Norm Penalties as Constrained Optimization

Sargur N. Srihari
srihari@cedar.buffalo.edu

Regularization Strategies

1. Parameter Norm Penalties
2. Norm Penalties as Constrained Optimization
3. Regularization and Under-constrained Problems
4. Data Set Augmentation
5. Noise Robustness
6. Semi-supervised learning
7. Multi-task learning

8. Early Stopping
6. Parameter tying and parameter sharing
7. Sparse representations
8. Bagging and other ensemble methods
9. Dropout
10. Adversarial training
11. Tangent methods

Topics in Norm Penalty Optimization

1. Lagrangian formulation
2. KKT multiplier
3. Equivalence to norm penalty
4. Explicit constraints and Reprojection

Constrained Optimization

- Consider the cost function regularized by a norm penalty $\boxed{\tilde{J}(\theta;X,y)=J(\theta;X,y)+\alpha\Omega(\theta)}$
- Recall we can minimize a function subject to constraints by constructing a generalized Lagrange function, consisting of the original objective function plus a set of penalties
  - Each penalty is a product between a coefficient called a Karush-Kuhn-Tucker (KKT) multiplier and a function representing whether constraint is satisfied

Lagrange Formulation

- If we wanted to constrain $\Omega(\theta)$ to be less than some constant k, we could construct a generalized Lagrange function

$$\boxed{L(\theta,\alpha;X,y)=J(\theta;X,y)+\alpha\Big(\Omega(\theta)-k\Big)}$$

- The solution to the constrained problem is given by

$$\boxed{\theta^*=\arg\min_{\theta}\max_{\alpha,\alpha\geq 0}L\Big(\theta,\alpha\Big)}$$

- Solving this problem requires modifying both θ and α
  – Many different procedures are possible

Insight into effect of constraint

- We can fix α* and view the problem as just a function of θ:

$$\boxed{\theta^* = \arg\min_{\theta} L\left(\theta, \alpha^*\right) = \arg\min_{\theta} J(\theta; X, y) + \alpha^* \Omega(\theta)}$$

- This is exactly the same as the regularized training problem of minimizing $\boxed{\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha\Omega(\theta)}$
- We can thus think of the parameter norm penalty as imposing a constraint on the weights

How α influences weights

- If Ω is L^2 norm
  - weights are then constrained to lie in an L^2 ball
- If Ω is the L^1 norm
  - Weights are constrained to lie in a region of limited L^1 norm
- Usually we do not know size of constraint region that we impose by using weight decay with coefficient α^* because the value of α^* does not directly tell us the value of k
  - Larger α will result in smaller constraint region
  - Smaller α will result in larger constraint region

Reprojection

- Sometimes we may wish to use explicit constraints rather than penalties
  - We can modify SGD to take a step downhill on $J(\theta)$ and then project θ back to the nearest point that satisfies $\Omega(\theta)<k$
  - This useful when we have an idea of what values of k is appropriate and we do not want to spend time searching for the value of α that corresponds to this k
- Rationale for explicit constraints/Reprojection
  1. Dead weights
  2. Stability

1. Eliminating dead weights

- A reason to use explicit constraints and reprojection rather than enforcing constraints with penalties:
  - Penalties can cause nonconvex optimization procedures to get stuck in local minima corresponding to small θ
    - This manifests as training with *dead units*
  - Explicit constraints implemented by reprojection can work much better because they do not encourage weights to approach the origin

2. Stability of Optimization

- Explicit constraints with reprojection can be useful because thse impose some stability on the optimization procedure
- When using high learning rates, it is possible to encounter a positive feedback learning loop in which large weights induce large gradients, which then induce a large update of the weights
  - Can lead to numerical overflow
- Explicit constraints with reprojection prevent this feedback loop from continuing to increase magnitudes of weights without bound