Statistical Techniques in Robotics (16-831, F08) Lecture #25 (Nov. 18th)

Sparsity

Lecturer: Drew Bagnell *Scribe: Matthew Tesch*

1 Sparsity

Sparsity is important in two situations:

1. when you are interested in a lot of features and only a few are relevant.
2. when you have a small amount of data relative to the number of features.

Some examples of these situations include:

1. **Spam Filtering:** The feature vector is all words and all combinations of words, which is huge; it is likely that there are only a few which correlate strongly with spam.
2. **Computer Vision:** There are often a lot of features (filter outputs, patches, etc), but only a few images.
3. **Image Denoising:** When we want to remove lots of noise and recover an underlying image.

1.1 Minimizing Regret in Sparse Problems: Choosing the right norm

Regret bound for online gradient descent:

$$R \leq |F|_2 |G|_2 \sqrt{(}T), \tag{1}$$

where

1. R is the regret,
2. G is the gradient,
3. F is the size of our space, and
4. T is the number of training points.

Many things tie in into the size of G, but it is roughly proportional to $sqrt(d)$, where d is the amount of features. Therefore, we need order d examples if we have d features, so T is fixed. Also, if $|G|_2$ is big, we pay a lot of regret.

One way to reduce R is by using a smaller set of hypotheses. Ideally, we want a constant dependence on d, but this is unrealistic. Also, we would like to only be dependent on the largest element of G; for this we use the infinity norm. And, reducing the size of the hypothesis space is useful as well;

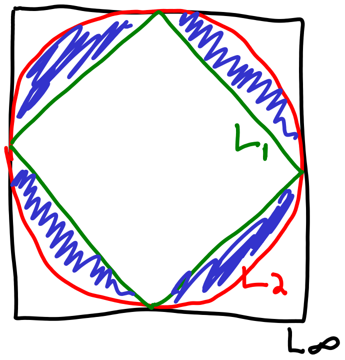

Figure 1: By choosing a different norm as a bound on a space, the volume can drastically change. The shaded blue area is an example of a region that contains points whose two-norms are less than some constant c, but one-norms are greater than c. Although in two dimensions, this is not a tremendous area difference, as the dimensionality of the space increases, this area represents a drastic reduction in size of the space of points with norm less than a set value.

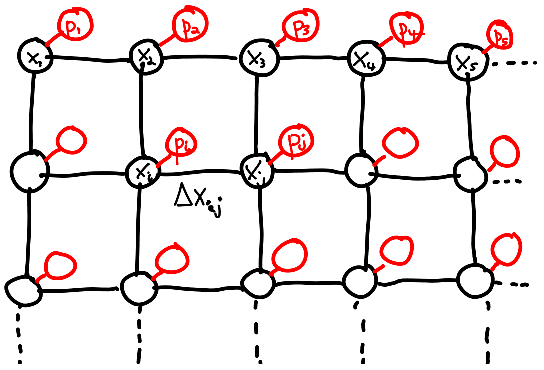

Figure 2: An example of a Markov random field that represents a noisy image. The p_{ij} is the observed value of the pixel, while x_{ij} is the value of the pixel in the original image.

to do so, we take the 1 norm of F instead of the 2 norm (see Figure 1). Note that this reduces the number of possible hypotheses, possibly removing the 'best' hypothesis, but the increase in performance is worth it for sparse problems. After these assumptions, our regret bound is now $R \leq |F|_1 |G|_\infty log(d)$, which is only a logarithmic dependence on d.

1.2 Where Sparsity Plays a Role: Total-Variation Method

In this problem, we have images that have accrued noise (through transmission, reproduction, or any other source), and we want to reconstruct the original, denoised picture. One method that could be used is a Markov random field; essentially, a lattice of pixels with neighbors connected (see Figure 2).

One important assumption is that most real images have a sparse number of changes; this is our prior on the image. Essentially, in a big grid of pixels, there are regions of constant pixel value, and only at the borders of these regions do the pixel values change; i.e. we assume sparsely-piecewise constant input images.

