Derivations for Linear Algebra and Optimization

John Duchi

Contents

1 Householder Transformations

Much of this section was copied and paraphrased from Heath's *Scientific Computing*. Anyways.

Suppose we are looking for an orthogonal transformation that annihilates desired components of a given vector. Recall that a square real matrix Q is said to be *orthogonal* if its columns are orthonormal, that is, that $Q^TQ = I$. Orthogonal transformations are nice because they preserve Euclidean norms of any vectors $\boldsymbol{v}$, as follows:

$$\|Q\boldsymbol{v}\|_2^2 = (Q\boldsymbol{v})^TQ\boldsymbol{v} = \boldsymbol{v}^TQ^TQ\boldsymbol{v} = \boldsymbol{v}^T\boldsymbol{v} = \|\boldsymbol{v}\|_2^2$$

These sorts of transformations are nice, because when applied to any linear system, they maintain Euclidean norms and won't penalize the difficulty of solving the problem numerically.

Now, we are really looking for an orthogonal transformation of a vector that annihilates its components while maintaining the two-norm (or Euclidean norm) $\|\cdot\|_2$. We can accomplish this with a *Householder transformation*, which is a matrix of the form

$$H = I - 2\frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{v}}.$$

Obviously, $\boldsymbol{v}$ is a non-zero vector. From our definition, it is easy to see that $H = H^T$, and a little math gives us that

$$HH = (I - 2\frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{v}})(I - 2\frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{v}}) = I - 4\frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{v}} + 4\frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{v}} \cdot \frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{v}} = I$$

To derive the full Householder, we first consider the problem of taking a vector $\boldsymbol{a}$ and annihilating all of its components but the first, giving us a vector $\alpha\boldsymbol{e}_1$, in such a way that $\|\boldsymbol{a}\|_2 = \alpha$. Basically, we would like to have

$$H\boldsymbol{a} = \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \alpha \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \|\boldsymbol{a}\|_2\boldsymbol{e}_1$$

Using the original formula for H, we have

$$\alpha\boldsymbol{e}_1 = H\boldsymbol{a} = \boldsymbol{a} - 2\frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{v}}\boldsymbol{a}$$

so that

$$\boldsymbol{v} = (\boldsymbol{a} - \alpha\boldsymbol{e}_1)\frac{\boldsymbol{v}^T\boldsymbol{v}}{2\boldsymbol{v}^T\boldsymbol{a}} = (\boldsymbol{a} \pm \|\boldsymbol{a}\|_2\boldsymbol{e}_1) \cdot C(\boldsymbol{v}, \boldsymbol{a})$$

Because the constant C term will be factored out by division when we determine the actual transformation H, we can eliminate it, giving us that

$$\boldsymbol{v} = \boldsymbol{a} - \alpha\boldsymbol{e}_1$$

In general, to avoid cancellation (whatever that means), we choose $\alpha = -\text{sign}(a_1)\|\boldsymbol{a}\|_2$.

In the more general case, when actually working with a system that we would like to transform orthogonally into an upper-triangular system (for easier solving), we would like to take an m-vector $\boldsymbol{a}$ and annihilate its last $m - k$ components. That is, we would like to take a vector

$$\boldsymbol{a} = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \end{bmatrix}$$

where $\boldsymbol{a}_1$ is a $k-1$-vector and $\boldsymbol{a}_2$ is an $m-k+1$-vector, and transform it into a vector which contains $\boldsymbol{a}_1$ as its first $k-1$ elements, as its k^{th} element we would like $\|\boldsymbol{a}\|_2$, and the remaining $m-k$ elements we would like to be zeros. Here is the derivation of the vector $\boldsymbol{v}$ to be used in the Householder transformation for this case:

$$H\boldsymbol{a} = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{0} \end{bmatrix} + \alpha\boldsymbol{e}_k = \left(I - 2\frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{v}}\right) \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \end{bmatrix} - 2\frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{v}}\boldsymbol{a}$$

Subtracting $\boldsymbol{a}$ from both sides and multiplying by negative one, we have

$$\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{a}_2 \end{bmatrix} - \alpha\boldsymbol{e}_k = 2\boldsymbol{v}\frac{\boldsymbol{v}^T\boldsymbol{a}}{\boldsymbol{v}^T\boldsymbol{v}} = \boldsymbol{v} \cdot C(\boldsymbol{v}, \boldsymbol{a})$$

and again we can eliminate the constant term to give us that we should set $\boldsymbol{v}$ to be

$$\boldsymbol{v} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{a}_2 \end{bmatrix} - \|\boldsymbol{a}_2\|_2\boldsymbol{e}_k$$

From the above discussion, it is fairly easy to see that given an $m \times n$ matrix A, if we apply n Householder transformations to eliminate the below diagonal elements for A, we will get a matrix R such that

$$H_nH_{n-1}\cdots H_1A = \begin{bmatrix} R \\ 0 \end{bmatrix}.$$

For further explication and use of the Householder, see Heath.

2 Convex Sets and Local/Global Minima

The definition of a *convex set* is a set $S \subseteq \mathbb{R}^n$ which contains the line segment between any two of its points. More specifically, if S is convex, then for any $\boldsymbol{x}, \boldsymbol{y} \in S$, $\alpha \in [0, 1]$,

$$\alpha\boldsymbol{x} + (1 - \alpha)\boldsymbol{y} \in S.$$

A function $f : S \subseteq \mathbb{R}^n \to \mathbb{R}$ is *convex* on a convex set S if its value along any line segment in S lies on or below the chord connecting the function values at the endpoints of the segment. That is,

$$f(\alpha\boldsymbol{x} + (1 - \alpha\boldsymbol{y})) \leq \alpha f(\boldsymbol{x}) + (1 - \alpha)f(\boldsymbol{y})$$

2.1 Local Minima as Global Minima

One really nice property of convex sets is that any local minimum of a convex function f on a convex set S is a global minimum of f on S. To see this, consider the definition of a local minimum:

$$f(\boldsymbol{x}) < f(\boldsymbol{x} + \boldsymbol{\varepsilon})$$

for some small $\boldsymbol{\varepsilon}$, which states that $\boldsymbol{x} \in S$ is the smallest value f takes on in some ball of radius $\varepsilon = \|\boldsymbol{\varepsilon}\|$ on S. Now, we consider some other point $\boldsymbol{y} \in S$. Assume, for the sake of contradiction, that $f(\boldsymbol{y}) < f(\boldsymbol{x})$, which means that $\boldsymbol{x}$ is not the global minimum of f on S. By the definition of convexity, the chord connecting $\boldsymbol{x}$ and $\boldsymbol{y}$ lies completely in S. Thus, we can assert that there is a point on the chord $\alpha\boldsymbol{x} + (1 - \alpha)\boldsymbol{y}$ that lies inside the ball, that is, there is some α_r such that

$$\|\alpha_r\boldsymbol{x} + (1 - \alpha_r)\boldsymbol{y} - \boldsymbol{x}\| < \varepsilon$$

