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Steepest Descent Example

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 + (1 - x_2) & -x_1 + 2x_2 - x_3 & -x_2 + 2x_3 + 1 \end{bmatrix}$$

$$\mathbf{d}^0 = -\nabla f(\mathbf{x}^0) = -\begin{bmatrix} 2(0) + 1 - 0 & -0 + 0 - 0 & -0 + 0 + 1 \end{bmatrix}$$

$$= -\begin{bmatrix} 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & -1 \end{bmatrix}$$

$$\mathbf{x}^1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} + \alpha^0 \cdot \begin{bmatrix} -1 & 0 & -1 \end{bmatrix}$$

Now, we need to determine α^0

Steepest Descent Example

$$f(\mathbf{x}^1) = (\alpha^0)^2 + (-\alpha^0)(1) + 0 - 0 + (\alpha^0)^2 + (-\alpha^0)$$

$$= 2(\alpha^0)^2 - 2(\alpha^0)$$

$$\frac{df(\mathbf{x}^1)}{d\alpha^0} = 4(\alpha^0) - 2$$

Now, set equal to zero and solve:

$$4(\alpha^0) = 2 \;\Rightarrow\; \alpha^0 = 2/4 = 1/2$$

Steepest Descent Example

So,

$$\mathbf{x}^1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} + \alpha^0 \cdot \begin{bmatrix} -1 & 0 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix}$$

$$\therefore \mathbf{x}^1 = \begin{bmatrix} -\frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix}$$

Steepest Descent Example

Take the negative gradient to find the next search direction:

$$\mathbf{d}^1 = -\nabla f(\mathbf{x}^1) = -\begin{bmatrix} -1+1+0 & \frac{1}{2}+0+\frac{1}{2} & 0-1+1 \end{bmatrix}$$

$$\therefore \mathbf{d}^1 = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}$$

Steepest Descent Example

Update the iteration formula:

$$\mathbf{x}^2 = \begin{bmatrix} -\frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix} + \alpha^1 \cdot \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{1}{2} & -\alpha^1 & -\frac{1}{2} \end{bmatrix}$$

Steepest Descent Example

Insert into the original function & take the derivative so that we can find α^1:

$$f(\mathbf{x}^2) = \frac{1}{4} + \left(-\frac{1}{2}\right)\left(1+\alpha^1\right) + (\alpha^1)^2 - (\alpha^1)\left(\frac{1}{2}\right) + \frac{1}{4} - \frac{1}{2}$$

$$= (\alpha^1)^2 - \alpha^1 - 1/2$$

$$\frac{df(\mathbf{x}^1)}{d\alpha^1} = 2(\alpha^1) - 1$$

Steepest Descent Example

Now we can set the derivative equal to zero and solve for α^1:

$$2(\alpha^1) = 1 \quad \Rightarrow \quad \alpha^1 = 1/2$$

Steepest Descent Example

Now, calculate $\mathbf{x}^2$:

$$\mathbf{x}^2 = \begin{bmatrix} -\frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix} + \alpha^1 \cdot \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix} + \begin{bmatrix} 0 & -\frac{1}{2} & 0 \end{bmatrix}$$

$$\therefore \mathbf{x}^2 = \begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

Steepest Descent Example

$$\mathbf{d}^2 = -\nabla f(\mathbf{x}^2) = -\left[-1+1+\frac{1}{2} \quad \frac{1}{2}-1+\frac{1}{2} \quad \frac{1}{2}-1+1\right]$$

$$\therefore \mathbf{d}^2 = \left[-\frac{1}{2} \quad 0 \quad -\frac{1}{2}\right]$$

So,

$$\mathbf{x}^3 = \left[-\frac{1}{2} \quad -\frac{1}{2} \quad -\frac{1}{2}\right] + \alpha^2 \cdot \left[-\frac{1}{2} \quad 0 \quad -\frac{1}{2}\right]$$

$$= \left[-\frac{1}{2}(\alpha^2+1) \quad -\frac{1}{2} \quad -\frac{1}{2}(\alpha^2+1)\right]$$

Steepest Descent Example

Find α^2: $$f(\mathbf{x}^3) = \frac{1}{2}(\alpha^2 + 1)^2 - \frac{3}{2}(\alpha^2 + 1) + \frac{1}{4}$$

$$\frac{df(\mathbf{x}^3)}{d\alpha^2} = (\alpha^2 + 1) - \frac{3}{2}$$

Set the derivative equal to zero and solve:

$$(\alpha^2 + 1) = \frac{3}{2} \Rightarrow \alpha^2 = 1/2$$

Steepest Descent Example

Calculate $\mathbf{x}^3$:

$$\mathbf{x}^3 = \begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} + \alpha^2 \cdot \begin{bmatrix} -\frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} + \begin{bmatrix} -\frac{1}{4} & 0 & -\frac{1}{4} \end{bmatrix}$$

$$\therefore \mathbf{x}^3 = \begin{bmatrix} -\frac{3}{4} & -\frac{1}{2} & -\frac{3}{4} \end{bmatrix}$$

Steepest Descent Example

Find the next search direction:

$$\mathbf{d}^3 = -\nabla f(\mathbf{x}^3) = -\begin{bmatrix} 0 & \frac{1}{2} & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{2} & 0 \end{bmatrix}$$

$$\mathbf{x}^4 = \begin{bmatrix} -\frac{3}{4} & -\frac{1}{2} & -\frac{3}{4} \end{bmatrix} + \alpha^3 \cdot \begin{bmatrix} 0 & -\frac{1}{2} & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{3}{4} & -\frac{1}{2}(\alpha^3 + 1) & -\frac{3}{4} \end{bmatrix}$$

Steepest Descent Example

Find α^3:

$$f(\mathbf{x}^4) = \frac{1}{4}(\alpha^3 + 1)^2 - \frac{3}{2}(\alpha^3) - \frac{3}{2}$$

$$\frac{df(\mathbf{x}^4)}{d\alpha^3} = \frac{1}{2}(\alpha^3 + 1) - \frac{9}{8} = 0$$

$$\Rightarrow \alpha^3 = \frac{5}{4}$$

Steepest Descent Example

So, $\mathbf{x}^4$ becomes:

$$\mathbf{x}^4 = \begin{bmatrix} -\frac{3}{4} & -\frac{1}{2} & -\frac{3}{4} \end{bmatrix} + \begin{bmatrix} 0 & -\frac{5}{8} & 0 \end{bmatrix}$$

$$\therefore \mathbf{x}^4 = \begin{bmatrix} -\frac{3}{4} & -\frac{9}{8} & -\frac{3}{4} \end{bmatrix}$$

Steepest Descent Example

The next search direction:

$$\mathbf{d}^4 = -\nabla f(\mathbf{x}^4) = -\begin{bmatrix} \frac{5}{8} & -\frac{3}{4} & \frac{5}{8} \end{bmatrix} = \begin{bmatrix} -\frac{5}{8} & \frac{3}{4} & -\frac{5}{8} \end{bmatrix}$$

$$\mathbf{x}^5 = \begin{bmatrix} -\frac{3}{4} & -\frac{9}{8} & -\frac{3}{4} \end{bmatrix} + \alpha^4 \cdot \begin{bmatrix} -\frac{5}{8} & \frac{3}{4} & -\frac{5}{8} \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{1}{4}(3+\frac{5}{2}\alpha^4) & -\frac{3}{4}(\frac{3}{2}-\alpha^4) & -\frac{1}{4}(3+\frac{5}{2}\alpha^4) \end{bmatrix}$$

Steepest Descent Example

Find α^4:

$$f(\mathbf{x}^5) = \frac{73}{32}(\alpha^4)^2 - \frac{43}{32}\alpha^4 - \frac{51}{64}$$

$$\frac{df(\mathbf{x}^5)}{d\alpha^4} = \frac{73}{16}\alpha^4 - \frac{43}{32} = 0$$

$$\Rightarrow \alpha^4 = {}^{43}\!/_{146}$$

Steepest Descent Example

Update $\mathbf{x}^5$:

$$\mathbf{x}^5 = \begin{bmatrix} -\frac{3}{4} & -\frac{9}{8} & -\frac{3}{4} \end{bmatrix} + \frac{43}{146} \cdot \begin{bmatrix} -\frac{5}{8} & \frac{3}{4} & -\frac{5}{8} \end{bmatrix}$$

$$\therefore \mathbf{x}^5 = \begin{bmatrix} -\frac{1091}{1168} & -\frac{66}{73} & -\frac{1091}{1168} \end{bmatrix}$$

Steepest Descent Example

Let's check to see if the convergence criteria is satisfied

Evaluate $\|\nabla f(\mathbf{x}^5)\|$:

$$\nabla f(\mathbf{x}^5) = \begin{bmatrix} \frac{21}{584} & \frac{35}{584} & \frac{21}{584} \end{bmatrix}$$

$$\left\|\nabla f(\mathbf{x}^5)\right\| = \sqrt{\left(21/584\right)^2 + \left(35/584\right)^2 + \left(21/584\right)^2} = 0.0786$$

Steepest Descent Example

So, $\|\nabla f(\mathbf{x}^5)\| = 0.0786$, which is very small and we can take it to be close enough to zero for our example

Notice that the answer of

$$\mathbf{x} = \begin{bmatrix} -\frac{1091}{1168} & -\frac{66}{73} & -\frac{1091}{1168} \end{bmatrix}$$

is very close to the value of $\mathbf{x}^* = \begin{bmatrix} -1 & -1 & -1 \end{bmatrix}$ that we obtained analytically

Quadratic Functions

- Quadratic functions are important for the next method we will look at
- A quadratic function can be written in the form: $\mathbf{x}^T\mathbf{Q}\mathbf{x}$ where $\mathbf{x}$ is the vector of variables and $\mathbf{Q}$ is a matrix of coefficients

Example:

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 2(x_1)^2 - 2x_1x_2 + 2(x_2)^2 - 2x_2x_3 + 2(x_3)^2$$

Multivariable Newton's Method

We can approximate the gradient of f at a point $\boldsymbol{x}^0$ by:

$$\nabla f(\mathbf{x}) \approx \nabla f(\mathbf{x}^0) + \nabla^2 f(\mathbf{x}^0) \cdot (\mathbf{x} - \mathbf{x}^0)$$

We can set the right-hand side equal to zero and rearrange to give:

$$\mathbf{x} = \mathbf{x}^0 - \left[\nabla^2 f(\mathbf{x}^0)\right]^{-1} \cdot \nabla f(\mathbf{x}^0)$$

Multivariable Newton's Method

We can generalize this equation to give an iterative expression for the Newton's Method:

$$\mathbf{x}^{k+1} = \mathbf{x}^{k} - \left[\nabla^2 f(\mathbf{x}^{k})\right]^{-1} \cdot \nabla f(\mathbf{x}^{k})$$

where k is the iteration number

Newton's Method Steps

1. Choose a starting point, $\mathbf{x}^0$
2. Calculate $\nabla f(\mathbf{x}^k)$ and $\nabla^2 f(\mathbf{x}^k)$
3. Calculate the next $\mathbf{x}$ using the equation

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \left[\nabla^2 f(\mathbf{x}^k)\right]^{-1} \cdot \nabla f(\mathbf{x}^k)$$

4. Use either of the convergence criteria discussed earlier to determine convergence. If it hasn't converged, return to step 2.

