Probability of points being “positive linear optimal” in high dimensionality

Ziao Wang

1. Background

Lemma 1

If X follows a d-dimensional standard Gaussian distribution, and v is a fixed unit vector, then $X \cdot d \sim N(0,1)$.

Justification: the d-dimensional standard Gaussian distribution is symmetric on any direction, like a ball; or, use the property that any linear combination of a multivariate normal vector is itself a univariate normal random variable.

Lemma 2

When $n >> 0$, the maximum value among n standard Gaussians is very unlikely to exceed $\sqrt{2\ln n}$.

Relying on Lemma 5.3 in “Concentration for Maximum” from a UCLA lecture by Partha S. Dey (scribe: Hsin-Po Wang) to supports this claim. Copying Lemma 5.3 here:

Suppose M_n is the maximum of n i.i.d. standard normals, then

$$P[M_n \geq \sqrt{2\log n} + \frac{x}{\sqrt{2\log n}}] \leq e^{-x}$$

Lemma 3

If x is standard Gaussian, its mean value given that $x > 0$ is $\sqrt{\frac{2}{\pi}}$. That is, $x \sim N(0,1) \to E[x|x>0] = \sqrt{\frac{2}{\pi}}$.

Derivation: $E[x|x>0] = \sqrt{\frac{2}{\pi}} \int_{x=0}^{\infty} xe^{-x^2/2}dx = \sqrt{\frac{2}{\pi}} \int_{u=0}^{\infty} e^{-u}du = \sqrt{\frac{2}{\pi}}$

2. Problem

Let$X_1, ..., X_n$ be i.i.d. samples from a d-dimensional standard Gaussian. We say sample X_i is “positive linear optimal” (PLO) if there exists a d-dimensional vector k whose elements are non-negative such that $X_i \cdot k > X_j \cdot k, \forall j \neq i$. Suppose $n >> d >> \ln n$. As $n \to \infty$, is every point almost always, for some probability, or almost never positive linear optimal?

2.1 Lower bound

If X_i is greatest in any dimension r, X_i is definitely for PLO - just choose $k = e_r$. The probability of being greatest in any dimension is $\frac{1}{n}$, so the probability of being greatest in any dimension is $1-(1-\frac{1}{n})^d \sim \frac{d}{n} \to 0$, a lower bound too loose.

2.2 Upper bound

If X_i is PLO, then X_i must be Pareto optimal: $\forall j$, we must not have $X_j > X_i$ on every dimension (i.e. X_i is dominated by X_j). The probability of being PLO is upper bounded by being Pareto optimal. The latter can be estimated as follows: the probability of X_i being dominated by X_j is the product of every dimension, $\frac{1}{2^d}$. Assume that $X_j > X_i$ is independent from $x_m > X_i$. Then the probability of X_i being Pareto optimal is about $1 - \frac{n}{2^d} \to 1$, also too loose.

3. Construction

Claim: Every sample is almost always PLO.

Construction: For sample X_i, we choose k to be $\frac{X_i^+}{||X_i^+||}$, where X_i^+ is the vector formed by taking the components of X_i that are positive and setting the non-positive components to 0. We see that k is a unit vector.

3.1. The Candidate's score

We know that every component of X_i^+ follows the "clipped" standard Gaussian, where the probability mass for negative values are clipped to 0 (which has 50% chance of happening). Therefore the "score" S (we use it to refer to the dot product of a sample with k) of X_i is

$$S = X_i \cdot k = X_i^+ \cdot k = X_i^+ \cdot \frac{X_i^+}{||X_i^+||} = ||X_i^+||$$

$$S^2 = X_i^+ \cdot X_i^+ = \Sigma_{r=1}^d (X_{i,r}^+)^2 \sim d/2$$

(Use the law of large numbers here)

3.2 The Competitor's score

X_i must compete against the rest of the samples. For $X_j, j \neq i$, since X_j is independent to k and k is a unit vector, we apply Lemma 1 and see $X_j \cdot k \sim N(0,1)$.

Among n-1 competitors, the strongest competitor's score T must not exceed $\sqrt{2 \ln n}$ a lot, according to Lemma 2. So $T^2 \sim 2 \ln n$.

3.3 Comparison

Since $d >> \ln n$ in the problem statement, we have $S^2 >> T^2$. Therefore every point is almost always PLO.

