Lagrange duality

Another way to arrive at the KKT conditions, and one which gives us some insight on solving constrained optimization problems, is through the *Lagrange dual.* The dual is a maximization program in $\boldsymbol{\lambda}, \boldsymbol{\nu}$ — it is always concave (even when the original program is not convex), and gives us a systematic way to lower bound the optimal value.

The Lagrangian

We consider an optimization program of the form

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\ f_0(\boldsymbol{x}) \quad \begin{aligned} f_m(\boldsymbol{x}) &\leq 0, \quad m = 1,\ldots,M \\ h_p(\boldsymbol{x}) &= 0, \quad p = 1,\ldots,P. \end{aligned} \tag{1}$$

Much of what we will say below applies equally well to nonconvex programs as well as convex programs, so we will make it clear when we are taking the f_m to be convex and the h_p to be affine. We will take the domain of all of the f_m and h_p to be all of $\mathbb{R}^N$ below; this just simplifies the exposition, we can easily replace this with the intersections of the $\operatorname{dom} f_m$ and $\operatorname{dom} h_p$. We will assume that the intersection of the feasible set,

$$\mathcal{C} = \{\boldsymbol{x} \ :\ f_m(\boldsymbol{x}) \leq 0,\ h_p(\boldsymbol{x}) = 0,\ m = 1,\ldots,M,\ \ p = 1,\ldots,P\}$$

is a non-empty and a subset $\mathbb{R}^N$.

The **Lagrangian** takes the constraints in the program above and integrates them into the objective function. The Lagrangian $L : \mathbb{R}^N \times \mathbb{R}^M \times \mathbb{R}^P \to \mathbb{R}$ associated with this optimization program is

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\boldsymbol{x}) + \sum_{m=1}^{M} \lambda_m f_m(\boldsymbol{x}) + \sum_{p=1}^{P} \nu_p h_p(\boldsymbol{x})$$

The $\boldsymbol{x}$ above are referred to as **primal variables**, and the $\boldsymbol{\lambda}, \boldsymbol{\nu}$ as either **dual variables** or **Lagrange multipliers**.

The **Lagrange dual function** $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) : \mathbb{R}^M \times \mathbb{R}^P \to \mathbb{R}$ is the minimum of the Lagrangian over all values of $\boldsymbol{x}$:

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x} \in \mathbb{R}^N} \left(f_0(\boldsymbol{x}) + \sum_{m=1}^{M} \lambda_m f_m(\boldsymbol{x}) + \sum_{p=1}^{P} \nu_p h_p(\boldsymbol{x}) \right).$$

Since the dual is a pointwise infimum of a family of affine functions in $\boldsymbol{\lambda}, \boldsymbol{\nu}$, g **is concave** regardless of whether or not the f_m, h_p are convex.

The key fact about the dual function is that is it is everywhere a lower bound on the optimal value of the original program. If $p^\star$ is the optimal value for (1), then

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq p^\star, \quad \text{for all } \ \boldsymbol{\lambda} \geq \boldsymbol{0}, \ \boldsymbol{\nu} \in \mathbb{R}^P.$$

This is (almost too) easy to see. For any feasible point $\boldsymbol{x}_0$,

$$\sum_{m=1}^{M} \lambda_m f_m(\boldsymbol{x}_0) + \sum_{p=1}^{P} \nu_p h_p(\boldsymbol{x}_0) \leq 0,$$

and so

$$L(\boldsymbol{x}_0, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq f_0(\boldsymbol{x}_0), \quad \text{for all } \ \boldsymbol{\lambda} \geq \mathbf{0}, \ \boldsymbol{\nu} \in \mathbb{R}^P,$$

meaning

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x} \in \mathbb{R}^N} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq L(\boldsymbol{x}_0, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq f_0(\boldsymbol{x}_0).$$

Since this holds for all feasible $\boldsymbol{x}_0$, $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq \inf_{\boldsymbol{x} \in \mathcal{C}} f_0(\boldsymbol{x}) = p^\star$.

The (Lagrange) dual to the optimization program (1) is

$$\underset{\boldsymbol{\lambda} \in \mathbb{R}^M, \boldsymbol{\nu} \in \mathbb{R}^P}{\text{maximize}} \ g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \quad \text{subject to} \quad \boldsymbol{\lambda} \geq \mathbf{0}. \tag{2}$$

The dual optimal value $d^\star$ is

$$d^\star = \sup_{\boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{\nu}} g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \sup_{\boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{\nu}} \ \inf_{\boldsymbol{x} \in \mathbb{R}^N} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}).$$

Since $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq p^\star$, we know that

$$d^\star \leq p^\star.$$

The quantity $p^\star - d^\star$ is called the **duality gap**. If $p^\star = d^\star$, then we say that (1) and (2) exhibit **strong duality**.