Consider some small $\alpha_r \in (0, 1)$, and let $\alpha_r\boldsymbol{x} + (1 - \alpha_r)\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{\varepsilon}_r$, which is a point inside the ball $\boldsymbol{x} + \boldsymbol{\varepsilon}$. Now, consider that

$$\begin{aligned} f(\boldsymbol{y}) &< f(\alpha\boldsymbol{x} + (1 - \alpha)\boldsymbol{y}) && (1)\\ &\leq \alpha f(\boldsymbol{x}) + (1 - \alpha)f(\boldsymbol{y}) && (2)\\ &\leq \alpha f(\boldsymbol{x}) + (1 - \alpha)f(\boldsymbol{x}) = f(\boldsymbol{x}) \end{aligned}$$

where equation (1) follows because $\boldsymbol{y}$ is, by our assumption, the global minimum of f on S, and equation (2) follows because f is convex on S. Note that the above implies that for any $\alpha \in [0, 1)$, we have that $\alpha f(\boldsymbol{x}) + (1 - \alpha)f(\boldsymbol{y}) < f(\boldsymbol{x})$, specifically, for $\alpha = \alpha_r$. We saw above, though, that for α_r, $\boldsymbol{x} + \boldsymbol{\varepsilon}_r$ was inside the ball given by $\boldsymbol{x} + \boldsymbol{\varepsilon}$, so that $f(\boldsymbol{x} + \boldsymbol{\varepsilon}_r) > f(\boldsymbol{x})$. This is a contradiction, however, because we saw from equation (2) that $\alpha f(\boldsymbol{x}) + (1 - \alpha)f(\boldsymbol{y}) \leq f(\boldsymbol{x})$. Our assumption that $f(\boldsymbol{y}) < f(\boldsymbol{x})$ must thus be false, and we have that if $\boldsymbol{x}$ is a local minimum of f on S, then $\boldsymbol{x}$ is a global minimum of f on S.

2.2 Uniqueness of Global Minima

If we have a function f which is strictly convex on a convex set S, that is, $f(\alpha\boldsymbol{x} + (1 - \alpha)\boldsymbol{y}) < \alpha f(\boldsymbol{x}) + (1 - \alpha)f(\boldsymbol{y})$ for $\alpha \in (0, 1)$, then a local minimum of f is the unique global minimum of f on S. To see this, suppose we have two distinct points, $\boldsymbol{x}$ and $\boldsymbol{y}$, which are local minima of f on S. By the above result, if $\boldsymbol{x}$ and $\boldsymbol{y}$ existed, we know that they are both global minima, so that $f(\boldsymbol{x}) = f(\boldsymbol{y})$. This means, however, that because f is strictly convex,

$$\begin{aligned} f(\alpha\boldsymbol{x} + (1 - \alpha)\boldsymbol{y}) &< \alpha f(\boldsymbol{x}) + (1 - \alpha)f(\boldsymbol{y})\\ &= \alpha f(\boldsymbol{x}) + (1 - \alpha)f(\boldsymbol{x}) = f(\boldsymbol{x}) \end{aligned}$$

This is a contradiction because $\boldsymbol{x}$ is a global minima of f on S, so our assumption that $\boldsymbol{y}$ also existed must be false. Thus, if a function f is strictly convex on a convex set S, any local minimum of f is the unique global minimum of f on S.

3 Second Order Convexity Conditions

This is a problem from Boyd's book on convex optimization. Suppose we have a function $f : \mathbb{R} \to \mathbb{R}$, which is convex, and $a, b \in \operatorname{dom} f$, with $a < b$. First, we consider $x \in [a, b]$. If $\theta \in [0, 1]$, we know that $f(\theta x + (1-\theta x)) \leq \theta f(x) + (1-\theta) f(x)$. Because this is the case, and because $x = \theta a + (1-\theta)b$ for some θ for any $x \in [a, b]$, it is easy to see that $\theta = \frac{x-b}{a-b} = \frac{b-x}{b-a}$. Thus,

$$\begin{aligned} f(x) &= f(\theta a + (1-\theta)b) \\ &\leq \theta f(a) + (1-\theta) f(b) \\ &= \frac{b-x}{b-a} f(a) + \frac{x-a}{b-a} f(b) \end{aligned}$$

Using this, we can prove some more interesting things. First, we have

$$\begin{aligned} f(x) + \frac{x-b}{b-a} f(a) &\leq \frac{x-a}{b-a} f(b) \\ (b-a) f(x) - b f(a) &\leq (x-a) f(b) - x f(a) \\ (b-a) f(x) - b f(a) + a f(a) &\leq (x-a) f(b) - x f(a) + a f(a) \\ [f(x) - f(a)] \cdot (b-a) &\leq (x-a)[f(b) - f(a)] \\ \frac{f(x) - f(a)}{x-a} &\leq \frac{f(b) - f(a)}{b-a} \end{aligned}$$

Similarly,

$$\begin{aligned} (x-b) f(a) &\leq (x-a) f(b) - f(x)(b-a) \\ (b-x) f(b) + (x-b) f(a) &\leq (b-a) f(b) - (b-a) f(x) \\ [f(b) - f(a)]\,(b-x) &\leq (b-a)[f(b) - f(x)] \\ \frac{f(b) - f(a)}{b-a} &\leq \frac{f(b) - f(x)}{b-x} \end{aligned}$$

If we suppose that f is differentiable, taking each side of the two inequalities in turn, we have (by L'Hôpital's rule) that

$$\lim_{x \to a} \frac{f(x) - f(a)}{x-a} = \lim_{x \to a} \frac{f'(x)}{1} = f'(a) \leq \frac{f(b) - f(a)}{b-a} \leq \lim_{x \to b} \frac{f(b) - f(x)}{b-x} = f'(b).$$

While we used the above inequalities to prove this, it also easily follows from the gradient of a convex function f being a global underestimator of f (that is $f(x) \geq f(y) + f'(y)(x-y)$).

