Matematikai Nyelvészet olvasószeminárium

Kornai András

2014 okt 10

Ritka vektoros kódolás (sparse overcomplete coding)

Ritka vektoros kódolás (sparse overcomplete coding)

- Hagyományos felállás: rövid vektorok (10-200 dim)

Ritka vektoros kódolás (sparse overcomplete coding)

- Hagyományos felállás: rövid vektorok (10-200 dim)
- Ujdonság: hosszu vektorok (10^5-10^6 dim de csak kevés nemnulla elem)

Ritka vektoros kódolás (sparse overcomplete coding)

- Hagyományos felállás: rövid vektorok (10-200 dim)
- Ujdonság: hosszu vektorok (10^5-10^6 dim de csak kevés nemnulla elem)
- Mikor jó? Ha "közeli" adatoknak Eukl. vagy Hamming-közeli vektorok felelnek meg

Ritka vektoros kódolás (sparse overcomplete coding)

- Hagyományos felállás: rövid vektorok (10-200 dim)
- Ujdonság: hosszu vektorok (10^5-10^6 dim de csak kevés nemnulla elem)
- Mikor jó? Ha "közeli" adatoknak Eukl. vagy Hamming-közeli vektorok felelnek meg
- Kezdetek: sparse distributed representation (Pentti Kanerva) erősen neurális filozófia

Ritka vektoros kódolás (sparse overcomplete coding)

- Hagyományos felállás: rövid vektorok (10-200 dim)
- Ujdonság: hosszu vektorok (10^5-10^6 dim de csak kevés nemnulla elem)
- Mikor jó? Ha "közeli" adatoknak Eukl. vagy Hamming-közeli vektorok felelnek meg
- Kezdetek: sparse distributed representation (Pentti Kanerva) erősen neurális filozófia
- Mai alakjára a szokásos gyanusítottak (Andrew Ng, Yann LeCun, ...) hozták

Mi az eszme?

Van sok (esetleg több millió) $\mathbf{y}$ bemenő vektorunk (k-dimenziós, k párszáz) és keresünk egy csomó (n, sokezer) ugyanilyen k hosszú $\mathbf{b}_i$ bázisvektort és egy $\mathbf{s}$ (n-hosszú) együtthatóvektort, amire $\mathbf{y} \approx \sum_i \mathbf{b}_i s_i$ (itt s_i az $\mathbf{s}$ együtthatóvektor i-edik komponense, csak kevés nem-nulla szerepel az összegben).

A k dimenziós vektorokhoz tkp. k darab bázisvektor is elég lenne, mi mégis sokkal többet (n-et) veszünk fel, ezért 'overcomplete' a bázis. A lényeg az, hogy a "közeli" adatoknak közeli $\mathbf{s}$ együtthatóvektorok feleljenek meg, másszóval a bázis olyan legyen, hogy a közeli adatok osztozzanak báziselemeken.

Miért jó ez?

Mert a sok (esetleg több millió) adat nagyrésze nincs cimkézve, a cimkézés drága. Viszont a k-dim térnek igazából csak egy kis manifoldját (sajnos nem alterét, hanem valami meghajlított felületet) töltik be, tehát a legjobb lenne valahogy úgy kódolni őket, hogy az csak lokálisan sima. Ahogy megyünk odébb a manifoldon, egyes bázisvektorokat eldobunk, és a helyükre másokat hozunk be.

A jelenség közismert az osztályozás minden területén. Pl. a halak közt nagyon jó megkülönböztető jegy a pikkelyes vagy símabőrű, vagy a gyerek/felnőtt, de a könyvtárak közt csak a második jegy hasznos, az elsőt szak/általános vagy hasonlókkal kell kicserélni.

Az első megközelítés (Lee Battle Raina Ng 2006)

Törekedjünk arra, hogy az $\mathbf{y} - \sum_i \mathbf{b}_i s_i$ hiba kicsi legyen. Tegyük fel, hogy ez eleve normális eloszlású 0 átlaggal és $\sigma^2 I$ kovarianciával. Gyűjtsük az $\mathbf{y}$ vektorokat egy Y mátrixba, a $\mathbf{b}_i$ báziselemeket B-be, és az $\mathbf{s}_j$ kódvektorokat S-be. A cél $||Y - BS||^2/2\sigma^2 + \beta \sum ||s||_1$ minimalizálása, ahol az első tag a mátrix Frobenius normája (az összes komponens négyzetösszege) a második pedig az L_1 regularizásió amitől az $\mathbf{s}_i$-k ritkák lesznek.

Megoldható kvadratikus optimalizációval (amihez túl nagy) vagy gradiens módszerrel (ami lassan konvergál) vagy speciális trükkökkel (adott esetben a Lagrange-duális megoldásával).

Goroshin and LeCun 2013

Auto-(en)kóder: olyan kóder, ami az adatból tanulja hogyan kell a kódvektorokat előállítani. Fentebb volt a 'ritka', de van 'zsugorító' (contractive) változat is. G&LC közös alapokra helyezik: [L1 penalty] prevents the a-e from learning to reconstruct all possible points in the input space and focuses the expressive power of the a-e on representing the data-manifold. Similarly, the contractive a-e avoids trivial solutions by introducing an auxiliary penalty which measures the square Frobenius norm of the Jacobian of the latent representation with respect to the inputs. This encourages a constant latent representation except around training samples where it is counteracted by the reconstruction term. These two approaches are strongly related. The contractive a-e explicitly encourages small entries in the Jacobian, whereas the sparse a-e is encouraged to produce mostly zero (sparse) activations which can be designed to correspond to mostly flat regions of the nonlinearity, thus also yielding small entries in the Jacobian. (See