Certificates of (sub)optimality

Any dual feasible[1] $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ gives us a lower bound on $p^\star$, since $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq p^\star$. If we have a primal feasible $\boldsymbol{x}$, then we know that

$$f_0(\boldsymbol{x}) - p^\star \ \leq \ f_0(\boldsymbol{x}) - g(\boldsymbol{\lambda}, \boldsymbol{\nu}).$$

We will refer to $f_0(\boldsymbol{x}) - g(\boldsymbol{\lambda}, \boldsymbol{\nu})$ as the **duality gap** for primal/dual (feasible) pair $\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}$. We know that

$$p^\star \in [g(\boldsymbol{\lambda}, \boldsymbol{\nu}), f_0(\boldsymbol{x})], \quad \text{and likewise} \quad d^\star \in [g(\boldsymbol{\lambda}, \boldsymbol{\nu}), f_0(\boldsymbol{x})].$$

If we are ever able to reduce this gap to zero, then we know that $\boldsymbol{x}$ is primal optimal, and $\boldsymbol{\lambda}, \boldsymbol{\nu}$ are dual optimal.

There are certain kinds of "primal-dual" algorithms that produce a series of (feasible) points $\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\nu}^{(k)}$ at every iteration. We can then use

$$f_0(\boldsymbol{x}^{(k)}) - g(\boldsymbol{\lambda}^{(k)}, \boldsymbol{\nu}^{(k)}) \leq \epsilon,$$

as a stopping criteria, and know that our answer would yield an objective value no further than ϵ from optimal.

Strong duality and the KKT conditions

Suppose that for a convex program, the primal optimal value $p^\star$ an the dual optimal value $d^\star$ are equal

$$p^\star = d^\star.$$

[1]We simply need $\boldsymbol{\lambda} \geq \mathbf{0}$ for $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ to be dual feasible.

If $\boldsymbol{x}^\star$ is a primal optimal point and $\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star$ is a dual optimal point, then we must have

$$\begin{aligned} f_0(\boldsymbol{x}^\star) &= g(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) \\ &= \inf_{\boldsymbol{x}\in\mathbb{R}^N} \left(f_0(\boldsymbol{x}) + \sum_{m=1}^{M} \lambda_m^\star f_m(\boldsymbol{x}) + \sum_{p=1}^{P} \nu_p^\star h_p(\boldsymbol{x}) \right) \\ &\le f_0(\boldsymbol{x}^\star) + \sum_{m=1}^{M} \lambda_m^\star f_m(\boldsymbol{x}^\star) + \sum_{p=1}^{P} \nu_p^\star h_p(\boldsymbol{x}^\star) \\ &\le f_0(\boldsymbol{x}^\star). \end{aligned}$$

The last inequality follows from the fact that $\lambda_m^\star \ge 0$ (dual feasibility), $f_m(\boldsymbol{x}^\star) \le 0$, and $h_p(\boldsymbol{x}^\star) = 0$ (primal feasibility). Since we started out and ended up with the same thing, all of the things above must be equal, and so

$$\lambda_m^\star f_m(\boldsymbol{x}^\star) = 0, \quad m = 1, \ldots, M.$$

Also, since we know $\boldsymbol{x}^\star$ is a minimizer of $L(\boldsymbol{x}, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star)$ (second equality above), which is an unconstrained convex function (with $\boldsymbol{\lambda}, \boldsymbol{\nu}$ fixed), the gradient with respect to $\boldsymbol{x}$ must be zero:

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) = \nabla f_0(\boldsymbol{x}^\star) + \sum_{m=1}^{M} \lambda_m^\star \nabla f_m(\boldsymbol{x}^\star) + \sum_{p=1}^{P} \nu_p^\star \nabla h_p(\boldsymbol{x}^\star) = \mathbf{0}.$$

Thus strong duality immediately leads to the KKT conditions holding at the solution.

Also, if you can find $\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star$ that obey the KKT conditions, not only do you know that you have a primal optimal point on your hands, but also we have strong duality (and $\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star$ are dual optimal). For if KKT holds,

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) = \mathbf{0},$$

meaning that $\boldsymbol{x}^\star$ is a minimizer of $L(\boldsymbol{x}, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star)$, i.e.

$$L(\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) \leq L(\boldsymbol{x}, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star),$$

thus

$$\begin{aligned} g(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) &= L(\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) \\ &= f_0(\boldsymbol{x}^\star) + \sum_{m=1}^{M} \lambda_m^\star f_m(\boldsymbol{x}^\star) + \sum_{p=1}^{P} \nu_p^\star h_p(\boldsymbol{x}^\star) \\ &= f_0(\boldsymbol{x}^\star), \quad \text{(by KKT)}, \end{aligned}$$

and we have strong duality.

The upshot of this is that the conditions for strong duality are essentially the same as those under which KKT is necessary.

The program (1) and its dual (2) have strong duality if the f_m are affine inequality constraints, or there is an $\boldsymbol{x} \in \mathbb{R}^N$ such that for all the f_i which are not affine we have $f_i(\boldsymbol{x}) < 0$.