Comments on Newton's Method

- We can see that unlike the previous two methods, Newton's Method uses both the gradient and the Hessian
- This usually reduces the number of iterations needed, but increases the computation needed for each iteration
- So, for very complex functions, a simpler method is usually faster

Newton's Method Example

For an example, we will use the same problem as before:

Minimize $f(x_1, x_2, x_3) = (x_1)^2 + x_1(1 - x_2) + (x_2)^2 - x_2x_3 + (x_3)^2 + x_3$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 - x_2 + 1 & -x_1 + 2x_2 - x_3 & -x_2 + 2x_3 + 1 \end{bmatrix}$$

Newton's Method Example

The Hessian is:

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

And we will need the inverse of the Hessian:

$$\left[\nabla^2 f(\mathbf{x})\right]^{-1} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}^{-1} = \begin{bmatrix} 3/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 3/4 \end{bmatrix}$$

Newton's Method Example

So, pick $\mathbf{x}^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Calculate the gradient for the 1st iteration:

$$\nabla f(\mathbf{x}^0) = \begin{bmatrix} 0-0+1 & -0+0-0 & -0+0+1 \end{bmatrix}$$

$$\Rightarrow \nabla f(\mathbf{x}^0) = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$$

Newton's Method Example

So, the new $\mathbf{x}$ is:

$$\mathbf{x}^1 = \mathbf{x}^0 - \left[\nabla^2 f(\mathbf{x}^0)\right]^{-1} \cdot \nabla f(\mathbf{x}^0)$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 3/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 3/4 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\therefore \mathbf{x}^1 = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

Newton's Method Example

Now calculate the new gradient:

$$\nabla f(\mathbf{x}^1) = \begin{bmatrix} -2+1+1 & 1-2+1 & 1-2+1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

Since the gradient is zero, the method has converged

Convex optimization problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_1(x) \leq 0 \\ & \cdots \\ & f_m(x) \leq 0 \end{array}$$

- objective and constraint functions are convex: for $0 \leq \theta \leq 1$

$$f_i(\theta x + (1-\theta)y) \leq \theta f_i(x) + (1-\theta) f_i(y)$$

- includes least-squares problems and linear programs as special cases
- can be solved exactly, with similar complexity as LPs
- surprisingly many problems can be solved via convex optimization

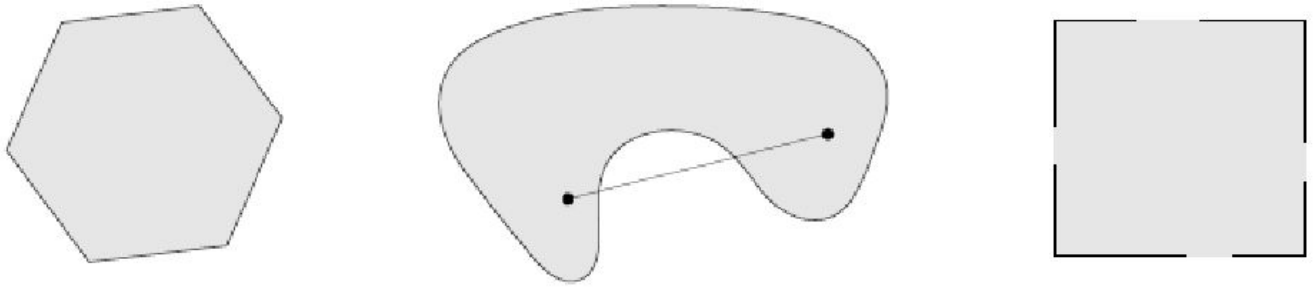
Slides: Boyd and Vandenberghe, Convex optimization

Convex set

contains line segment between any two points in the set

$$x_1, x_2 \in C, \quad 0 \leq \theta \leq 1 \quad \Longrightarrow \quad \theta x_1 + (1-\theta)x_2 \in C$$

Examples: one convex, two nonconvex sets



Examples and properties

- solution set of linear equations $Ax = b$ (affine set)
- solution set of linear inequalities $Ax \preceq b$ (polyhedron)
- norm balls $\{x \mid \|x\| \leq R\}$ and norm cones $\{(x,t) \mid \|x\| \leq t\}$
- set of positive semidefinite matrices $\mathbf{S}^n_+ = \{X \in \mathbf{S}^n \mid X \succeq 0\}$

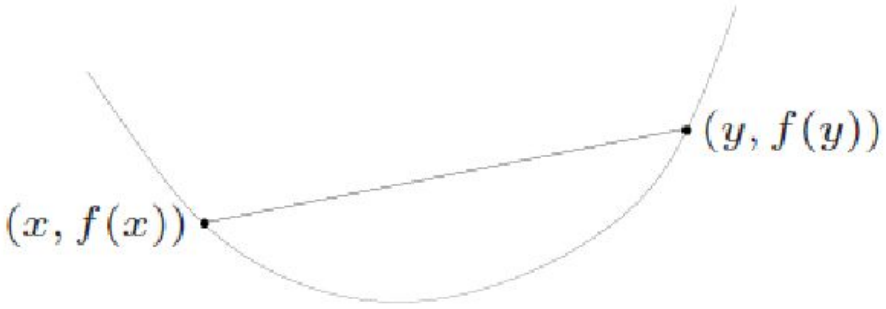
- image of a convex set under a linear transformation is convex
- inverse image of a convex set under a linear transformation is convex
- intersection of convex sets is convex

Convex function

domain $\mathbf{dom}\, f$ is a convex set and

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$$

for all $x, y \in \mathbf{dom}\, f$, $0 \leq \theta \leq 1$



f is concave if $-f$ is convex

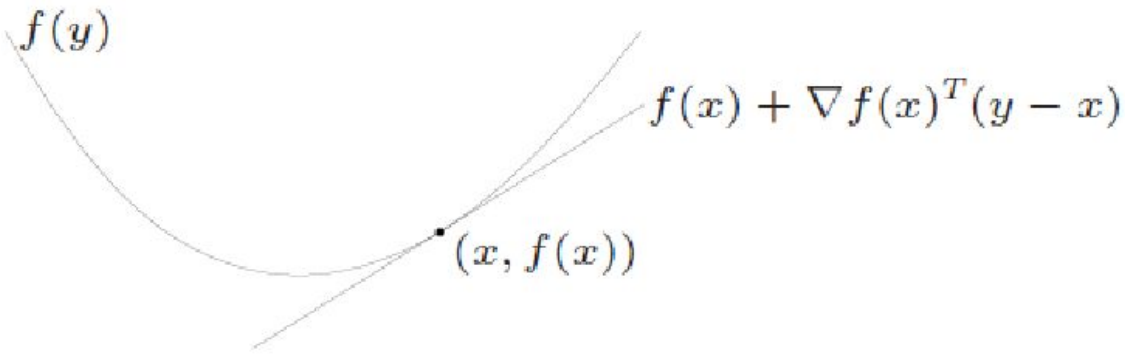
Examples

- $\exp x$, $-\log x$, $x \log x$ are convex
- x^α is convex for $x > 0$ and $\alpha \geq 1$ or $\alpha \leq 0$; $|x|^\alpha$ is convex for $\alpha \geq 1$
- quadratic-over-linear function $x^T x/t$ is convex in x, t for $t > 0$
- geometric mean $(x_1 x_2 \cdots x_n)^{1/n}$ is concave for $x \succeq 0$
- $\log \det X$ is concave on set of positive definite matrices
- $\log(e^{x_1} + \cdots e^{x_n})$ is convex
- linear and affine functions are convex and concave
- norms are convex

Differentiable convex functions

differentiable f is convex if and only if $\mathbf{dom}\, f$ is convex and

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \quad \text{for all } x, y \in \mathbf{dom}\, f$$



twice differentiable f is convex if and only if $\mathbf{dom}\, f$ is convex and

$$\nabla^2 f(x) \succeq 0 \quad \text{for all } x \in \mathbf{dom}\, f$$

Operations that preserve convexity

methods for establishing convexity of a function

1. verify definition

2. for twice differentiable functions, show $\nabla^2 f(x) \succeq 0$

3. show that f is obtained from simple convex functions by operations that preserve convexity

   - nonnegative weighted sum
   - composition with affine function
   - pointwise maximum and supremum
   - composition
   - minimization
   - perspective

Convex optimization problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{array}$$

$f_0, f_1, \ldots, f_m$ are convex functions

- feasible set is convex
- locally optimal points are globally optimal
- tractable, both in theory and practice

Comments on Example

- Because it uses the 2nd derivative, Newton's Method models quadratic functions exactly and can find the optimum point in one iteration.
- If the function had been a higher order, the Hessian would not have been constant and it would have been much more work to calculate the Hessian and take the inverse for each iteration.

Constrained Nonlinear Optimization

- Previously in this chapter, we solved NLP problems that only had objective functions, with no constraints.
- Now we will look at methods on how to solve problems that include constraints.

Convexity & global vs. local optima

- When minimizing a function, if we want to be sure that we can get a global solution via differentiation, we need to impose some requirements on our objective function.
- We will also need to impose some requirements on the feasible set S (set of possible values the solution $\underline{x}^*$ may take).

Min $f(\underline{x})$

subject to

$\underline{h}(\underline{x})=\underline{c}$

$\underline{g}(\underline{x})\geq \underline{b}$

Feasible set

$$\min f(\underline{x})$$

subject to

$$\underline{x} \in S$$

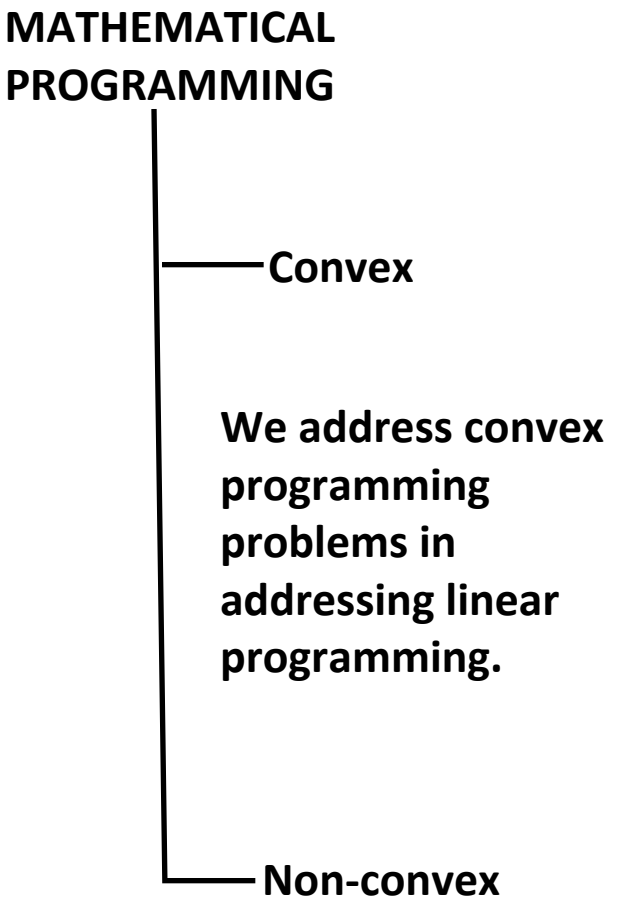
<u>Definition</u>: If f($\underline{x}$) is a convex function, and if S is a convex set, then the above problem is a *convex programming problem*.

<u>Definition</u>: If f($\underline{x}$) is not a convex function, or if S is not a convex set, then the above problem is a *non-convex programming problem*.

Convex vs. nonconvex programming problems

The desirable quality of a convex programming problem is that any *locally optimal solution* is also a *globally optimal solution*. If we have a method of finding a locally optimal solution, that method also finds for us the globally optimum solution.

The undesirable quality of a non-convex programming problem is that any method which finds a locally optimal solution does not necessarily find the globally optimum solution.



A convex programming problem

Two variables with one equality-constraint

$$\min f(x_1, x_2)$$
$$s.t. \quad h(x_1, x_2) = c$$

We focus on this one, but conclusions we derive will also apply to the other two. The benefit of focusing on this one is that we can visualize it.

Multi-variable with one equality-constraint.

$$\min f(\underline{x})$$
$$\text{s.t.} \quad h(\underline{x}) = c$$

Multi-variable with multiple equality-constraints.

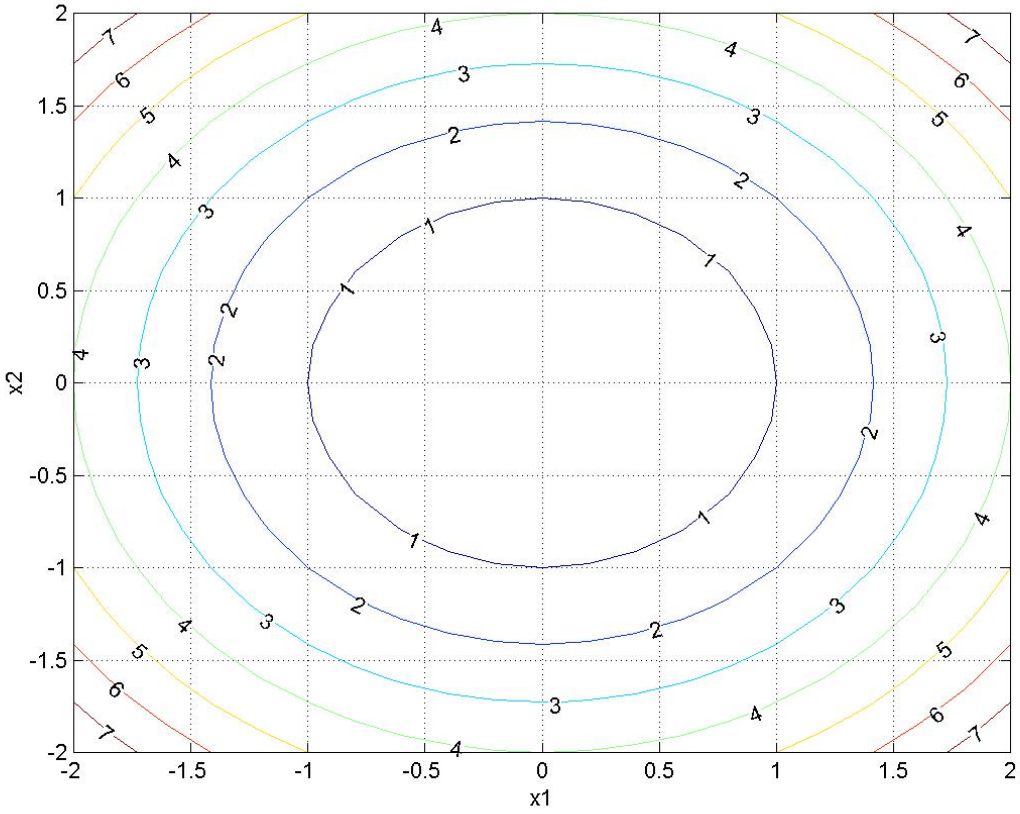
$$\min f(\underline{x})$$
$$\text{s.t.} \quad \underline{h}(\underline{x}) = \underline{c}$$

Contour maps/Constant Cost Contours/level Sets

Definition: A contour map is a 2-dimensional plane, i.e., a coordinate system in 2 variables, say, x_1, x_2, that illustrates curves (contours) of constant functional value $f(x_1, x_2)$.