Lastly, we assume that f is twice differentiable. Using the above result, that

$$f'(a) \leq \frac{f(b) - f(a)}{b-a} \leq f'(b)$$

we see that $f'(b) - f'(a) \geq 0$, so that $\frac{f'(y) - f'(x)}{y-x} \geq 0$ as long as $y > x$, and we then have

$$\lim_{y \to x} \frac{f'(y) - f'(x)}{y-x} = \lim_{y \to x} \frac{f''(y) - \frac{d}{dy} f'(x)}{1 - \frac{d}{dy} x} = \lim_{y \to x} f''(y) = f''(x) \geq 0.$$

Thus we see that if f is convex and twice differentiable, $f''(x) \geq 0$.

As for the opposite direction, we need simply use the fact that f is twice differentiable (so that f'' is integrable) and we can prove the converse. So suppose that $f''(u) \geq 0$ for all u in $\operatorname{dom} f$. It easily follows that if $y > z$, we have that $f'(y) \geq f'(z)$, because $f''(u) \geq 0$ implies that

$$\int_z^y f''(u) du = f'(y) - f'(z) \geq \int_z^y 0 \cdot du = 0$$

Now we simply take the integral from x to y of our result (assuming $y \geq x$, though the negative signs just change if the opposite is the case), seeing

$$\begin{aligned} f'(y) - f'(z) &\geq 0 \\ \int_x^y f'(y) - f'(z)dz &\geq \int_x^y 0 \cdot dz \\ f'(y)z|_x^y - f(z)|_x^y &\geq 0 \\ f'(y)(y-x) - f(y) + f(x) &\geq 0 \\ \Rightarrow \quad f(x) &\geq f(y) + f'(y)(x-y) \end{aligned}$$

This shows that a tangent to f is a global underestimator of f, so f is convex if $f''(x) \geq 0$ for $x \in \operatorname{dom} f$.

In the end, we see that a function f is convex if and only if $f''(x) \geq 0$.

3.1 Multivariate case

In the multivariate case, we would like to prove that if the hessian $\nabla^2 f$ of a function $f : \mathbb{R}^n \to \mathbb{R}$ is positive semi-definite, that is, $\nabla^2 f \succeq 0$ or $\boldsymbol{x}^T \nabla^2 f \boldsymbol{x} \geq 0$, that f is convex.

First, suppose that $\nabla^2 f \succeq 0$. Now, consider the single variable function g that restricts f to some arbitrary line in f's domain, that is, $\operatorname{dom} g = [0, 1]$, and

$$g(t) = f(t\boldsymbol{x} + (1-t)\boldsymbol{y})$$

for $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} f$. We now take the first and second derivatives of g with respect to t:

$$\begin{aligned} \frac{d}{dt} g(t) &= \nabla f(t\boldsymbol{x} + (1-t)\boldsymbol{y})^T (\boldsymbol{x} - \boldsymbol{y}) \\ \frac{d^2}{dt^2} g(t) &= (\boldsymbol{x} - \boldsymbol{y})^T \nabla^2 f(t\boldsymbol{x} + (1-t)\boldsymbol{y})(\boldsymbol{x} - \boldsymbol{y}) \\ g''(t) &\geq 0. \end{aligned}$$

Now, suppose that f is convex, and again restrict it to a line using g. Because g is the intersection of a convex set with f, g is convex and hence $g''(t) \geq 0$. Thus, we have that for all $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} f$,

$$g''(t) = (\boldsymbol{x} - \boldsymbol{y})^T \nabla^2 f(t\boldsymbol{x} + (1-t)\boldsymbol{y})(\boldsymbol{x} - \boldsymbol{y}) \geq 0.$$

As near as I can tell, we have proved that a twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ with open, convex domain is convex if and only if $\nabla^2 f \succeq 0$.

4 Iterative Methods for Optimization

In this section, we will be considering the problem of minimizing some function $f : \mathbb{R}^n \to \mathbb{R}$.

4.1 Steepest Decent

In general, we know that $\nabla f(\boldsymbol{x})$ is the direction of the fastest ascent for the function $f(\boldsymbol{x})$. As such, to minimize the function $f(\boldsymbol{x})$, we would like to move our solution in the $-\nabla f$ direction. For steepest descent, then, we define the update operator for our estimate $\boldsymbol{x}_{k+1}$ at step $k+1$ to be

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k).$$

To make this a smarter method, we would like to take the largest steps possible at each step of the search, so we would like to travel in the $-\nabla f(\boldsymbol{x})$ direction until $\|\nabla f(\boldsymbol{x})\|$ is small (or zero). To this end, we can define a function $\boldsymbol{x}_k(t) = \boldsymbol{x}_k - \nabla f(\boldsymbol{x}_k)t$, so that we would like to minimize

$$f(\boldsymbol{x}_k(t)) = f(\boldsymbol{x}_k - \nabla f(\boldsymbol{x}_k) \cdot t) = g(t),$$

which is simply a one-dimensional minimization problem we can solve using a line search. It is easy to see that if we minimize $g(t)$, we have traveled as far along the path of $-\nabla f(\boldsymbol{x}_k)$ as we can without making our function grow again.

As a more specific example, which can be enlightening, consider the function f defined as follows (where A is symmetric and positive definite)

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} + c$$

whose derivative, it is easy to see, is

$$\nabla f(\boldsymbol{x}) = \frac{1}{2}A\boldsymbol{x} + \frac{1}{2}A^T\boldsymbol{x} - \boldsymbol{b} = A\boldsymbol{x} - \boldsymbol{b}$$

Thus, to solve this minimization problem, we must solve $A\boldsymbol{x} = \boldsymbol{b}$.