Examples

1. **Inequality LP**. Calculate the dual of

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\ \langle \boldsymbol{x}, \boldsymbol{c}\rangle \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}.$$

Answer: The Lagrangian is

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{\lambda}) &= \langle \boldsymbol{x}, \boldsymbol{c}\rangle + \sum_{m=1}^{M} \lambda_m \left(\langle \boldsymbol{x}, \boldsymbol{a}_m\rangle - b_m\right) \\ &= \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{b} + \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}. \end{aligned}$$

This is a linear functional in $\boldsymbol{x}$ — it is unbounded below unless

$$\boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda} = \boldsymbol{0}.$$

Thus

$$g(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x}} \left(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{b} + \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}\right) = \begin{cases} -\langle \boldsymbol{\lambda}, \boldsymbol{b}\rangle, & \boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda} = \boldsymbol{0} \\ -\infty, & \text{otherwise.} \end{cases}$$

So the Lagrange dual program is

$$\begin{aligned} \underset{\boldsymbol{\lambda}\in\mathbb{R}^M}{\text{maximize}}\ -\langle \boldsymbol{\lambda}, \boldsymbol{b}\rangle \quad \text{subject to} \quad & \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda} = -\boldsymbol{c} \\ & \boldsymbol{\lambda} \geq \boldsymbol{0}. \end{aligned}$$

2. **Standard form LP**. Calculate the dual of

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\langle \boldsymbol{x}, \boldsymbol{c}\rangle \quad \text{subject to} \quad \begin{aligned} \boldsymbol{A}\boldsymbol{x} &= \boldsymbol{b} \\ \boldsymbol{\lambda} &\geq \boldsymbol{0}. \end{aligned}$$

Least-squares. Calculate the dual of

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\, \|\boldsymbol{x}\|_2^2 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}.$$

Check that the duality gap is zero.

Answer:

$$\underset{\boldsymbol{\nu}\in\mathbb{R}^M}{\text{maximize}}\, -\frac{1}{4}\boldsymbol{\nu}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{\nu}$$

3. **Minimum norm.** Calculate the dual of

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\, \|\boldsymbol{x}\| \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},$$

where $\|\cdot\|$ is a general valid norm.

Answer: Use $f_0(\boldsymbol{x}) = \|\boldsymbol{x}\|$ to ease notation below. We start with the Lagrangian:

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{\nu}) &= f_0(\boldsymbol{x}) + \sum_{p=1}^{P} \nu_p(\langle \boldsymbol{x}, \boldsymbol{a}_m\rangle - b_m) \\ &= f_0(\boldsymbol{x}) - \langle \boldsymbol{\nu}, \boldsymbol{b}\rangle + (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu})^{\mathrm{T}}\boldsymbol{x} \end{aligned}$$

and so

$$\begin{aligned} g(\boldsymbol{\nu}) &= -\langle \boldsymbol{\nu}, \boldsymbol{b}\rangle + \inf_{\boldsymbol{x}} \left(f_0(\boldsymbol{x}) + (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu})^{\mathrm{T}}\boldsymbol{x}\right) \\ &= -\langle \boldsymbol{\nu}, \boldsymbol{b}\rangle - \sup_{\boldsymbol{x}} \left(-f_0(\boldsymbol{x}) - (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu})^{\mathrm{T}}\boldsymbol{x}\right) \\ &= -\langle \boldsymbol{\nu}, \boldsymbol{b}\rangle - f_0^*(-\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu}), \end{aligned}$$

where f_0^* is the Fenchel dual of f_0:

$$f_0^\star(\boldsymbol{y}) = \sup_{\boldsymbol{x}}(\langle \boldsymbol{x}, \boldsymbol{y}\rangle - f_0(\boldsymbol{x})).$$

With $f_0 = \|\cdot\|$, we know already that

$$f_0^\star(\boldsymbol{y}) = \begin{cases} 0, & \|\boldsymbol{y}\|_* \leq 1, \\ \infty, & \text{otherwise} \end{cases},$$

so

$$g(\nu) = \begin{cases} -\langle \boldsymbol{\nu}, \boldsymbol{b}\rangle, & \|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu}\|_* \leq 1 \\ -\infty, & \text{otherwise} \end{cases}.$$

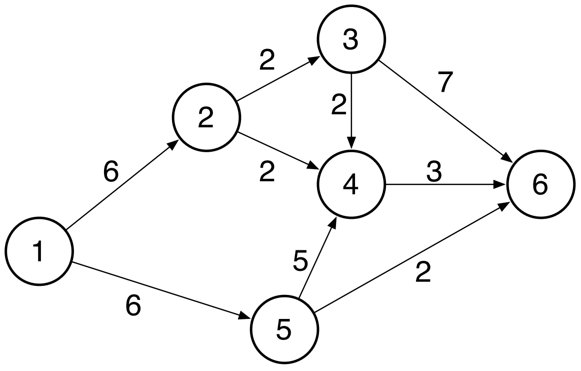
Thus the dual program is

$$\underset{\boldsymbol{\nu}\in\mathbb{R}^P}{\text{maximize}} \quad -\langle \boldsymbol{\nu}, \boldsymbol{b} \rangle \quad \text{subject to} \quad \|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\nu}\|_* \leq 1,$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$.

Max flow, Min cut

Consider the following network.



The nodes are routers, the edges are communications links; associated with each node is a capacity — node 1 can communicate to node 2 at as much as 6 Mbps, node 2 can communicate to node 4 at upto 2 Mbps, etc.

Question: Can node 1 (the source) communicate to node 6 (the sink) at 6 Mbps? 12 Mbps? What is the maximum rate?