3.4 Variant: Allow competitors to exclude negative scores

Consider what happens if competitors are allowed to exclude negative components from the dot product. Or equivalently, $X_1^+, ..., X_n^+$ is sampled from a clipped d-dimensional standard Gaussian. The candidate's score wouldn't change; will the competitors' scores increase so much as to surpass the candidate? For each competitor j, its score $T_j = X_j^+ \cdot k = \frac{1}{||X_i^+||} \Sigma_{r=0}^d X_{jr}^+ \cdot X_{ir}^+$

Apply Lemma 3, we see that $E[X_{jr}^+] = \frac{1}{2} * 0 + \frac{1}{2} E[x | x > 0, x \sim N(0,1)] = \sqrt{\frac{1}{2\pi}}$. We also know that $E[(X_{jr}^+)^2] = \frac{1}{2}$, thus $Var[X_{jr}^+] = (\frac{1}{2} - \frac{1}{2\pi})$.

On the first order,

$$E[||X_i^+|| * T_j] = \sqrt{\frac{1}{2\pi}}\Sigma_{r=1}^d X_{ir}^+ := \sqrt{\frac{1}{2\pi}} s_1$$

For the variance,

$$Var[||X_i^+|| * T_j] = \Sigma_{r=1}^d (\frac{1}{2} - \frac{1}{2\pi}) * (X_{ir}^+)^2 := (\frac{1}{2} - \frac{1}{2\pi}) s_2$$

Since T_j is the sum of many independent distributions, apply the law of large numbers on the component dimension (assuming $d >> 1$), T_j is approximately a normal distribution:

$$||X_i^+|| * T_j \sim N(\sqrt{\frac{1}{2\pi}} s_1,\ (\frac{1}{2} - \frac{1}{2\pi}) s_2) := N(\mu, \sigma^2)$$

Apply Lemma 2 here, among n competitors

$$||X_i^+|| * T_{max} \sim \mu + \sqrt{2 \ln n} * \sigma$$

Recall that $||X_i^+|| = \sqrt{s_2}$

$$T_{max} \sim \sqrt{\frac{1}{2\pi}} \frac{s_1}{\sqrt{s_2}} + \sqrt{(1 - \frac{1}{\pi}) \ln n}$$

Considering that $s_1 \sim d\sqrt{\frac{1}{2\pi}}$, $s_2 \sim \frac{d}{2}$,

$$T_{max} \sim \frac{\sqrt{d/2}}{\pi} + \sqrt{(1 - \frac{1}{\pi}) \ln n}$$

When $d >> \log n$, $T_{max} \sim \frac{1}{\pi}\sqrt{d/2}$ while $S \sim \sqrt{d/2}$. The strongest competitor is no longer order of magnitudes smaller than S (thanks to the exclusion of negative scores), but instead a constant fraction (specifically, $\frac{1}{\pi}$) of S. That said, the conclusion doesn't change - X_i almost always wins.

4. Numerical Sanity Check

```
import numpy as np

N = 100000
d = 1000
Xs = np.random.normal(size=(N,d))

# Find the metric vector
X0 = Xs[0] # We demonstrate that X0 is optimal
X0p = X0.clip(min=0)
L2_X0p = np.linalg.norm(X0p)
X0p_hat = X0p / L2_X0p
X0_score = np.dot(X0p_hat, X0) # is also L2_X0p
print("X0_score: %.4f" % X0_score)

# Find scores
scores = Xs @ X0p_hat
scores = np.partition(scores, -2)
print("1st score: %.4f, squared = %.4f ~ d/2 = %.2f" % (scores[-1], scores[-1]**2,
     d/2))
print("2nd score: %.4f, squared = %.4f < bd, bd ~ 2log(n) = %.4f" % (scores[-2],
    scores[-2]**2, 2*np.log(N)))

print("If we allow other candidates to remove their negative scores")
Xs = Xs.clip(min=0)
```

```
scores = Xs @ X0p_hat
scores = np.partition(scores, -2)
dpart = np.sqrt(d/2)/np.pi
npart = np.sqrt((1-1/np.pi)*np.log(N))
print("2nd score: %.4f ~ sqrt{d/2pi^2} + sqrt{(1-1/pi)ln(n)} = %.4f" % (scores
    [-2], dpart+npart))
```

Example Result:

```
X0_score: 21.2264
1st score: 21.2264, squared = 450.5582 ~ d/2 = 500.00
2nd score: 4.3707, squared = 19.1034 < bd, bd ~ 2log(n) = 23.0259
If we allow other candidates to remove their negative scores
2nd score: 9.6617 ~ sqrt{d/2pi^2} + sqrt{(1-1/pi)ln(n)} = 9.9191
```