Therefore, to use this assumption to obtain the original image, we penalize changes in neighboring pixel values. However, we want large changes to not be that much worse than small changes, so we penalize the 1-norm $|\Delta x|_1$ instead of the two norm.

To complete the algorithm, we use our input image as evidence: since this provides a good estimate of the desired image, we penalize deviations from this image with a 2-norm: $\sum_i (p_i - x_i)^2$.

This forms the basis of total variation denoising. In this algorithm, we simultaneously minimize the 1 norm between our estimates of original pixels and the 2 norm between our observation and our estimated original image. In other words, we minimize $\sum_i (p_i - x_i)^2 + \lambda|\Delta x|_1$. A high value for λ emphasizes smoothing; a low value for λ emphasizes keeping original image.

Simplest solution: compute subgradient, do gradient descent, iterate, get answer.

1.3 The moral of the story:

If you want sparsh-ish solutions, choose the L_1 norm, not L_2. The L_2 norm penalizes large deviations much more thatn the L_1, and so produces a smoothing effect. The L_1 norm allows a sparse number of larger shifts, instead.

1.4 Another Method for Sparsity: Exponentiated Gradient Descent

The basic Exponentiated Gradient Descent algorithm is as follows:

Let α_t be step size. Then, for each gradient step:

1. Predict w_t.
2. Recieve loss l_t and ∇l_t.
3. Use one of the following multiplicative update rules:
   (a) $w_{t+1} = w_t(1 - \alpha_t \nabla l_t)$.
   (b) $w_{t+1} = w_t e^{-\alpha_t \nabla l_t}$.
   (c) $w_{t+1} = w_t(1 - \nabla l_t)^{\alpha_t}$.
4. Optional normalization step (usually used for probability distributions): $w_{t+1} = w_{t+1}/z$, where z is sum of the weights.

Although three are listed, they are not equally common; the third almost never gets used. Also, note that there is a 'matlab dot-multiply' in these updates (i.e. element-wise vector multiplication), and that the exponentiation is element-wise as well.

Some sanity checks, or 'does this algorithm behave as we would like?':

1. Grad = 0 $\rightarrow$ same weights.

2. Our update move is within 90 degrees of the ‘correct’ direction (the direction of the gradient). In other words, $\langle w_{t+1} - w_t, -\nabla l_t \rangle$ should be positive. This is easy to test for at least the first multiplicative update rule above. We plug in for w_{t+1} to get: $\langle w_t - w_t - w_t \alpha_t \nabla l_t, -\nabla l_t \rangle > 0$. Since the left side of the inner product is just a weighted version of the right, we ensure positive correlation.

One way to think of exponentiated gradient is that it is just like grad. descent, except with ’distance’ replaced by ’KL-divergence’. Also, we can bound the regret as $R \le k|F|_1|G|_\infty \sqrt{\log dT}$ where k is a constant.

The advantage to this multiplicative approach is that irrelevant features are killed really fast. One disadvantage is that we can only have positive weights; if we have features that might be anti-correlated, we can double the feature vector by adding a negative of each feature. Another problem with this algorithm is that every feature must be about at the same scale; this way the gradient is about the same magnitude for each. Large differences is scale can cause domination of the gradient by a single feature.

1.5 The problem in document recognition

The one-norm encourages high weights for a small number of features; low weights for high number of features. This never helps on text; almost always makes it worse. Feature vector is REALLY sparse, so adding thousands of features only slightly increases $|G|_\infty$ or $|G|_2$; there is not much difference between the two. The regular gradient rule works just fine and considers more experts in this case, and is a better choice because of this.

1.6 Winnow

For the binary case, winnow is just the exponentiated gradient with support vector loss (the loss without margin for true winnow).

2 Review:

Two cases of ’sparsity’:

- **Subject identification from text:** the input is sparse, the weight vector isn’t necessarily
- **TV denoising:** the weight vector is sparse, but the input is dense

Also, in the denoising problems, we are learning a basis, then we use it for classify later if desired. Important: we are not classifying original input image, but a set of weights that help you reconstruct original input image.

Overall, exponentiated gradient is good for sparse solution; bad when input is sparse, b/c we are constraining ourselves to small hypothesis small, even though 2-norm approximates infinity norm.