Formally, we can model this type of problem as follows. The connections between the N nodes in the network are entries in an $N \times N$ matrix; entry (i, j) of the capacity matrix $\boldsymbol{C}$ records the capacity of link from node i to node j. If there is not a link from i to j, we set $C(i, j) = 0$. Here is the capacity matrix for the example network above:

$$\boldsymbol{C} = \begin{bmatrix} 0 & 6 & 0 & 0 & 6 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 5 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

We will solve for the $N \times N$ flow matrix $\boldsymbol{X}$:

$$X[i,j] = \text{flow from node } i \text{ to node } j.$$

Valid flows are positive and less than their respective capacities:

$$\boldsymbol{X} \geq \boldsymbol{0}, \quad \boldsymbol{X} \leq \boldsymbol{C}.$$

(The inequality above are to be understood entrywise, as the partial ordering for the non-negative orthant $\mathbb{R}_+^{N\times N}$.) To keep things simple, we will assume that there are no edges coming into the source or out of the sink. The *conservation of flow* means that the sum of all the outgoing flows at every non-source/sink node must be the same as the sum of all the incoming flows. The sum of all the outgoing flows from node i is simply the sum of all entries in row i of $\boldsymbol{X}$; the sum of all the incoming flows is the sum along column i. We will assume the source is node $i = 1$ and the sink is node $i = N$, so the conservation law becomes the $N - 2$ linear equality constraints

$$\sum_{j=2}^{N} X(i,j) = \sum_{k=1}^{N-1} X(k,i), \quad i = 2, \ldots, N-1.$$

The flow of the network, then, is simply the sum of everything coming out of the source:

$$\text{flow} = \sum_{i=2}^{N} X(1,i)$$

So, solving for the **maximum flow is a linear program**:

$$\underset{\boldsymbol{X}\in\mathbb{R}^{N\times N}}{\text{maximize}} \ \langle \boldsymbol{X}, \boldsymbol{S} \rangle \quad \text{subject to} \quad \begin{aligned} -\boldsymbol{X} &\leq \boldsymbol{0} \\ \boldsymbol{X} &\leq \boldsymbol{C} \\ \langle \boldsymbol{X}, \boldsymbol{L}_n \rangle &= \boldsymbol{0}, \quad n = 2, \ldots, N-1 \end{aligned}$$

where

$$\boldsymbol{S} = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \quad \boldsymbol{L}_2 = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & \cdots & -1 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix},$$

and similarly $\boldsymbol{L}_n$ consists of a single column (n) of ones (except for the last row) minus a single row (also n) of ones (except for the first column).

A general LP with both linear inequality and equality constraints

$$\underset{\boldsymbol{x}}{\text{minimize}} \ \langle \boldsymbol{x}, \boldsymbol{c} \rangle, \quad \text{subject to} \quad \begin{aligned} \boldsymbol{A}\boldsymbol{x} &\leq \boldsymbol{b} \\ \boldsymbol{W}\boldsymbol{x} &= \boldsymbol{y}, \end{aligned}$$

has dual

$$\underset{\boldsymbol{\lambda},\boldsymbol{\nu}}{\text{maximize}} \ -\langle \boldsymbol{\lambda}, \boldsymbol{b} \rangle - \langle \boldsymbol{\nu}, \boldsymbol{y} \rangle \quad \text{subject to} \quad \begin{aligned} \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda} + \boldsymbol{W}^{\mathrm{T}}\boldsymbol{\nu} + \boldsymbol{c} &= \boldsymbol{0} \\ \boldsymbol{\lambda} &\geq \boldsymbol{0}. \end{aligned}$$

A quick calculation shows that the dual of maxflow is then

$$\underset{\boldsymbol{\Lambda}_1,\boldsymbol{\Lambda}_2,\boldsymbol{\nu}}{\text{minimize}} \ \langle \boldsymbol{\Lambda}_1, \boldsymbol{C} \rangle \quad \text{subject to} \quad \begin{aligned} \boldsymbol{\Lambda}_1 - \boldsymbol{\Lambda}_2 + \boldsymbol{Q} - \boldsymbol{S} &= \boldsymbol{0} \\ \boldsymbol{\Lambda}_1 &\geq \boldsymbol{0} \\ \boldsymbol{\Lambda}_2 &\geq \boldsymbol{0} \end{aligned}$$

where

$$\boldsymbol{Q} = \begin{bmatrix} 0 & \nu_2 & \nu_3 & \cdots & \nu_{N-1} & 0 \\ 0 & 0 & \nu_3 - \nu_2 & \cdots & \nu_{N-1} - \nu_2 & -\nu_2 \\ 0 & \nu_2 - \nu_3 & 0 & \cdots & \nu_{N-1} - \nu_3 & -\nu_3 \\ \vdots & \cdots & & & & \vdots \\ 0 & \nu_2 - \nu_{N-1} & \nu_3 - \nu_{N-1} & \cdots & 0 & -\nu_{N-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

We see that the $\boldsymbol{\Lambda}_2$ are just slack variables, and we can re-write the program as

