06 - Regularization for Image Classification

Numerical Methods for Deep Learning

February 7, 2018

Regularization

- We are attempting to recover weights to fit some data.
- Simplest case - a single data $\mathbf{Y} = [x_1, x_2]$.
- In many cases (MNIST) non-unique solution (too many unknowns, too few equations)

In general - if the data "lives" in high dimensions (e.g. images) then we need many examples to have a unique classifier.
The examples need to be independent that is $\mathbf{Y}$ is full rank.

Regularization

- Symptom of the need for regularization - Hessian highly ill-conditioned
- Solution may be “wild” and oscillate.
- Add a demand on the solution to be regular

$$\min_{W} \; J(\mathbf{W};\mathbf{Y}) = E(\mathbf{W};\mathbf{Y}) + \alpha R(\mathbf{W})$$

Type of Regularization

Tikhonov

$$R(\mathbf{W}) = \frac{1}{2}\|\mathbf{W}\|_F^2$$

asks for all the entries to be small.

In some cases, $\mathbf{X}$ are images.

$$\mathbf{w}^\top \mathbf{x} \approx \int_\Omega w(\boldsymbol{\xi})\mathbf{x}(\boldsymbol{\xi})d\boldsymbol{\xi}.$$

Weighted Tikhonov

$$R(\mathbf{W}) = \frac{1}{2}\|\mathbf{LW}\|_F^2$$

asks for all the entries to be smooth.

Smooth Regularization

In some cases, $\mathbf{X}$ are images.

$$\mathbf{w}^\top \mathbf{x} \approx \int_\Omega w(\boldsymbol{\xi})\mathbf{x}(\boldsymbol{\xi})d\boldsymbol{\xi}.$$

Weighted Tikhonov

$$R(\mathbf{W}) = \frac{1}{2}\|\mathbf{L}\mathbf{W}\|_F^2$$

asks for all the entries to be smooth.

$$\mathbf{L} \approx \nabla^2$$

Discretization of ∇^2

Finite difference in 1D

$$\nabla^2 u \approx \frac{1}{h^2}(-2\mathbf{u}_j + \mathbf{u}_{j-1} + \mathbf{u}_{j+1}).$$

Finite difference in 2D

$$\nabla^2 u \approx \frac{1}{h^2}(-4\mathbf{u}_{ij} + \mathbf{u}_{i-1j} + \mathbf{u}_{i+1j} + \mathbf{u}_{ij-1} + \mathbf{u}_{ij+1}).$$

Code in 1D

```
L1D = @(n,h) 1/h^2 *...
 spdiags(ones(n,1)  * [1  -2  1],-1:1,n,n)
```

Discretization of ∇^2

In 2D $\quad \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

Use Kroneker products

$$\mathrm{vec}(\mathbf{LUI}) = (\mathbf{I}^\top \otimes \mathbf{L})\mathrm{vec}(\mathbf{U}).$$

Code in 2D

```
L = kron(speye(n2), L1D(n1,h1)) + ...
      kron(L1D(n2,h2),speye(n1) );
```

More about discrete ∇^2

Note that $\mathbf{L}$ can also be written as a convolution

$$\mathbf{L} = \frac{1}{h^2} \;\; \mathbf{U} * \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

.
In general - any differential operator with constant coefficients can be written as convolution

Newton like methods - recap

$$\min_{\mathbf{W}} \; J(\mathbf{W}) = E(\mathbf{W}) + \alpha R(\mathbf{W})$$

Requires derivatives of the regularization

Tip: Use $\nabla^2 R$ as a preconditioner for the conjugate gradient solver in the Newton iteration.

Test problems

- Simple linear problem
- Circle
- Peaks
- Spiral
- MNIST
- CIFAR10

Problems
Test your codes on all 5 problems. What is the best accuracy you can get.
Which of the methods is the most effective?