Example: Draw the contour map for $f(x_1, x_2) = x_1^2 + x_2^2$

```
[X,Y] =
meshgrid(-2.0:.2:2.0,-2.0:.
:2.0);
Z = X.^2+Y.^2;
[c,h]=contour(X,Y,Z);
clabel(c,h);
grid;
xlabel('x1');
ylabel('x2');
```

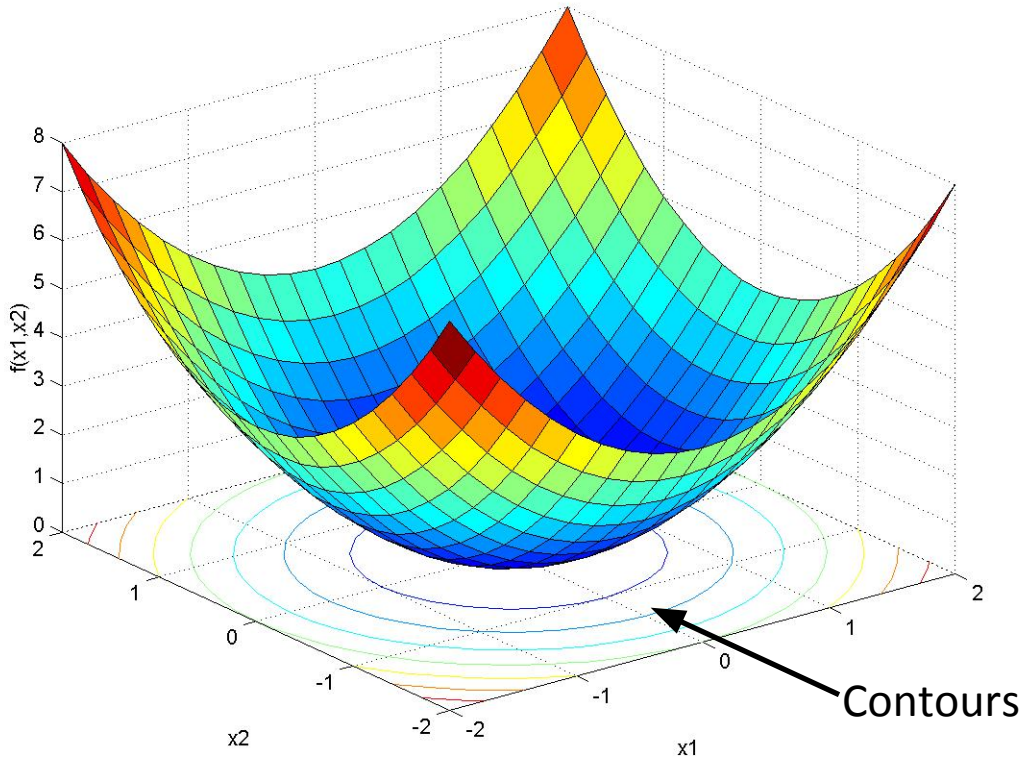


Contour maps and 3-D illustrations

Example: Draw the 3-D surface for $f(x_1, x_2) = x_1^2 + x_2^2$

```
[X,Y] =
meshgrid(-2.0:.2:2.0,-2.0:.2
:2.0);
Z = X.^2+Y.^2;
surfc(X,Y,Z)
xlabel('x1')
ylabel('x2')
zlabel('f(x1,x2)')
```

Height is f(x)



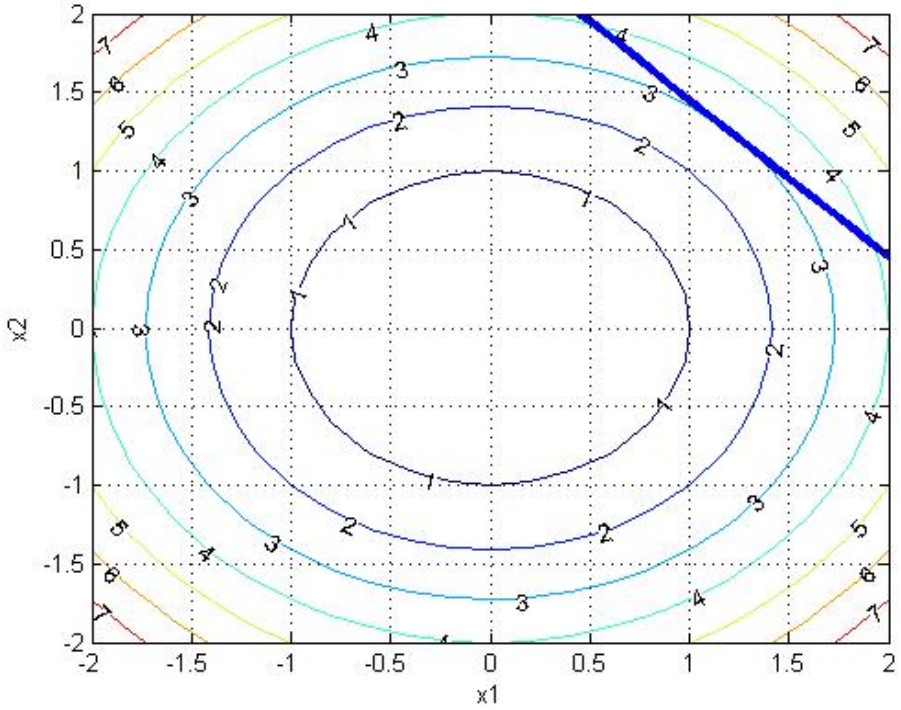
Each contour of fixed value *f* is the projection onto the *x1-x2* plane of a horizontal slice made of the 3-D figure at a value *f* above the *x1-x2* plane.

Solving a convex program: graphical analysis

Example: Solve this convex program: $\min f(x_1, x_2) = x_1^2 + x_2^2$

$s.t. \quad h(x_1, x_2) = x_1 + x_2 = \sqrt{6}$

A straight line is a convex set because a line segment between any two points on it remain on it.



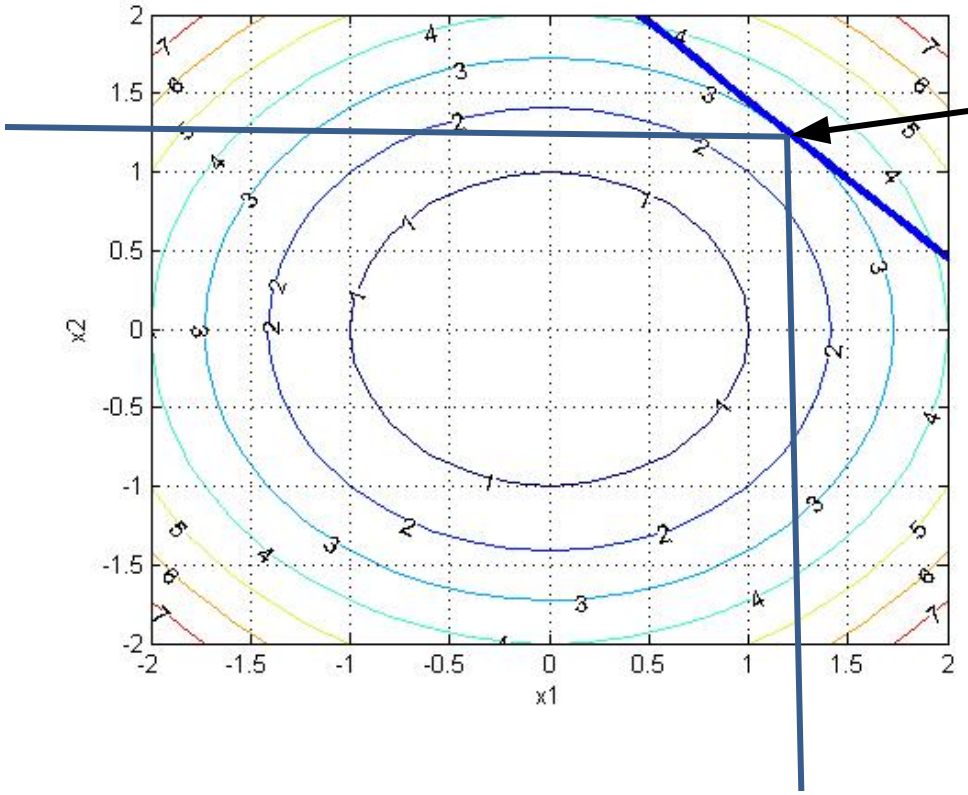
$x_1 + x_2 = \sqrt{6} \Rightarrow x_2 = -x_1 + \sqrt{6}$

Superimpose this relation on top of the contour plot for $f(x_1, x_2)$.

***1. $f(x_1, x_2)$* must be minimized, and so we would like the solution to be as close to the origin as possible;**

2. The solution must be on the thick line in the right-hand corner of the plot, since this line represents the equality constraint.

Solving a convex program: graphical analysis



Solution:

$$\underline{x}^* = (x_1, x_2)^* \approx (1.25, 1.25)$$
$$f(x_1, x_2)^* = 3$$

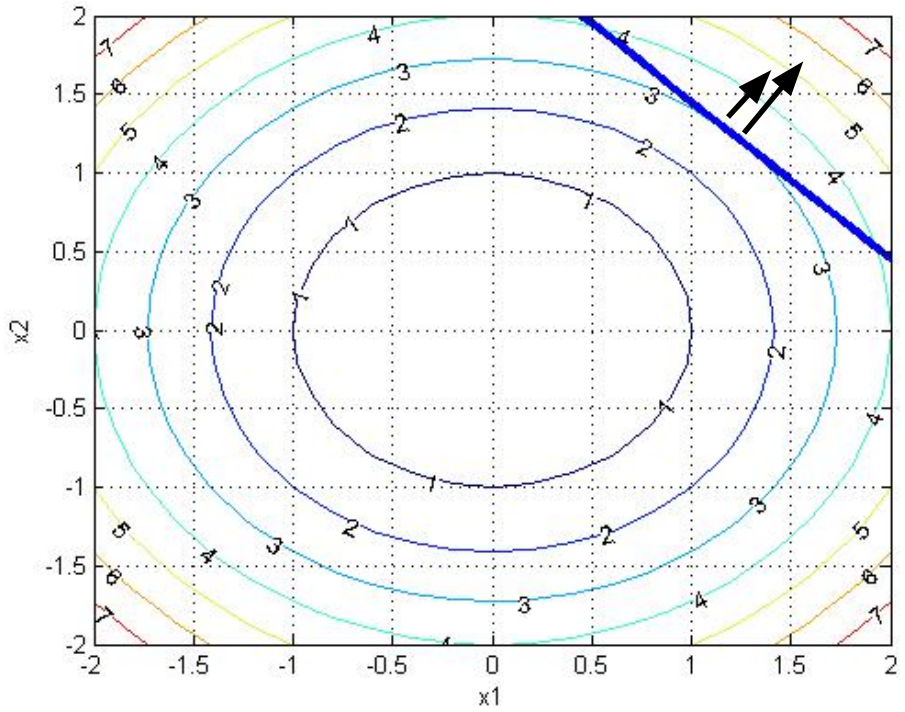
Any contour *f<3* does not intersect the equality constraint;
Any contour *f>3* intersects the equality constraint at two points.

The contour f=3 and the equality constraint just touch each other at the point $\underline{x}^*$.

"Just touch":
The two curves are <u>tangent</u> to one another at the solution point.

Solving a convex program: graphical analysis

The two curves are <u>tangent</u> to one another at the solution point.



The normal (gradient) vectors of the two curves, at the solution (tangent) point, are parallel.

This means the following two vectors are parallel:

$$\nabla\{f(x_1,x_2)^*\} = \nabla\left\{x_1^2 + x_2^2\right\}^* = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}^*$$

$$\nabla\{h(x_1,x_2)^* - \sqrt{6}\} = \nabla\left\{x_1 + x_2 - \sqrt{6}\right\}^* = \begin{bmatrix} 1 \\ 1 \end{bmatrix}^*$$

"Parallel" means that the two vectors have the same direction. We do not know that they have the same magnitude. To account for this, we equate with a "multiplier" λ:

$$\nabla f(x_1,x_2)^* = \lambda \nabla\left(h(x_1,x_2)^* - c\right)$$

Solving a convex program: graphical analysis

$$\nabla f(x_1, x_2)^* = \lambda \nabla \left(h(x_1, x_2)^* - c \right)$$

Moving everything to the left:

$$\nabla f(x_1, x_2)^* - \lambda \nabla \left(h(x_1, x_2)^* - c \right) = 0$$

Alternately:

$$\nabla f(x_1, x_2)^* + \lambda \nabla \left(c - h(x_1, x_2)^* \right) = 0$$

Performing the gradient operation (taking derivatives with respect to x_1 and x_2) :

$$\begin{bmatrix} \frac{\partial}{\partial x_1} \left(f(x_1, x_2) - \lambda \left(h(x_1, x_2) - c \right) \right) \\ \frac{\partial}{\partial x_2} \left(f(x_1, x_2) - \lambda \left(h(x_1, x_2) - c \right) \right) \end{bmatrix}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In this problem, we already know the solution, but what if we did not? Then could we use the above equations to find the solution?