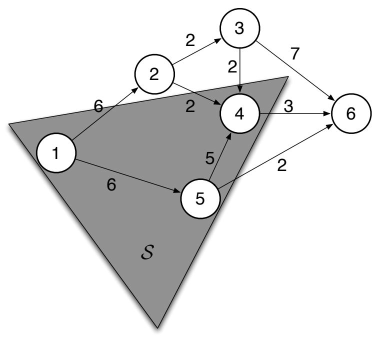
$$\underset{\boldsymbol{\Lambda},\boldsymbol{\nu}}{\text{minimize}} \ \langle \boldsymbol{\Lambda}, \boldsymbol{C} \rangle \quad \text{subject to} \quad \begin{aligned} \boldsymbol{\Lambda} + \boldsymbol{Q} &\geq \boldsymbol{S} \\ \boldsymbol{\Lambda} &\geq \boldsymbol{0}, \end{aligned}$$

or equivalently

$$\underset{\boldsymbol{\Lambda},\boldsymbol{\nu}}{\text{minimize}} \ \sum_{i,j} \lambda_{i,j} C_{i,j} \quad \text{subject to} \quad \begin{aligned} \lambda_{i,j} - \nu_j + \nu_i &\geq 0, \quad 2 \leq i,j \leq N-1 \\ \lambda_{1,j} + \nu_j &\geq 1, \quad j = 2, \ldots, N-1 \\ \lambda_{i,N} - \nu_i &\geq 0, \quad i = 2, \ldots, N-1 \\ \lambda_{1,N} &\geq 1 \\ \lambda_{i,j} &\geq 0, \quad 1 \leq i,j \leq N. \end{aligned} \tag{1}$$

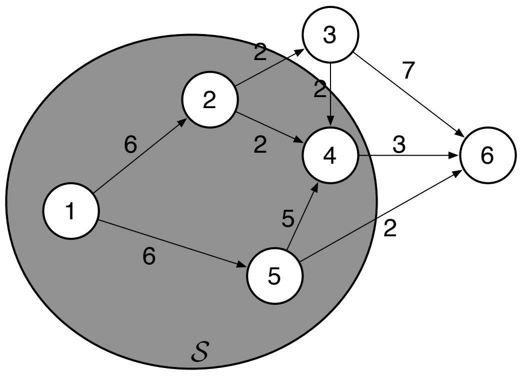
We will argue below that this dual program is equivalent to finding the **minimum cut** in the network. A cut of the network separates the vertices into two sets: one containing the source (we call this set $\mathcal{S}$, and one containing the sink. The capacity of the cut is the total value of the edges coming out of $\mathcal{S}$ — we are separating the sets by "cutting off the flow" along these edges.

For example, here we have $\mathcal{S} = \{1, 4, 5\}$:



The edges in the cut are $1 \to 2$, $4 \to 6$, and $5 \to 6$; the capacity of this cut is $6 + 3 + 2 = 11$.

In this example, we have $\mathcal{S} = \{1, 2, 4, 5\}$:



The edges in this cut are $2 \to 3$, $4 \to 6$, and $5 \to 6$. The capacity of this cut is $2 + 3 + 2 = 7$.

In general, a cut is specified by a subset of vertices containing the source but not the sink: $\mathcal{S} \subset \{1, \ldots, N\}$, $1 \in \mathcal{S}$, $N \notin \mathcal{S}$. The associated capacity is

$$\text{capacity}(\mathcal{S}) = \sum_{i \in \mathcal{S},\ j \notin \mathcal{S}} C_{i,j}$$

What is the minimum value of the smallest cut? We will argue that it is same as the optimal value of the solution $d^\star$ of the dual program in (1). First, suppose that $\mathcal{S}$ is a valid cut. From $\mathcal{S}$, we can easily find a dual feasible point that matches its capacity: for $n = 1, \ldots, N$, take

$$\nu_n = \begin{cases} 1, & n \in \mathcal{S}, \\ 0, & n \notin \mathcal{S}, \end{cases} \quad \text{and} \quad \lambda_{i,j} = \begin{cases} \max(\nu_i - \nu_j, 0), & i \neq 1,\ j \neq N, \\ 1 - \nu_j, & i = 1, \\ \nu_i, & j = N \end{cases}.$$

Notice that these choices obey the constraints in the dual, and that $\lambda_{i,j}$ will be 1 if $i \to j$ is cut, and 0 otherwise, so

$$\text{capactity}(\mathcal{S}) = \sum_{i,j} \lambda_{i,j} C_{i,j}.$$

Every cut is feasible, so

$$d^\star \leq \text{MINCUT}.$$

Now we show that for every solution $\boldsymbol{\nu}^\star, \boldsymbol{\lambda}^\star$ of the dual, there is a cut that has a capacity at most $d^\star$. The argument for this is nifty: we generate a cut *at random*, and then show that the expected value of the capacity of the cut is less than $d^\star$ — this means there must be at least one with a capacity of $d^\star$ or less.

