1c) $\sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i \left(\frac{1}{3}\right) r = \boxed{\frac{r}{3}}$ since $\sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i$ converges to 1

"respin"

1d) $L(p) = p^3(1-p)^2$

$\ln(L(p)) = \ln(p^3) + \ln((1-p)^2)$

$\ln(L(p)) = 3\ln(p) + 2\ln(1-p)$

$\max(f(x)) = \max(\ln(f(x)))$
for $f(x) \geq 0$

$\frac{d}{dp} \ln(L(p)) = \frac{3}{p} - \frac{2}{p-1} = \frac{5p-3}{p^2-p} = 0 \quad \boxed{p = \frac{3}{5}}$

1e) $\boxed{\nabla f(w) = 2\sum_{i}^{n}\sum_{j}^{n} (a_i^T w - b_j^T w)(a_i^T - b_j^T) + 2\lambda \sum_{j}^{d} w_j}$

1a) $f'(x) = \sum_{i=1}^{n} w_i(x-b_i) = 0 \quad \sum_{i=1}^{n} w_i x = \sum_{i=1}^{n} w_i b_i$

$$\boxed{x = \frac{\sum_{i=1}^{n} w_i b_i}{\sum_{i=1}^{n} w_i}}$$

1b) At every step of the sum, $g(x)$ can choose a to match $\text{sign}(x_j)$. Therefore the product of a and x_j is always positive. $f(x)$ can only choose 1 a for the entire function and therefore when x contains elements that are both positive and negative, the products of some of those iterations will be negative and therefore less than $g(x)$.

3 cases: $x \in \mathbb{R}^d$

$\forall x_j \geq 0 \quad f(x) = g(x)$

$\forall x_j \leq 0 \quad f(x) = g(x)$

$\exists x_j > 0 \wedge \exists x_j < 0 \quad g(x) > f(x)$

$\boxed{\text{Therefore } g(x) \geq f(x)}$

Scanned by CamScanner

2a) rectangle with a width and b length has $(n-a+1)(n-b+1)$ placements.

$$\frac{1}{2}\sum_{a}^{n}\sum_{b}^{n}(n-a+1)(n-b+1) \simeq n^4 \text{ for one rectangle}$$

$$n^4 \cdot n^4 \cdot n^4 = n^{12} \qquad \boxed{O(n^{12})}$$

2b) We can use dynamic programming to solve this efficiently. At any city j, take the minimum of all your options backwards. The run time is

$\boxed{O(n^2)}$

2c) This problem is a reformulation of pascals triangle. To solve for the lower right hand corner we need to use the formula for binomial coefficients and account for our axis (2x2 vs 3x3).

1	1	1	1
1	2	3	4
1	3	6	10
1	4	10	20

$$\boxed{\frac{(2(n-1))!}{((n-1)!)^2}}$$

2d) $f(w) = \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i^T w - b_j^T w)^2 + \lambda \|w\|_2^2$

$$= w^2 \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i^T - b_j^T)^2 + \lambda \sum_{j=1}^{d} w_j^2$$

where I factored w out and moved it through the summation.

Since the double summation does not depend on w, it can be done in preprocessing. The preprocessing takes $O(nd^2)$.

Also realize that $w^2 = w \cdot w = \sum_{j=1}^{d} w_j^2$

so this only needs to be computed once.

The time to compute $f(w) \in O(d^2)$