Solving a convex program: analytical analysis

In this problem, we already know the solution, but what if we did not? Then could we use the above equations to find the solution?

$$\begin{bmatrix} \frac{\partial}{\partial x_1}\left(f(x_1,x_2)-\lambda\left(h(x_1,x_2)-c\right)\right) \\ \frac{\partial}{\partial x_2}\left(f(x_1,x_2)-\lambda\left(h(x_1,x_2)-c\right)\right) \end{bmatrix}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

NO! Because we only have 2 equations, yet 3 unknowns: x_1, x_2, λ. So we need another equation. Where do we get that equation?

Recall our equality constraint: $h(x_1, x_2)-c=0$. **This must be satisfied!**

Therefore:

$$\begin{bmatrix} \frac{\partial}{\partial x_1}\left(f(x_1,x_2)-\lambda\left(h(x_1,x_2)-c\right)\right) \\ \frac{\partial}{\partial x_2}\left(f(x_1,x_2)-\lambda\left(h(x_1,x_2)-c\right)\right) \\ h(x_1,x_2)-c \end{bmatrix}^* = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Three equations, three unknowns, we can solve.

Solving a convex program: analytical analysis

Observation: The three equations are simply partial derivatives of the function

$$f(x_1, x_2) - \lambda\left(h(x_1, x_2) - c\right)$$

$$\begin{bmatrix} \frac{\partial}{\partial x_1}\left(f(x_1, x_2) - \lambda\left(h(x_1, x_2) - c\right)\right) \\ \frac{\partial}{\partial x_2}\left(f(x_1, x_2) - \lambda\left|h(x_1, x_2) - c\right|\right) \\ h(x_1, x_2) - c \end{bmatrix}^* = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

This is obviously true for the first two equations , but it is not so obviously true for the last one. But to see it, observe

$$\frac{\partial}{\partial \lambda}\left(f(x_1, x_2) - \lambda\left(h(x_1, x_2) - c\right)\right) = 0$$

$$\Rightarrow -h(x_1, x_2) + c = 0 \Rightarrow h(x_1, x_2) = c$$

Formal approach to solving our problem

Define the Lagrangian function:

$$\mathsf{L}(x_1, x_2, \lambda) = f(x_1, x_2) - \lambda\big(h(x_1, x_2) - c\big)$$

In a convex programming problem, the "first-order conditions" for finding the solution is given by

$$\nabla \mathsf{L}(x_1, x_2, \lambda) = 0$$

OR

$$\frac{\partial}{\partial x_1}\mathsf{L}(x_1, x_2, \lambda) = 0$$

$$\frac{\partial}{\partial x_2}\mathsf{L}(x_1, x_2, \lambda) = 0$$

$$\frac{\partial}{\partial \lambda}\mathsf{L}(x_1, x_2, \lambda) = 0$$

Or more compactly

$$\frac{\partial}{\partial \underline{x}}\mathsf{L}(\underline{x}, \lambda) = 0$$

$$\frac{\partial}{\partial \lambda}\mathsf{L}(\underline{x}, \lambda) = 0$$

where we have used $\underline{x}=(x_1, x_2)$

Applying to our example

Define the Lagrangian function:

$$\mathrm{L}(x_1, x_2, \lambda) = f(x_1, x_2) - \lambda\left(h(x_1, x_2) - c\right)$$
$$= x_1^2 + x_2^2 - \lambda\left(x_1 + x_2 - \sqrt{6}\right)$$

$$\nabla \mathrm{L}(x_1, x_2, \lambda) = 0$$

$$\frac{\partial}{\partial x_1}\mathrm{L}(x_1, x_2, \lambda) = 0 \qquad \frac{\partial}{\partial x_1}\mathrm{L}(x_1, x_2, \lambda) = 2x_1 - \lambda = 0$$

OR $\frac{\partial}{\partial x_2}\mathrm{L}(x_1, x_2, \lambda) = 0$ ⇒ $\frac{\partial}{\partial x_2}\mathrm{L}(x_1, x_2, \lambda) = 2x_2 - \lambda = 0$

$$\frac{\partial}{\partial \lambda}\mathrm{L}(x_1, x_2, \lambda) = 0 \qquad \frac{\partial}{\partial \lambda}\mathrm{L}(x_1, x_2, \lambda) = -\left(x_1 + x_2 - \sqrt{6}\right) = 0$$

A set of 3 linear equations and 3 unknowns; we can write in the form of Ax=b.

Applying to our example

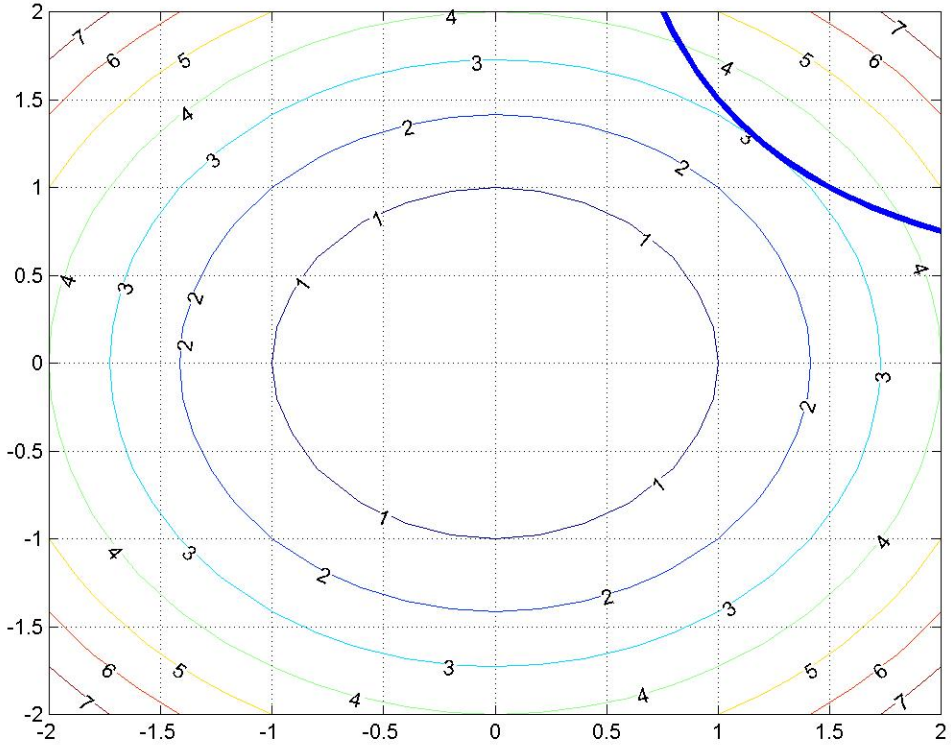
$$\begin{bmatrix} 2 & 0 & -1 \\ 0 & 2 & -1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sqrt{6} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 2 & -1 \\ 1 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \sqrt{6} \end{bmatrix} = \begin{bmatrix} 1.2247 \\ 1.2247 \\ 2.4495 \end{bmatrix}$$

Now, let’s go back to our example with a nonlinear equality constraint.

Example with nonlinear equality

Non-convex because a line connecting two points in the set do not remain in the set. (see "notes" of this slide)



$$\min f(x_1, x_2) = x_1^2 + x_2^2$$

$$s.t. \quad h(x_1, x_2) = 2x_1x_2 = 3$$

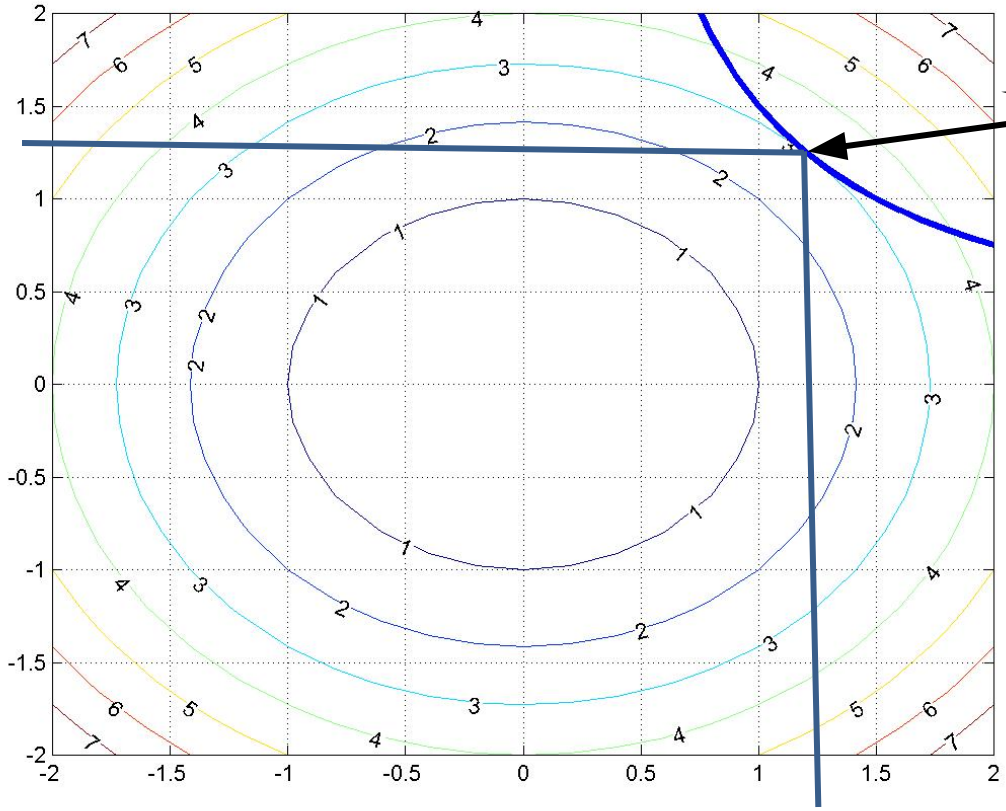
$$2x_1x_2 = 3 \Rightarrow x_2 = \frac{3}{2x_1}$$

Superimpose this relation on top of the contour plot for $f(x_1,x_2)$.

***1. $f(x_1,x_2)$* must be minimized, and so we would like the solution to be as close to the origin as possible;**

2. The solution must be on the thick line in the right-hand corner of the plot, since this line represents the equality constraint.

Example with nonlinear equality



Solution:

$$\underline{x}^* = (x_1, x_2)^* \approx (1.25, 1.25)$$

$$f(x_1, x_2)^* = 3$$

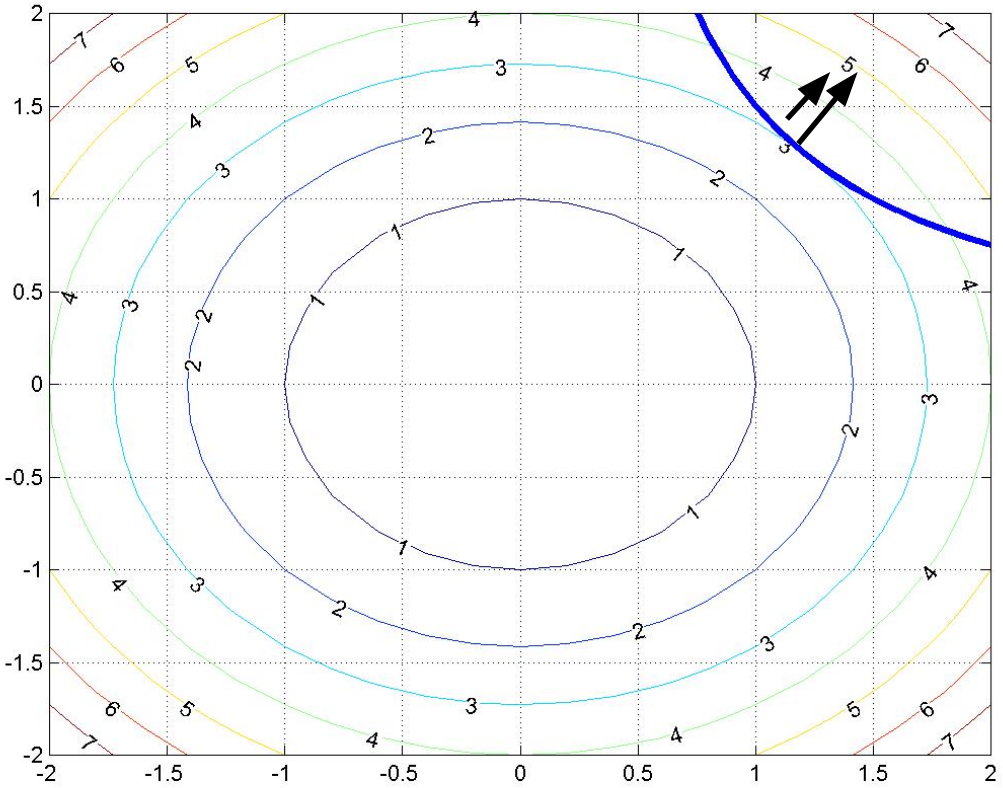
Any contour *f<3* does not intersect the equality constraint;
Any contour *f>3* intersects the equality constraint at two points.