For steepest descent, we know that our search direction is $-\nabla f(\boldsymbol{x}) = \boldsymbol{b} - A\boldsymbol{x}$, and we also know that the residual for any potential solution $\boldsymbol{x}$ is given by

$$\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x} = A\boldsymbol{x}_{\text{exact}} - A\boldsymbol{x} = A(\boldsymbol{x}_{\text{exact}} - \boldsymbol{x}) = A\boldsymbol{e}_{\text{error}}$$

Unfortunately, we very rarely know anything about our error $\boldsymbol{e}$, but we do know that in steepest descent we would like to go in the direction $\boldsymbol{r}$ until we are parallel to the level curves of our function, which is (as stated above) a one dimensional minimization problem. That is, we would like to have $\nabla f(\boldsymbol{x}_k) \perp \boldsymbol{s}_k$, where $\boldsymbol{s}_k$ is the search direction at iteration k of steepest descent. This implies that $\nabla f(\boldsymbol{x})^T\boldsymbol{s}_k = 0$ if and only if $\nabla f(\boldsymbol{x}_{k+1})^T\boldsymbol{s}_k = 0$. Recognizing that $\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k$, we can solve for our updates as follows:

$$\begin{aligned}
0 &= \boldsymbol{r}_{k+1}^T\boldsymbol{r}_k \\
&= (\boldsymbol{b} - A\boldsymbol{x}_{k+1})^T\boldsymbol{r}_k \\
&= (\boldsymbol{b} - A(\boldsymbol{x}_k + \alpha\boldsymbol{s}_k))^T\boldsymbol{r}_k \\
&= (\boldsymbol{b} - A\boldsymbol{x}_k - \alpha A\boldsymbol{r}_k)^T\boldsymbol{r}_k \\
&= (\boldsymbol{b} - A\boldsymbol{x}_k)^T\boldsymbol{r}_k - \alpha(A\boldsymbol{r}_k)^T\boldsymbol{r}_k \\
&= \boldsymbol{r}_k^T\boldsymbol{r}_k - \alpha\boldsymbol{r}_k^T A\boldsymbol{r}_k
\end{aligned}$$

In the end, this gives that $\alpha = \frac{\boldsymbol{r}_k^T\boldsymbol{r}_k}{\boldsymbol{r}_k^T A\boldsymbol{r}_k}$.

In short, here is the steepest descent method used to find the solution to the equation $A\boldsymbol{x} = \boldsymbol{b}$:

$$\begin{aligned}
\boldsymbol{r}_k &= \boldsymbol{b} - A\boldsymbol{x}_k \\
\alpha &= \frac{\boldsymbol{r}_k^T\boldsymbol{r}_k}{\boldsymbol{r}_k^T A\boldsymbol{r}_k} \\
\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \alpha\boldsymbol{r}_k
\end{aligned}$$

The more general formulation is

$$\begin{aligned}
\alpha &= \operatorname*{argmin}_t f(\boldsymbol{x}_k - \nabla f(\boldsymbol{x}_k) \cdot t) \\
\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \alpha\nabla f(\boldsymbol{x}_k)
\end{aligned}$$

4.1.1 Digression on Choosing Directions

As a thought experiment, suppose our initial guess had an error $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_{\text{exact}}$, where $\boldsymbol{e}$ is an eigenvector of A. This would give us that

$$\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x} = A\boldsymbol{x}_{\text{exact}} - A\boldsymbol{x} = A(\boldsymbol{x}_{\text{exact}} - \boldsymbol{x}) = -A\boldsymbol{e} = -\lambda\boldsymbol{e}$$

With the above, we see that

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \frac{\boldsymbol{r}_k^T\boldsymbol{r}_k}{\boldsymbol{r}_k^T A\boldsymbol{r}_k}\boldsymbol{r}_k \\
&= \boldsymbol{x}_k + \frac{\boldsymbol{r}_k^T\boldsymbol{r}_k}{\boldsymbol{r}_k^T A(-\lambda\boldsymbol{e})} \cdot (-\lambda\boldsymbol{e}) \\
&= \boldsymbol{x}_k + \frac{\boldsymbol{r}_k^T\boldsymbol{r}_k}{\lambda\boldsymbol{r}_k^T\boldsymbol{e}}\boldsymbol{e} \\
&= \boldsymbol{x}_k + \frac{\boldsymbol{r}_k^T\boldsymbol{r}_k}{-\boldsymbol{r}_k^T\boldsymbol{r}_k}\boldsymbol{e} \\
&= \boldsymbol{x}_k - \boldsymbol{e} = \boldsymbol{x}_k - (\boldsymbol{x}_k - \boldsymbol{x}_{\text{exact}}) \\
&= \boldsymbol{x}_{\text{exact}}
\end{aligned}$$

What this tells us is that if we can pick an eigenvector of the matrix as our search direction, we can get our result immediately. That would be pretty neat.

4.2 Conjugate Gradient

Conjugate gradient is a search method that tries to choose its search directions not completely orthogonally, but by recalling its previous search directions and scaling its newest directions with them, which generally leads to a nice speedup. This scaling makes the directions it searches *conjugate*; a more specific definition is that two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are conjugate (with respect to a matrix A) if they are orthogonal in the inner product $(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T A\boldsymbol{y} = 0$.

To motivate this section, we are again going to be considering solving the system $A\boldsymbol{x} = \boldsymbol{b}$, where A is symmetric and positive definite. Now, let us move on to trying to solve our minimization problem by picking orthogonal search directions. That is, we would like to pick directions $\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_k$, where $\boldsymbol{s}_i^T\boldsymbol{s}_j = 0$ for $i \neq j$. Recall that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha\boldsymbol{s}_k$, which implies that

$$\begin{aligned}
\boldsymbol{x}_{k+1} - \boldsymbol{x}_{\text{exact}} &= \boldsymbol{x}_k - \boldsymbol{x}_{\text{exact}} + \alpha\boldsymbol{s}_k \\
\boldsymbol{e}_{k+1} &= \boldsymbol{e}_k + \alpha\boldsymbol{s}_k \qquad (3)
\end{aligned}$$

If we could go in in one direction of search until all error was orthogonal to the search direction, we could easily take large steps in the search space for the minimum of our function. The problem is that we do not know our error exactly, so we would like to eliminate all of our residual errors.

To this end, we make what are known as A-orthogonal steps rather than strictly orthogonal directions (we will see why in a bit). To eliminate our residual errors, we have

$$\begin{aligned}
0 &= \boldsymbol{r}_{k+1}^T\boldsymbol{s}_k = -(A\boldsymbol{e}_{k+1})^T\boldsymbol{s}_k \\
&= -A(\boldsymbol{e}_k + \alpha\boldsymbol{s}_k)^T\boldsymbol{s}_k
\end{aligned}$$

From this, we can see that

$$\alpha = \frac{\boldsymbol{s}_k^T A\boldsymbol{e}_k}{\boldsymbol{s}_k^T A\boldsymbol{s}_k} = \frac{\boldsymbol{s}_k^T\boldsymbol{r}_k}{\boldsymbol{s}_k^T A\boldsymbol{s}_k}$$

This is A-orthogonal because we would like to constrain our search so that $\boldsymbol{s}_j^T A\boldsymbol{s}_k = 0$ for all $j \neq k$.