Let Z be a uniform random variable on $[0, 1]$. Along with $\boldsymbol{\lambda}^\star, \nu_2^\star, \ldots, \nu_{N-1}^\star$ generated by solving (1), take $\nu_1 = 1$ and $\nu_N = 0$. Create a cut $\mathcal{S}$ with the rule:

$$\text{if } \nu_n^\star > Z, \text{ then take } n \in \mathcal{S}.$$

The probability that a particular edge $i \to j$ is in this cut is

$$\begin{aligned}
\mathrm{P}\left(i \in \mathcal{S}, j \notin \mathcal{S}\right) &= \mathrm{P}\left(\nu_j^\star \leq Z \leq \nu_i^\star\right) \\
&\leq \begin{cases} \max(\nu_i^\star - \nu_j^\star, 0), & 2 \leq i, j \leq N-1, \\ 1 - \nu_j^\star, & i = 1;\ j = 2, \ldots, N-1, \\ \nu_i^\star, & i = 2, \ldots, N-1;\ j = N \\ 1, & i = 1;\ j = N. \end{cases} \\
&\leq \lambda_{i,j}^\star,
\end{aligned}$$

where the last inequality follows simply from the constraints in the dual program (1). This cut is random, so its capacity is a random variables, and its expectation is

$$\begin{aligned}
\mathrm{E}[\mathrm{capacity}(\mathcal{S})] &= \sum_{i,j} C_{i,j}\, \mathrm{P}\left(i \in \mathcal{S}, j \notin \mathcal{S}\right) \\
&\leq \sum_{i,j} C_{i,j} \lambda_{i,j}^\star \\
&= d^\star.
\end{aligned}$$

Thus there must be a cut whose capacity is at most $d^\star$. This establishes that

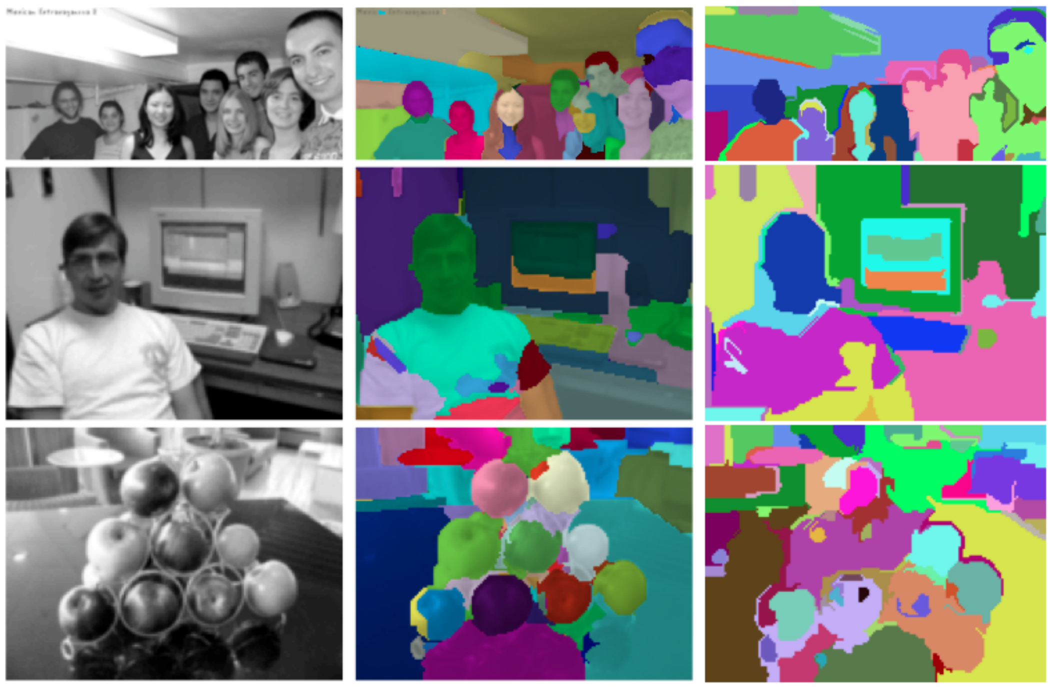
$$\mathrm{MINCUT} \leq d^\star.$$

Combining these two facts of course means that

$$d^\star = \mathrm{MINCUT} = \mathrm{MAXFLOW} = p^\star,$$

where $p^\star$ is the solution of the primal, and equality follows from strong duality for linear programming.

The minimum cut problem is interesting by itself. Among other things, it can be used to perform image segmentation:

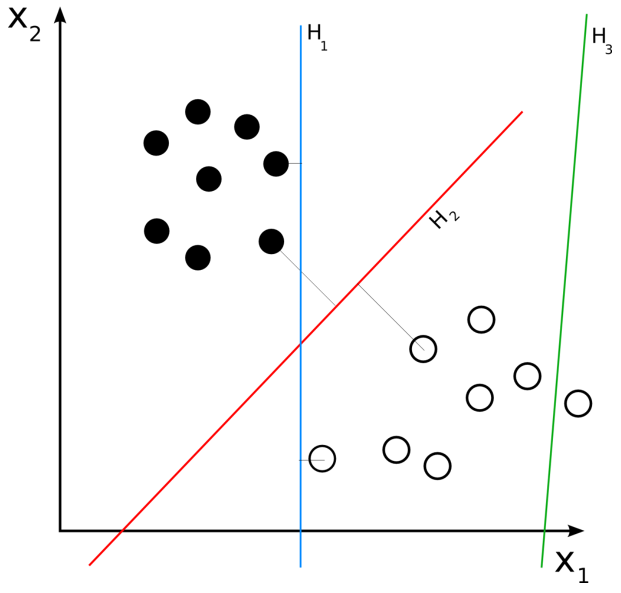


(From F. Estrada et al. (2004), "Spectral embedding and min cut for image segmentation")

You set this problem up by connecting each pixel to a foreground "source" (with some capacity that would represent the foreground value) and to a background "sink" (with some capacity that would represent the background value), and penalizing if adjacent pixels get assigned to different modes. (See the reference above for all of the details.)