The contour f=3 and the equality constraint just touch each other at the point $\underline{x}^*$.

"Just touch":
The two curves are <u>tangent</u> to one another at the solution point.

Example with nonlinear equality

The two curves are <u>tangent</u> to one another at the solution point.



The normal (gradient) vectors of the two curves, at the solution (tangent) point, are parallel.

This means the following two vectors are parallel:

$$\nabla\{f(x_1,x_2)^*\} = \nabla\left\{x_1^2 + x_2^2\right\}^* = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}^*$$

$$\nabla\{h(x_1,x_2)^* - 3\} = \nabla\left\{2x_1x_2 - 3\right\}^* = \begin{bmatrix} 2x_2 \\ 2x_1 \end{bmatrix}^*$$

"Parallel" means that the two vectors have the same direction. We do not know that they have the same magnitude. To account for this, we equate with a "multiplier" λ:

$$\nabla f(x_1,x_2)^* = \lambda\nabla\left(h(x_1,x_2)^* - c\right)$$

Example with nonlinear equality

This gives us the following two equations.

$$\begin{bmatrix} \frac{\partial}{\partial x_1}\left(f(x_1,x_2)-\lambda\left(h(x_1,x_2)-c\right)\right) \\ \frac{\partial}{\partial x_2}\left(f(x_1,x_2)-\lambda\left(h(x_1,x_2)-c\right)\right) \end{bmatrix}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

And we add the equality constraint to give 3 equations, 3 unknowns:

$$\begin{bmatrix} \frac{\partial}{\partial x_1}\left(f(x_1,x_2)-\lambda\left(h(x_1,x_2)-c\right)\right) \\ \frac{\partial}{\partial x_2}\left(f(x_1,x_2)-\lambda\left(h(x_1,x_2)-c\right)\right) \\ h(x_1,x_2)-c \end{bmatrix}^* = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Three equations, three unknowns, we can solve.

Example with nonlinear equality

Define the Lagrangian function:

$$\mathsf{L}(x_1, x_2, \lambda) = f(x_1, x_2) - \lambda\left(h(x_1, x_2) - c\right)$$

$$= x_1^2 + x_2^2 - \lambda\left(2x_1x_2 - 3\right)$$

$$\nabla \mathsf{L}(x_1, x_2, \lambda) = 0$$

$$\frac{\partial}{\partial x_1}\mathsf{L}(x_1, x_2, \lambda) = 0 \qquad \frac{\partial}{\partial x_1}\mathsf{L}(x_1, x_2, \lambda) = 2x_1 - 2\lambda x_2 = 0$$

OR $\frac{\partial}{\partial x_2}\mathsf{L}(x_1, x_2, \lambda) = 0$ ⇒ $\frac{\partial}{\partial x_2}\mathsf{L}(x_1, x_2, \lambda) = 2x_2 - 2\lambda x_1 = 0$

$$\frac{\partial}{\partial \lambda}\mathsf{L}(x_1, x_2, \lambda) = 0 \qquad \frac{\partial}{\partial \lambda}\mathsf{L}(x_1, x_2, \lambda) = -\left(2x_1x_2 - 3\right) = 0$$

You can solve this algebraically to obtain

$$x_1 = x_2 = \sqrt{\frac{3}{2}} = 1.2247$$

$$x_1 = x_2 = -\sqrt{\frac{3}{2}} = -1.2247$$

and f=3 in both cases

Example with nonlinear equality

Our approach worked in this case, i.e., we found a local optimal point that was also a global optimal point, but because it was not a convex programming problem, we had no guarantee that this would happen.

The conditions we established, below, we call first order conditions.

For convex programming problems, they are first order *sufficient conditions* to provide the global optimal point.

For nonconvex programming problems, they are first order *necessary conditions* to provide the global optimal point.

$$\frac{\partial}{\partial \underline{x}} \mathsf{L}(\underline{x}, \lambda) = 0$$

$$\frac{\partial}{\partial \lambda} \mathsf{L}(\underline{x}, \lambda) = 0$$

Multiple equality constraints

$$\min \; f(\underline{x})$$
$$s.t. \quad \underline{h}(\underline{x}) = \underline{c}$$

We assume that f and h are continuously differentiable.

$$\mathsf{L}(\underline{x}, \underline{\lambda}) = f(\underline{x}) - \lambda_1\left(h_1(\underline{x}) - c_1\right) - \lambda_2\left(h_2(\underline{x}) - c_2\right) - \ldots - \lambda_m\left(h_m(\underline{x}) - c_m\right)$$

First order necessary conditions that ($\underline{\mathbf{x}}$*, $\underline{\boldsymbol{\lambda}}$*) solves the above:

$$\frac{\partial}{\partial \underline{x}} \mathsf{L}(\underline{x}^*, \underline{\lambda}^*) = 0$$

$$\frac{\partial}{\partial \underline{\lambda}} \mathsf{L}(\underline{x}^*, \underline{\lambda}^*) = 0$$

Multiple equality & 1 inequality constraint

$$\min f(\underline{x})$$
$$s.t. \quad \underline{h}(\underline{x}) = \underline{c}$$
$$g(\underline{x}) \geq b$$

We assume that f, h, and g are continuously differentiable.

Solution approach:

- **Ignore the inequality constraint and solve the problem. (this is just a problem with multiple equality constraints).**
- **If inequality constraint is satisfied, then problem is solved.**
- **If inequality constraint is violated, then the inequality constraint must be <u>binding</u> □ inequality constraint enforced with equality:**

$$g(\underline{x}) = b$$

Let's look at this new problem where the inequality is binding.

Multiple equality & 1 inequality constraint

$$\min f(\underline{x})$$
$$s.t. \quad \underline{h}(\underline{x}) = \underline{c}$$
$$g(\underline{x}) = b$$

We assume that f, h, and g are continuously differentiable.

$$\mathsf{L}(\underline{x}, \underline{\lambda}, \mu) = f(\underline{x}) - \lambda_1\left(h_1(\underline{x}) - c_1\right) - \lambda_2\left(h_2(\underline{x}) - c_2\right) - \ldots - \lambda_m\left(h_m(\underline{x}) - c_m\right) - \mu\left(g(\underline{x}) - b\right)$$

First order necessary conditions that ($\underline{\mathbf{x}}$*, $\underline{\boldsymbol{\lambda}}$*, μ*) solves the above:

$$\frac{\partial}{\partial \underline{x}}\mathsf{L}(\underline{x}^*, \underline{\lambda}^*, \mu^*) = 0$$

$$\frac{\partial}{\partial \underline{\lambda}}\mathsf{L}(\underline{x}^*, \underline{\lambda}^*, \mu^*) = 0$$

$$\frac{\partial}{\partial \mu}\mathsf{L}(\underline{x}^*, \underline{\lambda}^*, \mu^*) = 0$$

We were able to write down this solution only after we knew the inequality constraint was binding. Can we generalize this approach?

Multiple equality & 1 inequality constraint

$$L(\underline{x},\underline{\lambda},\mu) = f(\underline{x}) - \lambda_1\left(h_1(\underline{x}) - c_1\right) - \lambda_2\left(h_2(\underline{x}) - c_2\right) - \ldots - \lambda_m\left(h_m(\underline{x}) - c_m\right) - \mu\left(g(\underline{x}) - b\right)$$

If inequality is not binding, then apply first order necessary conditions by ignoring it:

- **μ=0**
- **g(x)-b≠0 (since it is not binding!)**

If inequality is binding, then apply first order necessary conditions treating inequality constraint as an equality constraint

- **μ≠0**
- **g(x)-b≠0 (since it is binding!)**

Either way:
μ(g(x)-b)=0

This relation encodes our solution procedure!
It can be used to generalize our necessary conditions

Multiple equality & multiple inequality constraints

$$\min f(\underline{x})$$
$$s.t. \quad \underline{h}(\underline{x}) = \underline{c}$$
$$\underline{g}(\underline{x}) = \underline{b}$$

We assume that f, h, and g are continuously differentiable.

$$\mathsf{L}(\underline{x}, \underline{\lambda}, \underline{\mu}) = f(\underline{x}) - \lambda_1\left(h_1(\underline{x}) - c_1\right) - \lambda_2\left(h_2(\underline{x}) - c_2\right) - \ldots - \lambda_m\left(h_m(\underline{x}) - c_m\right)$$
$$- \mu_1\left(g_1(\underline{x}) - b_1\right) - \mu_2\left(g_1(\underline{x}) - b_1\right) - \ldots - \mu_n\left(g_1(\underline{x}) - b_1\right)$$

First order necessary conditions that ($\underline{x}^*$, $\underline{\lambda}^*$, $\underline{\mu}^*$) solves the above:

$$\frac{\partial}{\partial \underline{x}} \mathsf{L}(\underline{x}^*, \underline{\lambda}^*, \underline{\mu}^*) = 0$$
$$\frac{\partial}{\partial \underline{\lambda}} \mathsf{L}(\underline{x}^*, \underline{\lambda}^*, \underline{\mu}^*) = 0$$
$$\frac{\partial}{\partial \underline{\mu}} \mathsf{L}(\underline{x}^*, \underline{\lambda}^*, \underline{\mu}^*) = 0$$
$$\mu_k^*\left(g_k(\underline{x}^*) - b\right) = 0 \quad \forall k$$
$$\mu_k^* \geq 0 \quad \forall k$$

These conditions also referred to as the Kurash-Kuhn-Tucker (KKT) conditions

Complementarity condition: Inactive constraints have a zero multiplier.

Nonnegativity on inequality multipliers.

An additional requirement

$$\min f(\underline{x})$$
$$s.t. \quad \underline{h}(\underline{x}) = \underline{c}$$
$$\underline{g}(\underline{x}) = \underline{b}$$

We assume that f, h, and g are continuously differentiable.

For KKT to guarantee finds a local optimum, we need the Kuhn-Tucker Constraint Qualification (even under convexity).

This condition imposes a certain restriction on the constraint functions .

Its purpose is to rule out certain irregularities on the boundary of the feasible set, that would invalidate the Kuhn-Tucker conditions should the optimal solution occur there.

We will not try to tackle this idea, but know this:

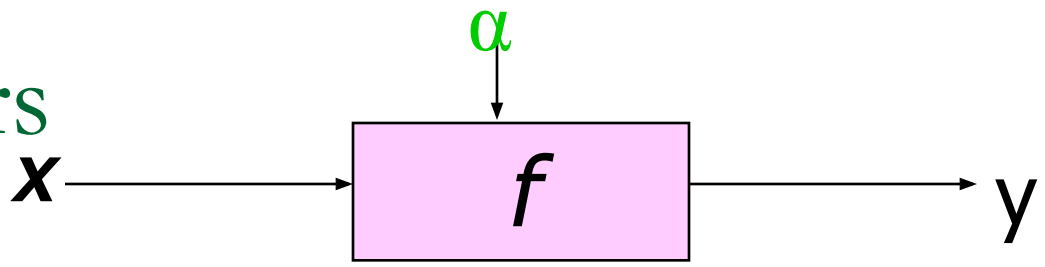
- **If the feasible region is a convex set formed by *linear* constraints only, then the constraint qualification will be met, and the Kuhn-Tucker conditions will always hold at an optimal solution.**

A Case Study: Support Vector Machines

Introduction

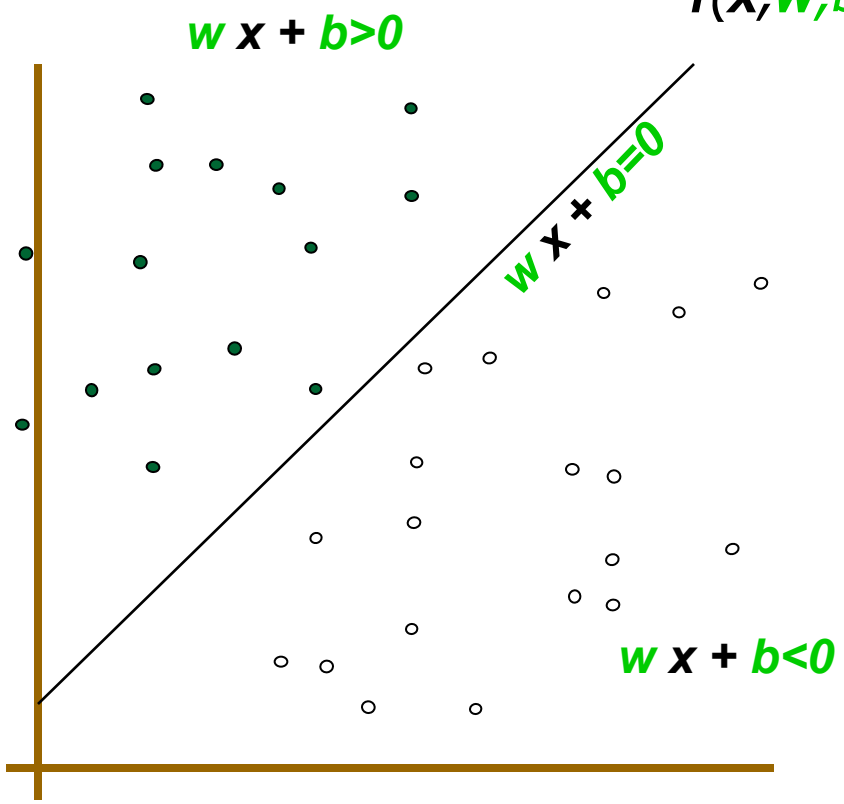
- Support vector machines were invented by V. Vapnik and his co-workers in 1970s in Russia and became known to the West in 1992.
- SVMs are linear classifiers that find a hyperplane to separate two class of data, positive and negative.
- Kernel functions are used for nonlinear separation.
- SVM not only has a rigorous theoretical foundation, but also performs classification more accurately than most other methods in applications, especially for high dimensional data.
- It is perhaps the best (non-deep) classifier for text classification.