Let us suppose that we would like to write our error as the scaled sum of a series of our search terms. That is, $\boldsymbol{e}_0 = \sum_j a_j \boldsymbol{s}_j$ (this sum takes into account all steps in the search). From this, we can derive some interesting properties of our error.

$$\begin{aligned}
\boldsymbol{s}_k^T A \boldsymbol{e}_0 &= \boldsymbol{s}_k^T A \sum_j a_j \boldsymbol{s}_j \\
&= \sum_j a_j \boldsymbol{s}_k^T A \boldsymbol{s}_j \\
&= a_k \boldsymbol{s}_k^T A \boldsymbol{s}_k + \sum_{j \neq k} a_j \boldsymbol{s}_k A \boldsymbol{s}_j \\
&= a_k \boldsymbol{s}_k^T A \boldsymbol{s}_k \qquad (4)
\end{aligned}$$

where equation (4) follows because we are using A-orthogonal directions, so the summation goes to zero. This tells us that (again using A-orthogonal directions)

$$a_k = \frac{\boldsymbol{s}_k^T A \boldsymbol{e}_0}{\boldsymbol{s}_k^T A \boldsymbol{s}_k} = \frac{\boldsymbol{s}_k^T A (\boldsymbol{e}_0 + \sum_{j=0}^{k-1} \alpha_j \boldsymbol{s}_j)}{\boldsymbol{s}_k^T A \boldsymbol{s}_k}$$

Consider equation (3): $\boldsymbol{e}_{k+1} = \boldsymbol{e}_k + \alpha \boldsymbol{s}_k$. From this, we can see that

$$\begin{aligned}
\boldsymbol{e}_k &= \boldsymbol{e}_{k-1} + \alpha_k \boldsymbol{s}_{k-1} = \boldsymbol{e}_{k-2} + \alpha_{k-2} \boldsymbol{s}_{k-2} + \alpha_{k-1} \boldsymbol{s}_{k-1} \\
&= \boldsymbol{e}_0 + \sum_{j=0}^{k-1} \alpha_j \boldsymbol{s}_j
\end{aligned}$$

Because of this, we have

$$a_k = \frac{\boldsymbol{s}_k^T A \boldsymbol{e}_k}{\boldsymbol{s}_k^T A \boldsymbol{s}_k} = -\frac{\boldsymbol{s}_k^T \boldsymbol{r}_k}{\boldsymbol{s}_k^T A \boldsymbol{s}_k} = -\alpha_k$$

This allows us to consider the error at the k^{th} step of our search.

$$\begin{aligned}
\boldsymbol{e}_k &= \boldsymbol{e}_0 + \sum_{j=0}^{k-1} \alpha_j \boldsymbol{s}_j = \sum_j a_j \boldsymbol{s}_j + \sum_{j=0}^{k-1} \alpha_j \boldsymbol{s}_j \\
&= -\sum_j \alpha_j \boldsymbol{s}_j + \sum_{j=0}^{k-1} \alpha_j \boldsymbol{s}_j
\end{aligned}$$

If we pre-multiply $\boldsymbol{e}_k$ by one of our search directions transposed with A, in the case when $i \leq k-1$, we have

$$\begin{aligned}
\boldsymbol{s}_i^T A \boldsymbol{e}_k &= -\boldsymbol{s}_i^T \boldsymbol{r}_k \\
&= -\sum_j \alpha_j \boldsymbol{s}_i^T A \boldsymbol{s}_j + \sum_{j=0}^{k-1} \alpha_j \boldsymbol{s_i}^T A \boldsymbol{s}_j \\
&= -\alpha_i \boldsymbol{s}_i^T A \boldsymbol{s_i} + \alpha_i \boldsymbol{s}_i^T A \boldsymbol{s}_i \\
&= 0 \qquad (5)
\end{aligned}$$

Similarly, for $i \geq k$, we see that

$$-\boldsymbol{s}_i^T \boldsymbol{r}_k = -\alpha_i \boldsymbol{s}_i^T A \boldsymbol{s}_i$$

Note also from the second equation above that if we find a search direction for every dimension of the space we are trying to optimize, then our error at step k will go to zero.

We now have a lot of math that is really doing very little for us unless we can find our A-orthogonal directions to optimize. We can use simple Gram-Schmidt orthonormalization to remove the components of search directions already seen to choose a new search direction. That is, for some $\boldsymbol{v}$ not in the direction of $\boldsymbol{s}_k$, to get a our new vector $\boldsymbol{s}_k$ which is A-orthogonal to all of the $\boldsymbol{s}_j$, we use the following:

$$\boldsymbol{s}_k = \boldsymbol{v} - \sum_{j=0}^{k-1} \frac{\boldsymbol{v}^T A \boldsymbol{s}_j}{\boldsymbol{s}_j^T A \boldsymbol{s}_j} \boldsymbol{s}_j$$

For $i \leq k-1$, we have

$$\begin{aligned} \boldsymbol{s}_k^T A \boldsymbol{s}_i &= \boldsymbol{v}^T A \boldsymbol{s}_i - \sum_{j=0}^{k-1} \frac{\boldsymbol{v}^T A \boldsymbol{s}_j}{\boldsymbol{s}_j^T A \boldsymbol{s}_j} \boldsymbol{s}_j^T A \boldsymbol{s}_i \\ &= \boldsymbol{v}^T A \boldsymbol{s}_i - \frac{\boldsymbol{v}^T A \boldsymbol{s}_i}{\boldsymbol{s}_i^T A \boldsymbol{s}_i} \boldsymbol{s}_i^T A \boldsymbol{s}_i = 0 \end{aligned}$$

where the last line follows because our previous search vectors $\boldsymbol{s}_j$ have all been A-orthogonalized. In the other case for i, that $i \geq k$, we see that

$$\begin{aligned} \boldsymbol{s}_k^T \boldsymbol{r}_i &= \boldsymbol{v}^T \boldsymbol{r}_i - \sum_{j=0}^{k-1} \frac{\boldsymbol{v}^T A \boldsymbol{s}_j}{\boldsymbol{s}_j^T A \boldsymbol{s}_j} \boldsymbol{s}_j^T \boldsymbol{r}_i \\ &= \boldsymbol{v}^T \boldsymbol{r}_i \end{aligned} \tag{6}$$

because, as we proved above in equation (5), $\boldsymbol{s}_j^T \boldsymbol{r}_i = 0$ for $j < i$. We now have a relatively tedious way to calculate our next orthogonal search directions; armed with this, we can calculate the distance we should take steps in our search. Recalling that $\alpha_k = \frac{\boldsymbol{s}_k^T \boldsymbol{r}_k}{\boldsymbol{s}_k^T A \boldsymbol{s}_k}$, we see that $\boldsymbol{s}_k^T \boldsymbol{r}_k = \boldsymbol{v}^T \boldsymbol{r}_k$ implies that $\alpha_k = \frac{\boldsymbol{v}^T \boldsymbol{r}_k}{\boldsymbol{s}_k^T A \boldsymbol{s}_k}$.