Support vector machines

Consider the following fundamental binary classification problem. We are given points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M \in \mathbb{R}^N$ with labels $y_1, \ldots, y_M$, where $y_m \in \{-1, +1\}$. We would like to find a hyperplane (i.e. affine functional) which *separates* the points[1]:



$\mathcal{H}_1$ and $\mathcal{H}_2$ above separate the points in $\mathbb{R}^2$, but $\mathcal{H}_3$ does not. To choose among the hyperplanes which separate the points, we will take the one with maximum margin (maximize the distance to the closest point in either class).

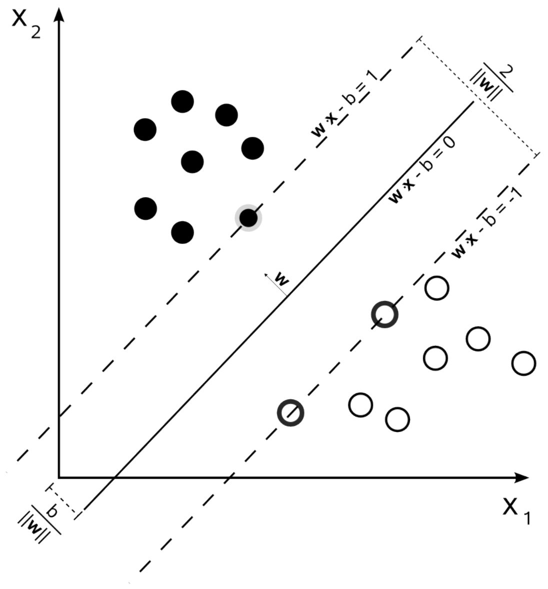
To restate this, we want to find a $\boldsymbol{w} \in \mathbb{R}^N$ and $b \in \mathbb{R}$ such that

$$\begin{aligned} \langle \boldsymbol{x}_m, \boldsymbol{w} \rangle - b &\geq 1, \quad \text{when } y_m = 1, \\ \langle \boldsymbol{x}_m, \boldsymbol{w} \rangle - b &\leq -1, \quad \text{when } y_m = -1. \end{aligned}$$

[1]From Wikipedia: "Svm separating hyperplanes (SVG)" by User:ZackWeinberg, based on PNG version by User:Cyc.

Of course, it is possible that no separating hyperplane exists; in this case, there will be no feasible points in the program above. It is straightforward, though, to modify this discussion to allow "mislabeled" points.

In the formulation above, the distance between the two (parallel) hyperplanes[2] is $2/\|\boldsymbol{w}\|_2$:



Thus maximizing this distance is the same as minimizing $\|\boldsymbol{w}\|_2$.

We have the program

$$\underset{\boldsymbol{w}\in\mathbb{R}^N,\ b\in\mathbb{R}}{\text{minimize}}\ \frac{1}{2}\|\boldsymbol{w}\|_2^2 \quad \text{subject to} \quad y_m(b-\langle \boldsymbol{x}_m, \boldsymbol{w}\rangle)+1 \le 0,\ m = 1,\ldots,M.$$

This is a linearly constrained quadratic program, and is clearly con-

[2]From Wikipedia: "Svm max sep hyperplane with margin" by Cyc - Own work. Licensed under Public Domain via Wikimedia Commons.

vex. The Lagrangian is

$$\begin{aligned} L(\boldsymbol{w}, b, \boldsymbol{\lambda}) &= \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \sum_{m=1}^{M} \lambda_m \left[y_m(b - \langle \boldsymbol{x}_m, \boldsymbol{w}\rangle) + 1\right] \\ &= \frac{1}{2}\|\boldsymbol{w}\|_2^2 + b\,\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{1}, \end{aligned}$$

where $\boldsymbol{X}$ is the $N \times M$ matrix

$$\boldsymbol{X} = \begin{bmatrix} y_1\boldsymbol{x}_1 & y_2\boldsymbol{x}_2 & \cdots & y_M\boldsymbol{x}_M \end{bmatrix}.$$

The dual function is

$$g(\boldsymbol{\lambda}) = \inf_{\boldsymbol{w}, b} \left(\frac{1}{2}\|\boldsymbol{w}\|_2^2 + b\,\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{1}\right).$$

Since b is unconstrained above, we see that the presence of $b\,\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{y}$ means that the dual will be $-\infty$ unless $\langle \boldsymbol{\lambda}, \boldsymbol{y}\rangle = 0$. Minimizing over $\boldsymbol{w}$, we need the gradient equal to zero,

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}, b, \boldsymbol{\lambda}) = \mathbf{0}, \quad \Rightarrow \quad \boldsymbol{w} - \boldsymbol{X}\boldsymbol{\lambda} = \mathbf{0}.$$