Linear Classifiers



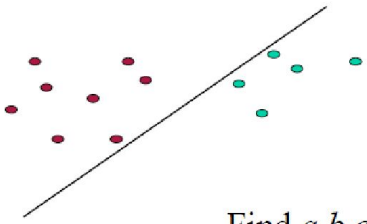
- denotes +1
- denotes -1

$f(x,w,b) = sign(w\,x + b)$



How would you classify this data?

2-D Case



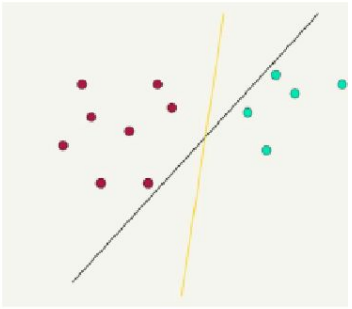
Find a,b,c, such that

$ax + by \geq c$ for red points

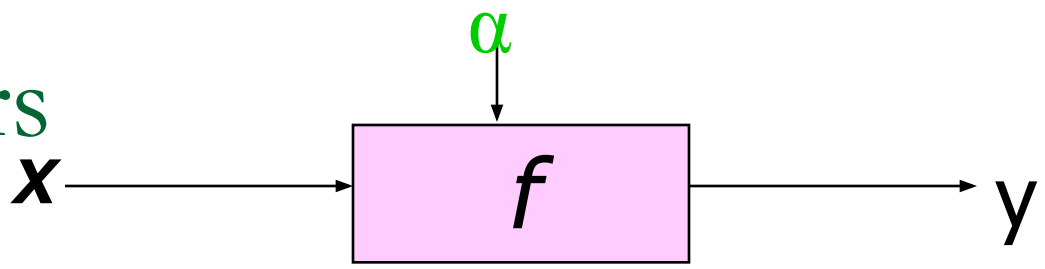
$ax + by \leq$ (or $<$) c for green points.

Which Hyperplane to pick?

- Lots of possible solutions for a,b,c.
- Some methods find a separating hyperplane, but not the optimal one (e.g., neural net)
- But: Which points should influence optimality?
  - All points?
    - Linear regression
    - Neural nets
  - Or only "difficult points" close to decision boundary
    - Support vector machines

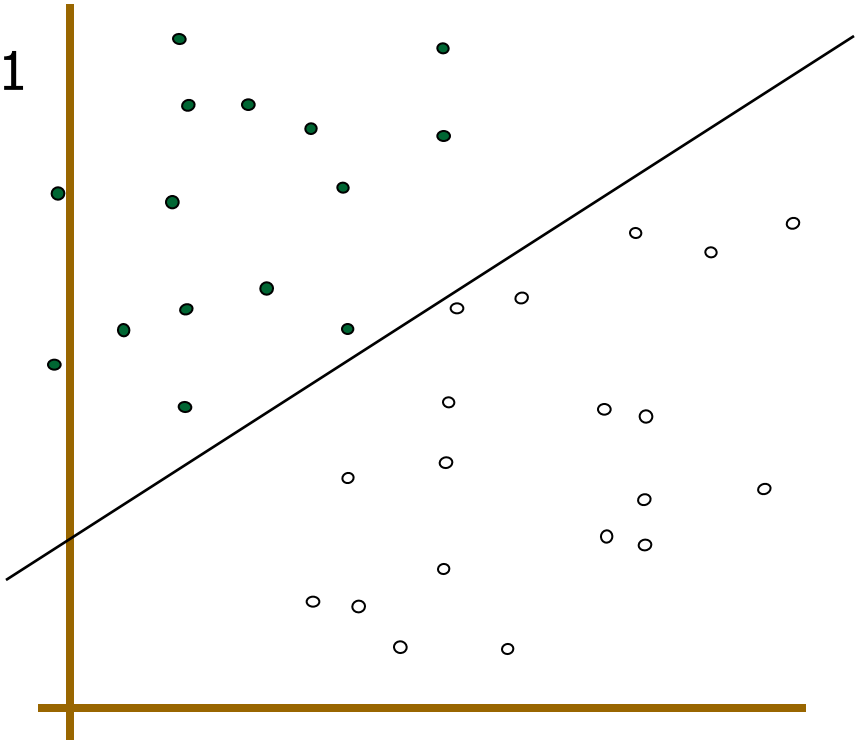


Linear Classifiers



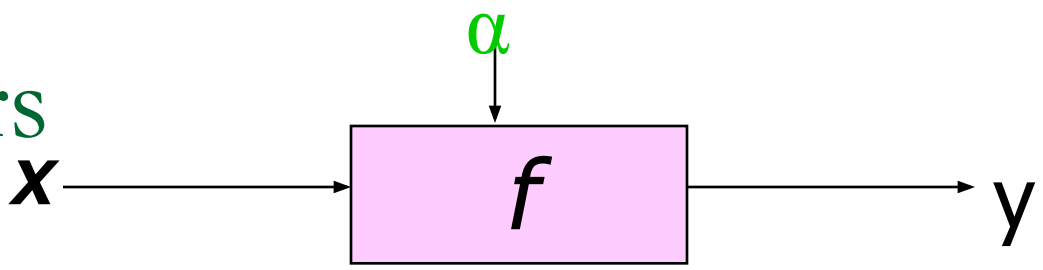
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$

- denotes +1
- denotes -1



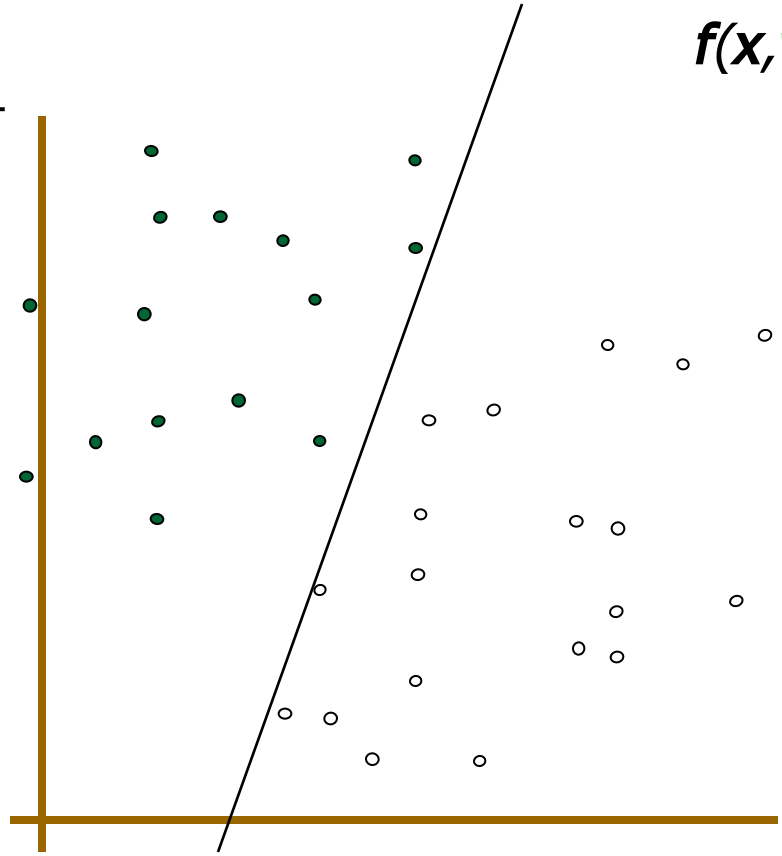
How would you classify this data?

Linear Classifiers



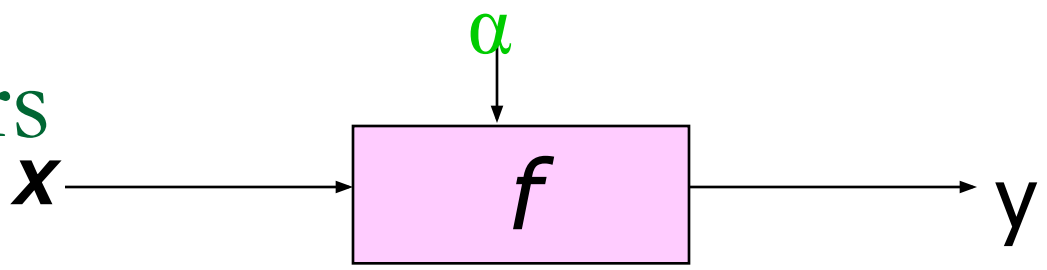
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$

- denotes +1
- denotes -1



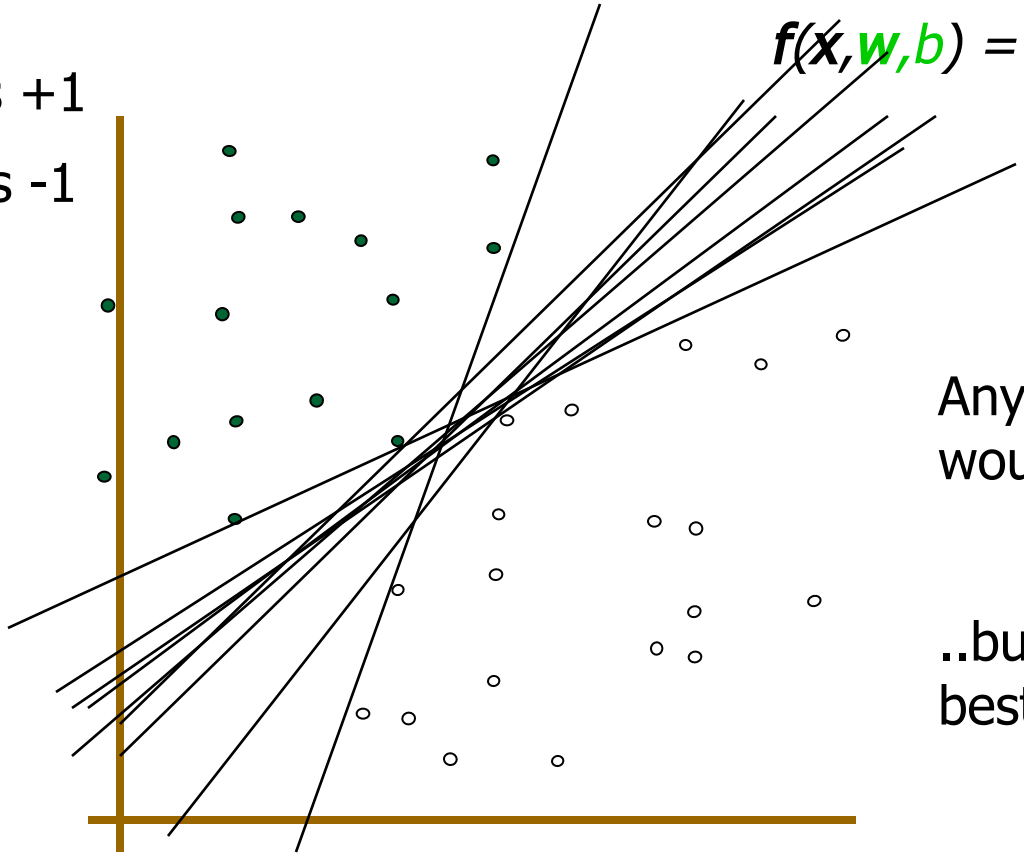
How would you classify this data?