Naturally, there is a simpler way to calculate the new search directions using the old ones than an expensive Gram-Schmidt orthogonalization step. Consider that

$$\boldsymbol{r}_{k+1} = b - A\boldsymbol{x}_{k+1} = b - A(\boldsymbol{x}_k + \alpha_k \boldsymbol{s}_k) = b - A\boldsymbol{s}_k - \alpha_k A \boldsymbol{s}_k = \boldsymbol{r}_k - \alpha_k A \boldsymbol{s}_k$$

so that $\alpha_k A \boldsymbol{s}_k = \boldsymbol{r}_k - \boldsymbol{r}_{k+1}$. This tells us that

$$\alpha_k \boldsymbol{r}_i^T A \boldsymbol{s}_k = \boldsymbol{r}_i^T \boldsymbol{r}_k - \boldsymbol{r}_i^T \boldsymbol{r}_{k+1} \text{ and } \boldsymbol{r}_i^T \boldsymbol{r}_{k+1} = \boldsymbol{r}_i^T \boldsymbol{r}_k - \alpha_k \boldsymbol{r}_i^T A \boldsymbol{s}_k$$

Given that we are trying to find the $k+1$ search directions from the previous step, we set $i = k+1$. Now, we have that $\alpha_k \boldsymbol{r}_{k+1}^T A \boldsymbol{s}_k = \boldsymbol{r}_{k+1}^T \boldsymbol{r}_k - \boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1}$. Because when $k < i$, $\boldsymbol{v}_k^T \boldsymbol{r}_i = \boldsymbol{s}_k^T \boldsymbol{r}_i = 0$ (see equations (6) and (5)), we know that if we simply set $\boldsymbol{v}_k$ to be $\boldsymbol{r}_k$, then we have that $\boldsymbol{r}_k^T \boldsymbol{r}_i = 0$. As such, since $k < k+1$, we see that

$$\boldsymbol{r}_{k+1}^T A \boldsymbol{s}_k = \frac{\boldsymbol{r}_{k+1}^T \boldsymbol{r}_k - \boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1}}{\alpha_k} = -\frac{\boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1}}{\alpha_k}$$

Naturally, when $i > k+1$, we have $\alpha_k \boldsymbol{r}_i^T A \boldsymbol{s}_k = 0$.

Taking the above two paragraphs and combining them into the update rule for our search directions, (recalling that we set $\boldsymbol{r}_k = \boldsymbol{v}_k$), we have the following update rule:

$$\begin{aligned} \boldsymbol{s}_{k+1} &= \boldsymbol{r}_{k+1} - \sum_{j=0}^{k} \frac{\boldsymbol{r}_{k+1}^T A \boldsymbol{s}_j}{\boldsymbol{s}_j^T A \boldsymbol{s}_j} \boldsymbol{s}_j \\ &= \boldsymbol{r}_{k+1} + \frac{\boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^T \boldsymbol{r}_k} \boldsymbol{s}_k \end{aligned}$$

The sum goes to zero except for the $j = k$ term, since $\alpha_j \boldsymbol{r}_{k+1}^T A \boldsymbol{s}_j = 0$ except when $j = k$.

Combining all of the above work for minimizing the function $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} + c$, we have the following set of update rules for the conjugate gradient method:

$$\begin{aligned}
\boldsymbol{s}_0 &= \boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0 \\
\alpha_k &= \frac{\boldsymbol{r}_k^T \boldsymbol{r}_k}{\boldsymbol{s}_k^T A \boldsymbol{s}_k^T} \\
\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \alpha_k \boldsymbol{s}_k \\
\boldsymbol{r}_{k+1} &= \boldsymbol{r}_k - \alpha_k A \boldsymbol{s}_k \\
\boldsymbol{s}_{k+1} &= \boldsymbol{r}_{k+1} + \frac{\boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^T \boldsymbol{r}_k} \boldsymbol{s}_k
\end{aligned}$$

In the more general case, when we are trying to optimize an arbitrary function $f : \mathbb{R}^n \to \mathbb{R}$, we modify the method above a little. First, recall that for the case of the quadratic function of with A, $\boldsymbol{b}$, and c, that $\nabla f(\boldsymbol{x}) = A - \boldsymbol{b}^T\boldsymbol{x} = -\boldsymbol{r}$. As such, for an arbitrary f, we can see that $\boldsymbol{r}_k$ corresponds to $\nabla f(\boldsymbol{x}_k)$, giving us the more general conjugate gradient method.

$$\begin{aligned}
\boldsymbol{s}_0 &= -\nabla f(\boldsymbol{x}_0) \\
\alpha_k &= \operatorname*{argmin}_t f(\boldsymbol{x}_k - \boldsymbol{s}_k \cdot t) \\
\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \alpha_k \boldsymbol{s}_k \\
\boldsymbol{s}_{k+1} &= -\nabla f(\boldsymbol{x}_{k+1}) + \frac{\nabla f(\boldsymbol{x}_{k+1})^T \nabla f(\boldsymbol{x}_{k+1})}{\nabla f(\boldsymbol{x}_k)^T \nabla f(\boldsymbol{x}_k)} \boldsymbol{s}_k
\end{aligned}$$