This means that we must have $\boldsymbol{w} = \boldsymbol{X}\boldsymbol{\lambda}$, which itself is a very handy fact as it gives us a direct passage from the dual solution to the primal solution. With these substitutions, the dual function is

$$g(\boldsymbol{\lambda}) = \begin{cases} \frac{1}{2}\|\boldsymbol{X}\boldsymbol{\lambda}\|_2^2 - \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\lambda} + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{1}, & \langle \boldsymbol{\lambda}, \boldsymbol{y}\rangle = 0, \\ -\infty, & \text{otherwise.} \end{cases}$$

The dual SVM program is then

$$\begin{aligned} \underset{\boldsymbol{\lambda}}{\text{maximize}} \quad -\frac{1}{2}\|\boldsymbol{X}\boldsymbol{\lambda}\|_2^2 + \sum_{m=1}^{M} \lambda_m \quad \text{subject to} \quad & \langle \boldsymbol{\lambda}, \boldsymbol{y}\rangle = 0 \\ & \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned}$$

Given the solution $\boldsymbol{\lambda}^\star$ above, we can take $\boldsymbol{w}^\star = \boldsymbol{X}\boldsymbol{\lambda}^\star$, and the classifier is

$$\begin{aligned} f(\boldsymbol{x}) &= \langle \boldsymbol{x}, \boldsymbol{w}^\star \rangle - b^\star \\ &= \langle \boldsymbol{x}, \boldsymbol{X}\boldsymbol{\lambda}^\star \rangle - b^\star \\ &= \sum_{m=1}^{M} \lambda_m^\star y_m \langle \boldsymbol{x}, \boldsymbol{x}_m \rangle - b^\star. \end{aligned}$$

Notice that the data $\boldsymbol{x}_m$ appear only as linear functionals (i.e. inner products with) $\boldsymbol{x}$.

The key realization is that the for the dual program, the functional depends on the data $\boldsymbol{x}_m$ only through inner products, as

$$\|\boldsymbol{X}\boldsymbol{\lambda}\|_2^2 = \sum_{\ell=1}^{M} \sum_{m=1}^{M} y_\ell y_m \langle \boldsymbol{x}_\ell, \boldsymbol{x}_m \rangle.$$

This means we can replace $\langle \boldsymbol{x}_\ell, \boldsymbol{x}_m \rangle$ with any "positive kernel function" $K(\boldsymbol{x}_\ell, \boldsymbol{x}_m) : \mathbb{R}^N \otimes \mathbb{R}^N \to \mathbb{R}$ — a positive kernel just means that the $M \times M$ matrix $K(\boldsymbol{x}_\ell, \boldsymbol{x}_m)$ is in S_+^M for all choices of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$.

For example: you might take

$$K(\boldsymbol{x}_\ell, \boldsymbol{x}_m) = (1 + \langle \boldsymbol{x}_\ell, \boldsymbol{x}_m \rangle)^2 = 1 + 2\langle \boldsymbol{x}_\ell, \boldsymbol{x}_m \rangle + \langle \boldsymbol{x}_\ell, \boldsymbol{x}_m \rangle^2.$$

This means we have replaced the inner product of two vectors with the inner product between two vectors which have been mapped into a higher dimensional space:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_N \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_N^2 \\ \sqrt{2}x_1x_2 \\ \vdots \\ \sqrt{2}x_{N-1}x_N \end{bmatrix}$$

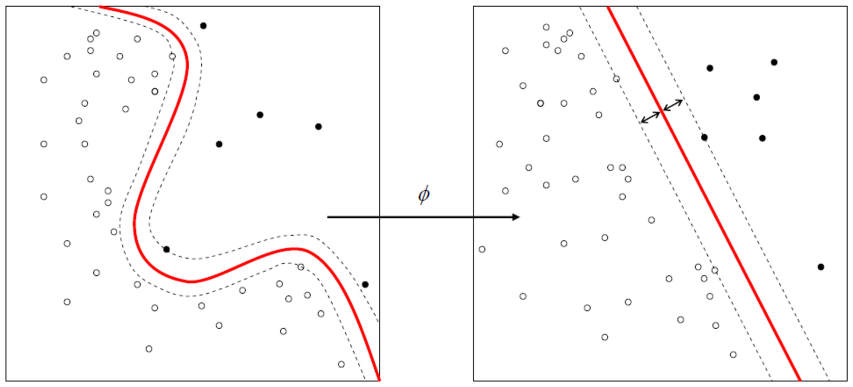
A set of linear constraints on the coordinates on the right, then, corresponds to a second order curve constraint (parabola, ellipse, hyperbola) on the coordinate on the left.

Many kernels are possible. The advantage is that to train and use the classifier, you never have to explicitly move to the higher dimensional space — you just need to be able to compute $K(\boldsymbol{x}_\ell, \boldsymbol{x}_m)$ for any pair of inputs in $\mathbb{R}^N$. A popular choice of kernel is

$$K(\boldsymbol{x}_\ell, \boldsymbol{x}_m) = \exp\left(-\gamma\|\boldsymbol{x}_\ell - \boldsymbol{x}_m\|_2^2\right).$$

This is a perfectly valid positive kernel, and it is straightforward to compute it for any pair of inputs. But it corresponds to mapping the $\boldsymbol{x}_m$ into an infinite dimensional space, then finding a hyperplane.

Here is an example from Wikipedia[3]:



[3] "Kernel Machine" by Alisneaky — Own work. Licensed under CC0 via Wikimedia Commons.