Linear Classifiers



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$

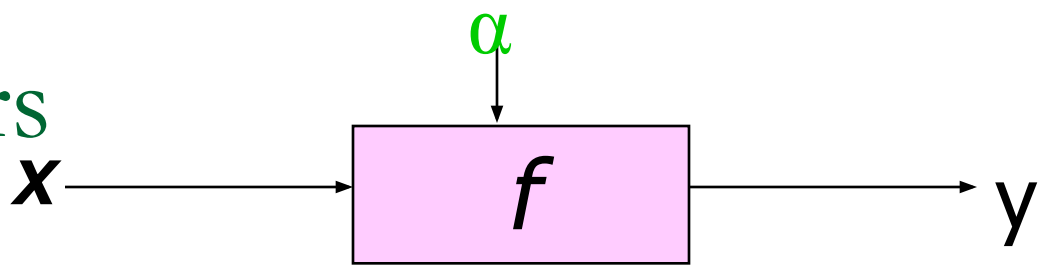
- denotes +1
- denotes -1



Any of these would be fine..

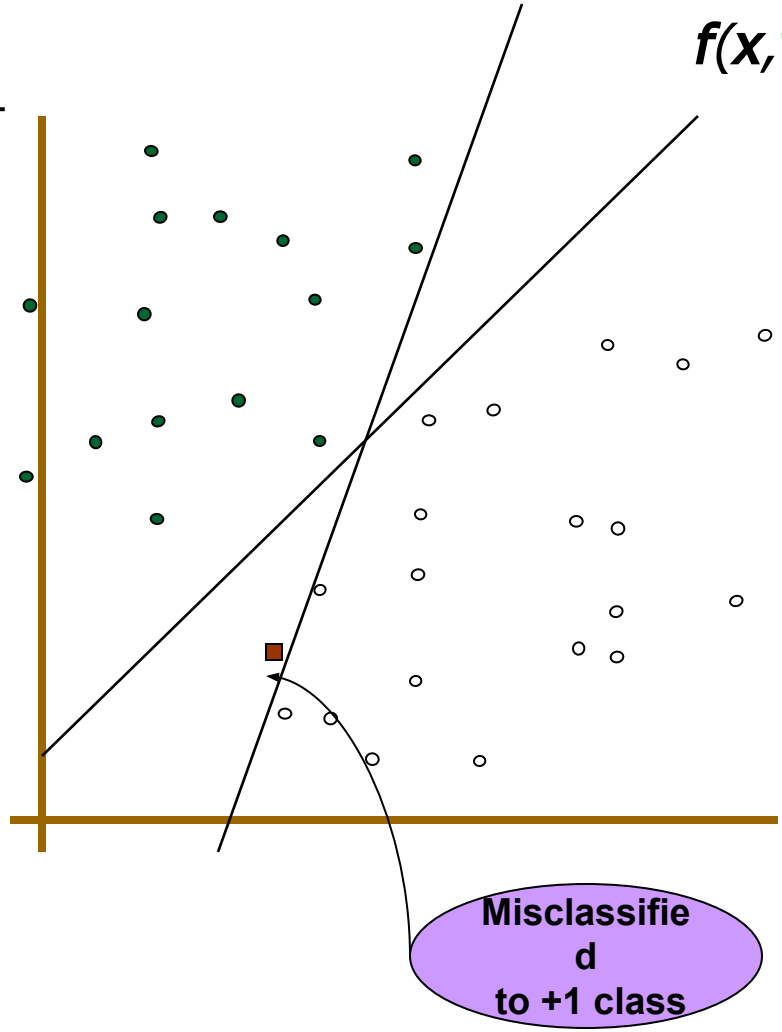
..but which is best?

Linear Classifiers



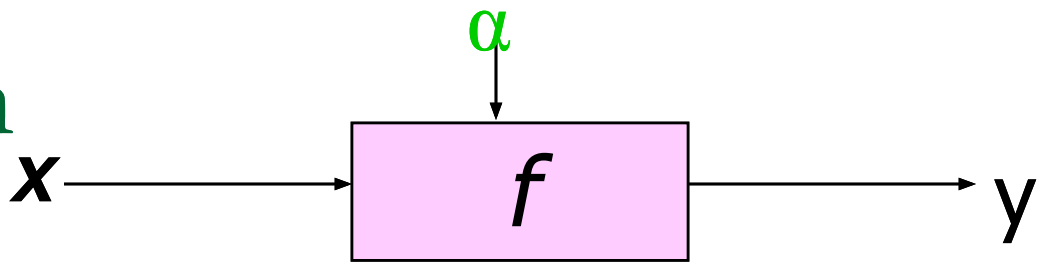
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$

- denotes +1
- denotes -1



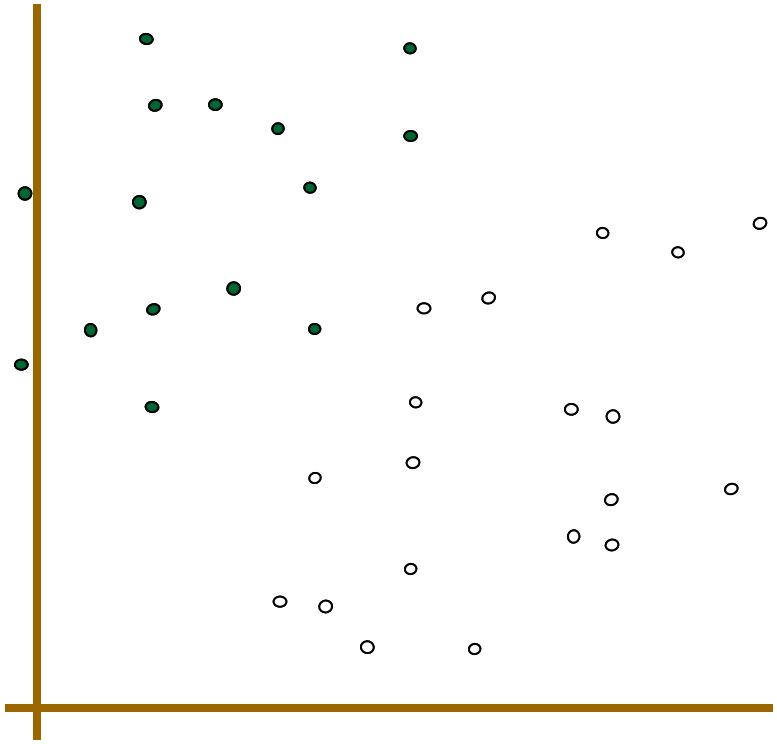
How would you classify this data?

Classifier Margin



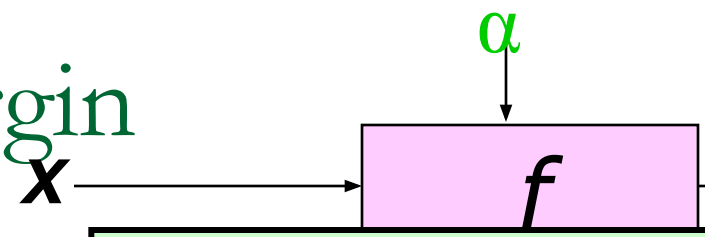
$f(x,w,b) = sign(w\,x + b)$

- denotes +1
- denotes -1



Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

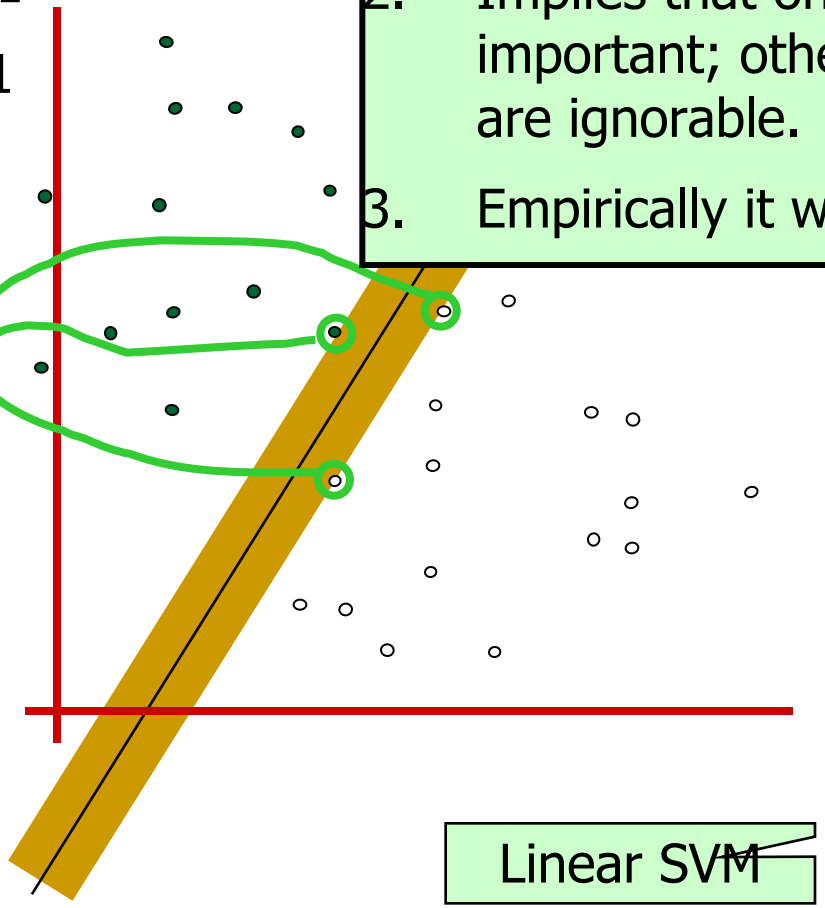
Maximum Margin



1. Maximizing the margin is good according to intuition and PAC theory
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

- denotes +1
- denotes -1

Support Vectors are those datapoints that the margin pushes up against

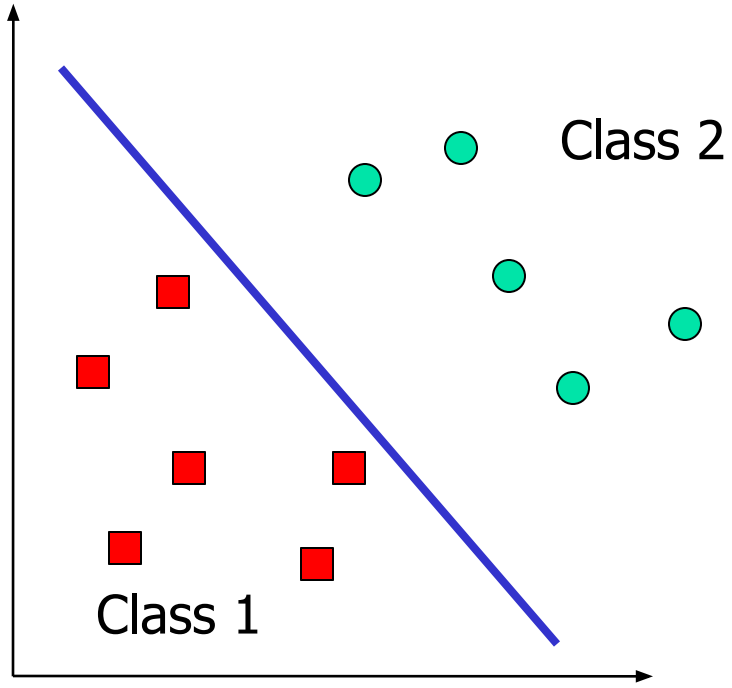


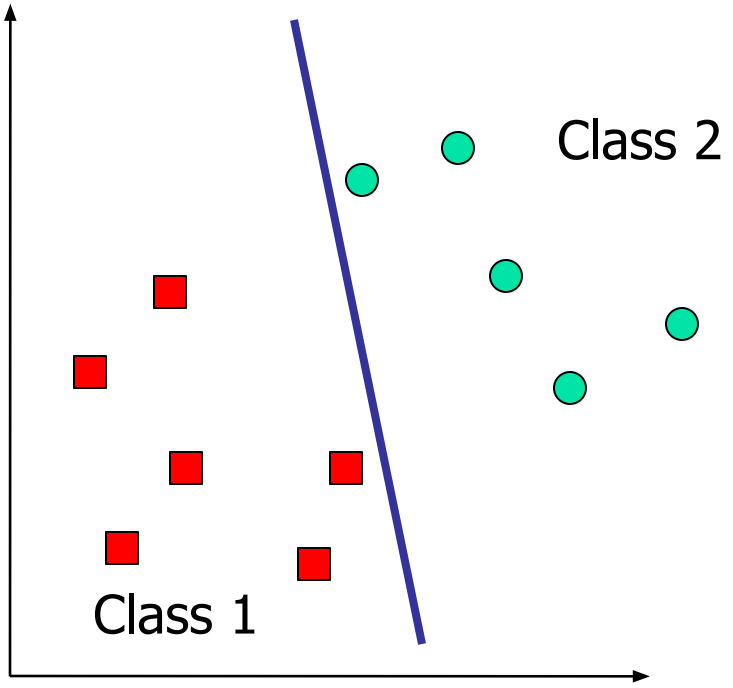
linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Example of Bad Decision Boundaries