5 Perceptron Convergence

Foregoing, temporarily, any introduction to the perceptron algorithm, let us denote a hyperplane by

$$f(\boldsymbol{x}) = \boldsymbol{\beta}_1^T \boldsymbol{x} + \beta_0 = 0$$

or simply by $\boldsymbol{\beta}^T \boldsymbol{x}_{\text{ext}} = 0$, where $\boldsymbol{x}_{\text{ext}} = \langle \boldsymbol{x}, 1 \rangle$. If we have separability, that is, that for some $\boldsymbol{\beta}$

$$y^{(i)} \boldsymbol{\beta}^T \boldsymbol{x}_{\text{ext}}^{(i)} > 0 \ \ \forall i$$

we can scale $\boldsymbol{\beta}$ accordingly to give us normalization for the points closest to the hyperplane. I do not have a great justification for this, but it is relatively (though somewhat unjustifiably) easy to see that if the above condition holds then for some $C > 0$ and $\boldsymbol{z}^{(i)} = \boldsymbol{x}_{\text{ext}}^{(i)} / \|\boldsymbol{x}_{\text{ext}}^{(i)}\|$, then $y^{(i)} \boldsymbol{\beta} \boldsymbol{z}^{(i)} \geq C$. Given this, we can simply divide both sides of the equation by C, giving us a new $\boldsymbol{\beta}_*$ such that

$$y^{(i)} \boldsymbol{\beta}_*^T \boldsymbol{z}^{(i)} \geq 1.$$

In the perceptron algorithm, we have a current estimate for $\boldsymbol{\beta}$, let us call that $\boldsymbol{\beta}_k$. The perceptron will identify a point $\boldsymbol{z}^{(i)}$ that is misclassified and produce the update $\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \boldsymbol{z}^{(i)} y^{(i)}$. With this, we can bound the difference of $\boldsymbol{\beta}_k$ and $\boldsymbol{\beta}_*$, which implies that the algorithm will converge to a separating hyperplane. Consider the following:

$$\begin{aligned}
\|\boldsymbol{\beta}_{k+1} - \boldsymbol{\beta}_*\|^2 &= \|\boldsymbol{\beta}_{k+1}\|^2 + \|\boldsymbol{\beta}_*\|^2 - 2\boldsymbol{\beta}_*^T \boldsymbol{\beta}_{k+1} \\
&= \|\boldsymbol{\beta}_k + \boldsymbol{z}^{(i)} y^{(i)}\|^2 + \|\boldsymbol{\beta}_*\|^2 - 2\boldsymbol{\beta}_*^T (\boldsymbol{\beta}_k + \boldsymbol{z}^{(i)} y^{(i)}) \\
&= 2y^{(i)} \boldsymbol{\beta}_k^T \boldsymbol{z}^{(i)} + \boldsymbol{\beta}_k^T \boldsymbol{\beta}_k + \|\boldsymbol{z}^{(i)}\|^2 + \|\boldsymbol{\beta}_*\|^2 - 2\boldsymbol{\beta}_*^T \boldsymbol{\beta}_k - 2y^{(i)} \boldsymbol{\beta}_*^T \boldsymbol{z}^{(i)} \\
&\leq 2y^{(i)} \boldsymbol{\beta}_k^T \boldsymbol{z}^{(i)} + \|\boldsymbol{\beta}_k\|^2 + 1 + \|\boldsymbol{\beta}_*\|^2 - 2\beta_*^T \beta_k - 2 \\
&= (\boldsymbol{\beta}_k - \boldsymbol{\beta}_*)^T (\boldsymbol{\beta}_k - \boldsymbol{\beta}_*) - 1 + 2y^{(i)} \boldsymbol{\beta}_k^T \boldsymbol{z}^{(i)} \\
&< \|\boldsymbol{\beta}_k - \boldsymbol{\beta}_*\|^2 - 1 \qquad (7)
\end{aligned}$$

where the fourth line follows because $y^{(i)}\boldsymbol{\beta}_*^T\boldsymbol{z}^{(i)} \geq 1$, and the last line, equation (7), follows because $\boldsymbol{\beta}_k$ misclassified $\boldsymbol{z}^{(i)}$, making the $y^{(i)}\boldsymbol{\beta}_k^T\boldsymbol{z}^{(i)}$ term negative. Noting this contraction, we see that the perceptron algorithm will converge to a separating hyperplane (in the case of separable data) in no more than $\|\boldsymbol{\beta}_0 - \boldsymbol{\beta}_*\|^2$ steps. Unfortunately, because the scaling can be very bad to give us the 1-margin separating hyperplane from above, this could be a huge number of steps. That, my friends, is why we use support vector machines.

6 Duality of SVMs

In this section, we consider the Support Vector Machine and its dual. We will focus on the case of data that is not separable, so that we are trying to minimize a norm of the vector w plus penalty terms for un-separated subject to data points. That is, we are seeking

$$\begin{array}{ll} \underset{w}{\text{argmin}} & \frac{1}{2}w^Tw + c^T\xi \\ \text{subject to} & y^{(i)}(w^Tx^{(i)} + b) \geq 1 - \xi_i, \;\; i = 1, \ldots, m \\ & \xi \succeq 0 \end{array}$$

The Lagrangian for this is

$$\mathcal{L}(w, \xi, \lambda, \beta) = \frac{1}{2}w^Tw + c^T\xi + \sum_{i=1}^m \lambda_i(1 - \xi_i - y^{(i)}(w^Tx^{(i)} + b)) - \beta^T\xi$$

Taking derivatives, we have that

$$\begin{aligned} \nabla_w\mathcal{L} &= w + \sum_{i=1}^m \lambda_i(-y^{(i)}x^{(i)}) \\ \nabla_b\mathcal{L} &= -\sum_{i=1}^m \lambda_i y^{(i)} \\ \nabla_\xi\mathcal{L} &= c - \lambda - \beta \end{aligned}$$

Setting each of the above to zero and solving, we have that $\lambda_i = c_i - \beta_i$, which, because $\beta_i > 0$ for the optimal solution (because of duality, etc.), implies that $0 \leq \lambda_i \leq c_i$ in the dual. We also have that $w = \sum_{i=1}^m \lambda_i y^{(i)}x^{(i)}$ and the constraint that in the dual, $\sum_{i=1}^m \lambda_i y^{(i)} = 0$. Substituting,

$$\frac{1}{2}\left(\sum_{i=1}^m \lambda_i y^{(i)}x^{(i)}\right)^T\left(\sum_{i=1}^m \lambda_i y^{(i)}x^{(i)}\right) + c^T\xi + \sum_{i=1}^m \lambda_i - \lambda^T\xi - \beta^T\xi - \sum_{i=1}^m \lambda_i y^{(i)}\left(\sum_{j=1}^m \lambda_j y^{(j)}(x^{(j)})^Tx^{(i)} + b\right)$$

After some simplification and using the Lagrange constraint from the ξ, we have that our dual optimization problem becomes

$$\begin{array}{ll} \underset{\lambda}{\text{argmax}} & -\frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m \lambda_i\lambda_j y^{(i)}y^{(j)}(x^{(i)})^Tx^{(j)} + \sum_{i=1}^m \lambda_i \\ \text{s.t.} & 0 \preceq \lambda \preceq c \\ & \sum_{i=1}^m \lambda_i y^{(i)} = 0 \end{array}$$

which can be optimized using pairwise coordinate ascent on the λ_is (as in the SMO algorithm).

