Regularization for Image Classification

Numerical Methods for Deep Learning

Why use regularization?

We are attempting to train weights $\mathbf{W} \in \mathbb{R}^{n_c \times n_f}$ to express the relation between some data $\mathbf{Y} \in \mathbb{R}^{n_f \times n}$ and their labels $\mathbf{C} \in \mathbb{R}^{n_c \times n}$ by solving

$$\min_{\mathbf{W}} E(\mathbf{W}) = E(\mathbf{C}, \mathbf{W}, \mathbf{Y})$$

Recall: $\mathrm{rank}(\mathbf{Y}) \leq \min\{n_f, n\}$

- $n < n_f$: No unique solution
- $n > n_f$: $\mathbf{Y}$ may still be rank-deficient

Challenges in image classification:

- data is high dimensional ($n_f \approx$ number of pixels/voxels/frames)
- higher resolution $\rightsquigarrow$ need more examples?
- higher resolution $\rightsquigarrow$ larger rank?

Regularization

If Hessian $\nabla^2 E$ highly ill-conditioned, regularization is needed.

- Symptom: weights are large or oscillatory.
- Alternative: Estimate condition number (costly!)

Solution: require solution to be regular

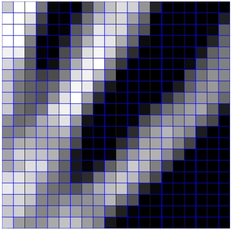
$$\min_{W} \ \phi(\mathbf{W}) = E(\mathbf{W}) + \lambda R(\mathbf{W}),$$

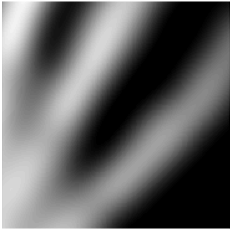
where

- R is a regularizer, $R(\mathbf{W})$ large when $\mathbf{W}$ is irregular and small otherwise
- λ is a regularization parameter (needs to be chosen)
- Mathematically: R makes sure $\mathbf{W}^*$ lies in desired function space (and is sufficiently *regular*).

Excellent references include [1, 2, 3].

What is an Image?





Digital images are arrays $\mathbf{U} = \mathbb{R}^{m_1 \times m_2 \times c}$ ($c = 1 \rightsquigarrow$ grey only).

- perhaps most common interpretation in image processing

Continuous point of view: Images are functions supported on a domain $\Omega \in \mathbb{R}^2$ $u : \Omega \to \mathbb{R}^c$.

- choose function space (e.g., continuous, differentiable)
- discretize on regular grids $\rightsquigarrow$ digital image
- apply operators to images (e.g., gradient in edge detection)

Type of Regularization

Classical Tikhonov (aka weight decay)

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

requires elements to be small.

When $\mathbf{Y}$ are images, also columns in $\mathbf{W}$ can be seen as images

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

General Tikhonov: Let $\mathbf{L}$ be a given matrix

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

If $\mathbf{L}$ is discrete derivative operator, entries need to be smooth.

Discretization of ∇^2

Idea: Ensure classifier is smooth by using $\mathbf{L} \approx \nabla^2$.

Finite difference in 1D: Let $\mathbf{u} \in \mathbb{R}^m$ be discretization of $u : [0,1] \to \mathbb{R}$ on regular grid with pixel size $h = 1/m$

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

Code in 1D

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

Finite difference in 2D: Let $\mathbf{U} \in \mathbb{R}^{m \times m}$ be discretization of $u : [0,1]^2 \to \mathbb{R}$ on regular grid with pixel size $h = 1/m$

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

Discretization of ∇^2

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

Use Kroneker products

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

Code in 2D

```
L = kron(speye(m2), L1D(m1,h1)) + ...
      kron(L1D(m2,h2),speye(m1) );
```

More about discrete ∇^2

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

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

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

Continuous interpretation allows re-computing a convolution kernel for different image resolutions.

Recap: Numerical Optimization

Require derivatives of the regularization to efficiently solve

$$\min_{\mathbf{W}} \ \phi(\mathbf{W}) = E(\mathbf{W}) + \lambda R(\mathbf{W})$$

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

Exercise: Setup smoothness regularizer and test it on MNIST and CIFAR-10

References

[1] P. C. Hansen. *Rank-deficient and discrete ill-posed problems*. SIAM Monographs on Mathematical Modeling and Computation. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998.

[2] P. C. Hansen. *Discrete inverse problems*, volume 7 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2010.

[3] C. R. Vogel. *Computational Methods for Inverse Problems*. SIAM, Philadelphia, 2002.