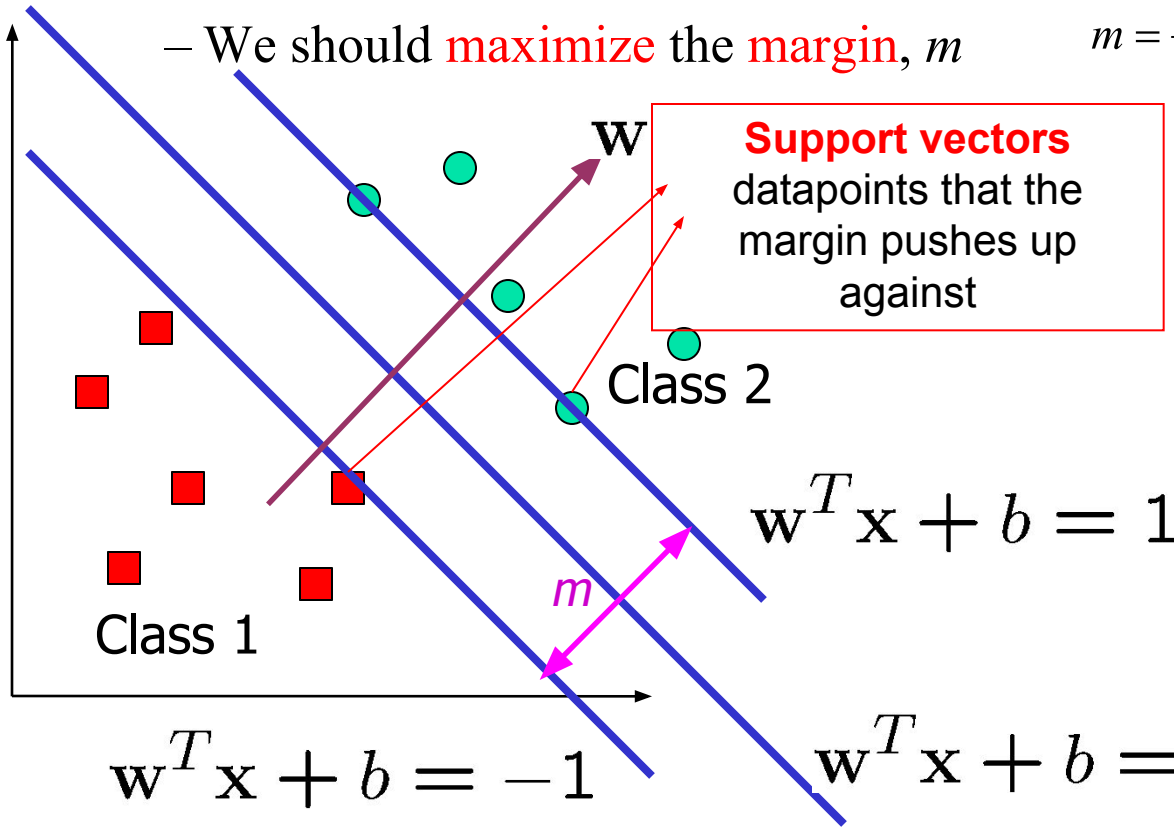


Good Decision Boundary: Margin Should Be Large

The decision boundary should be as far away from the data of both classes as possible

– We should maximize the margin, m

$$m = \frac{2}{\sqrt{w.w}} \qquad m = \frac{2}{||\mathbf{w}||}$$



$$\|\mathbf{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}.$$

The maximum margin linear classifier is the linear classifier with the maximum margin. This is the simplest kind of SVM (Called an Linear SVM)

The Optimization Problem

Let $\{x_1, ..., x_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of x_i

The decision boundary should classify all points correctly $\Rightarrow$

A constrained optimization problem

$$m = \frac{2}{||\mathbf{w}||}$$

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad \forall i$$

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

- $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

Lagrangian of Original Problem

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

subject to $1-y_i(\mathbf{w}^T\mathbf{x}_i+b) \leq 0$ for $i = 1, \ldots, n$

The Lagrangian is

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

Lagrangian multipliers

– Note that $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

Setting the gradient of $\mathcal{L}$ w.r.t. **w** and b to zero, we have

$$\mathbf{w} + \sum_{i=1}^{n} \alpha_i(-y_i)\mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \boxed{\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i}$$

$$\boxed{\sum_{i=1}^{n} \alpha_i y_i = 0}$$

$\alpha_i \geq 0$

The Dual Optimization Problem

We can transform the problem to its dual

Dot product of X

$$\max. \; W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

α's New variables
(Lagrangian multipliers)

This is a convex quadratic programming (QP) problem

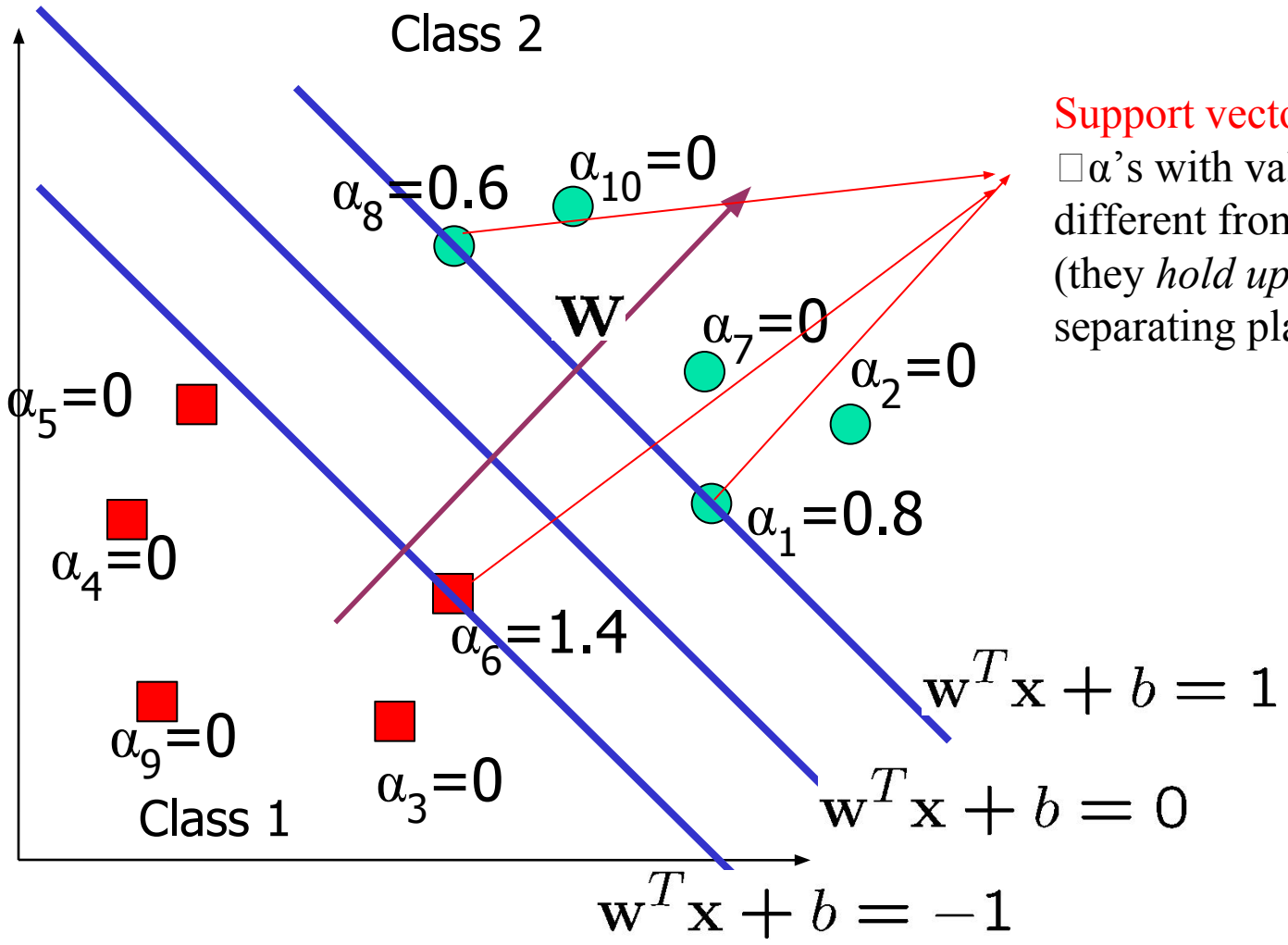
- Global maximum of α_i can always be found

well established tools for solving this optimization problem (e.g. cplex)

Note:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

A Geometrical Interpretation

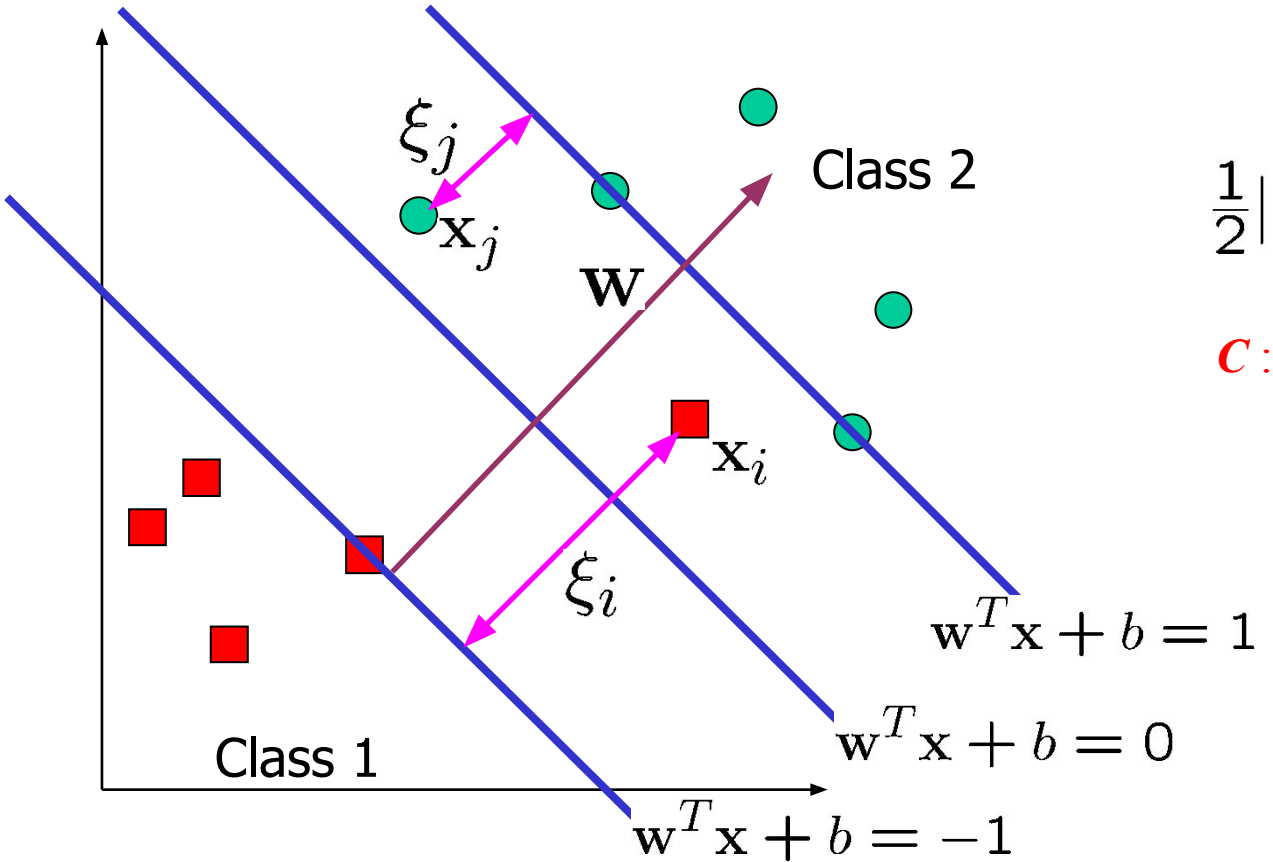


Support vectors
- α's with values different from zero (they *hold up* the separating plane)!

Non-linearly Separable Problems

We allow "error" ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T\mathbf{x}+b$

ξ_i approximates the number of misclassified samples



New objective function:

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

C : tradeoff parameter between error and margin; chosen by the user; large C means a higher penalty to errors

The Optimization Problem

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \boxed{C \geq \alpha_i \geq 0}, \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

w is also recovered as

The only difference with the linear separable case is that there is an upper bound C on α_i

Once again, a QP solver can be used to find α_i efficiently!!!

Reference

- **Support Vector Machine Classification of Microarray Gene Expression Data**, Michael P. S. Brown William Noble Grundy, David Lin, Nello Cristianini, Charles Sugnet, Manuel Ares, Jr., David Haussler
- www.cs.utexas.edu/users/mooney/cs391L/svm.**ppt**
- **Text categorization with Support Vector Machines:**
  learning with many relevant features

  T. Joachims, ECML - 98

Applications of SVMs

- Bioinformatics
- Machine Vision
- Text Categorization
- Ranking (e.g., Google searches)
- Handwritten Character Recognition
- Time series analysis

  Lots of very successful applications!!!