7 Projection of a vector onto linear constraints

Let us suppose that we would like to project a vector z onto the set $\{x|Ax = b\}$. Thus, we would like to find the x that minimizes $||z - x||$ subject to $Ax = b$. First, we get our Lagrangian

$$\mathcal{L}(x, \lambda) = z^T z - 2x^T z + x^T x + \lambda^T (Ax - b)$$

whose derivative we take with respect to x. This gives us

$$\nabla_x \mathcal{L}(x, \lambda) = -2z + 2x + A^T \lambda.$$

Setting this to zero, we find that $x = z - A^T\lambda/2$. Plugging this into the lagrangian above, we have

$$\begin{aligned}
\mathcal{L}(\lambda) &= z^T z - 2\left(z - \frac{1}{2}A^T\lambda\right)^T z + \left(z - \frac{A^T\lambda}{2}\right)^T \left(z - \frac{A^T\lambda}{2}\right) + \lambda^T \left(A\left(z - \frac{A^T\lambda}{2}\right) - b\right) \\
&= z^T z - 2z^T z + \lambda^T Az + z^T z + \frac{\lambda^T AA^T\lambda}{4} - \lambda^T Az - +\lambda^T Az - \frac{\lambda^T AA^T\lambda}{2} - \lambda^T b \\
&= \lambda^T Az - \frac{\lambda^T AA^T\lambda}{4} - \lambda^T b
\end{aligned}$$

The derivative of the above with respect to λ is

$$\nabla_\lambda \mathcal{L}(\lambda) = Az - AA^T\lambda/2 - b,$$

and setting this to 0, we have

$$\lambda = 2(AA^T)^{-1}(Az - b).$$

Finally, plugging this into $x = z - A^T\lambda/2$, we have that

$$x = z - A^T(AA^T)^{-1}Az + A^T(AA^T)^{-1}b.$$

Thus, the projection of z onto the null space of A is $I - A^T(AA^T)^{-1}A$, and the projection onto $Ax = b$ given the projection into the nullspace is simply adding $A^T(AA^T)^{-1}b$, which is a solution of $Ax = b$.

8 Minimization of a linear function subject to elliptical constraints

In this section, we consider the following problem:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & x^T Ax \leq 1 \end{array}$$

where $A \in \mathbf{S}^n_{++}$. First, we form the Lagrangian:

$$\mathcal{L}(x, \nu) = c^T x + \nu(x^T Ax - 1).$$

Taking the gradient with respect to x, we have

$$\nabla_x \mathcal{L}(x, \nu) = c + 2\nu Ax$$

so that $x = -\frac{1}{2\nu}A^{-1}c$. We can solve assuming equality because unless the ellipse is unbounded or $c = 0$, the optimal value will be on the edge of the ellipsoid $x^T Ax \leq 1$. Plugging this in to the inequality constraint, we have that

$$\frac{1}{4\nu^2}c^T A^{-1}AA^{-1}c = 1 \quad \text{so} \quad \nu = \pm\frac{1}{2}||A^{-\frac{1}{2}}c||_2.$$

Plugging all this in, we find that the assignments to x and the minimum and maximum values are

$$x^* = \pm \frac{1}{||A^{-\frac{1}{2}}c||_2} A^{-1}c \quad \text{so} \quad c^T x^* = -\frac{1}{||A^{-\frac{1}{2}}c||_2} c^T A^{-1} c = ||A^{-\frac{1}{2}}c||_2.$$

9 KL Divergence for Gaussian distributions

First, recall that the KL divergence between two distributions P and Q is defined as

$$D_{KL}(P||Q) = \mathrm{E}_P \left[\log \frac{P}{Q} \right].$$

Also, the density function for a multivariate Gaussian (normal) distribution with mean μ and covariance matrix Σ is

$$p(x) = \frac{1}{(2\pi)^{n/2} \det(\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu) \right).$$

Now, consider two multivariate Gaussians in $\mathbb{R}^n$, P_1 and P_2. We have

$$\begin{aligned}
& D(P_1||P_2) \\
&= \mathrm{E}_{P_1}[\log P_1 - \log P_2] \\
&= \frac{1}{2}\mathrm{E}_{P_1}\left[-\log\det\Sigma_1 - (x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + \log\det\Sigma_2 + (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2) \right] \\
&= \frac{1}{2}\log\frac{\det\Sigma_2}{\det\Sigma_1} + \frac{1}{2}\mathrm{E}_{P_1}\left[-(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2) \right] \\
&= \frac{1}{2}\log\frac{\det\Sigma_2}{\det\Sigma_1} + \frac{1}{2}\mathrm{E}_{P_1}\left[-\mathrm{tr}(\Sigma_1^{-1}(x-\mu_1)(x-\mu_1)^T) + \mathrm{tr}(\Sigma_2^{-1}(x-\mu_2)(x-\mu_2)^T) \right] \\
&= \frac{1}{2}\log\frac{\det\Sigma_2}{\det\Sigma_1} + \frac{1}{2}\mathrm{E}_{P_1}\left[-\mathrm{tr}(\Sigma_1^{-1}\Sigma_1) + \mathrm{tr}(\Sigma_2^{-1}(xx^T - 2x\mu_2^T + \mu_2\mu_2^T)) \right] \\
&= \frac{1}{2}\log\frac{\det\Sigma_2}{\det\Sigma_1} - \frac{1}{2}n + \frac{1}{2}\mathrm{tr}(\Sigma_2^{-1}(\Sigma_1 + \mu_1\mu_1^T - 2\mu_2\mu_1^T + \mu_2\mu_2^T)) \\
&= \frac{1}{2}\left(\log\frac{\det\Sigma_2}{\det\Sigma_1} - n + \mathrm{tr}(\Sigma_2^{-1}\Sigma_1) + \mathrm{tr}(\mu_1^T\Sigma_2^{-1}\mu_1 - 2\mu_1^T\Sigma_2^{-1}\mu_2 + \mu_2^T\Sigma_2^{-1}\mu_2) \right) \\
&= \frac{1}{2}\left(\log\frac{\det\Sigma_2}{\det\Sigma_1} - n + \mathrm{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2-\mu_1)^T\Sigma_2^{-1}(\mu_2-\mu_1) \right)
\end{aligned}$$