Main: prdpt, zerograd

Code: python blender, mitsuba

Higher order: conjugate gradient, newton, gauss newton?, quasi newton, higher order…

Variance reduction: mcmc

CG:

For sparse Ax=b

Residual -> steepest descent

Orthogonal residual r\_i and r\_(i+1)

Conjugate direction d0 d1 …

Use ri as ui…

Build CG, newton quasi newton…

What problems are these mainly for

Original:

图片包含 文本

描述已自动生成

Using pytorch lbfgs to try: Unstable, slow, …

Lbfgs after adam: 文本

中度可信度描述已自动生成

1. bfgs with 1st gradient
2. 2nd gradient methods with hessian.

For 17/06:

Notes:

1. Ax\_true = b

Error: e = x – x\_true

Residual: difference to b: r = b-Ax

So, r = -Ae

1. Steepest descent – direction is the residual, line search gets the lowest point where gradient is orthogonal to the line

图示

描述已自动生成文本, 信件

描述已自动生成

1. Conjugate directions:

This steepest descent often takes steps in the same direction (zigzag).

So pick a set of basis directions and only take one descent for each direction.

文本

描述已自动生成

First though use orthogonal directions di, so error is always orthogonal to the best point in the descent direction.

图示, 工程绘图

描述已自动生成

Cannot use orthogonal directions because it requires us to know x\_true to compute error e.

Instead use A orthogonal(conjugate): xTAx = 0.

So new requirement is e(i+1)T A di = 0.

Finding this is equivalent to finding minimum point in direction di:

文本, 信件

描述已自动生成

We can solve for alpha now:

文本, 信件

描述已自动生成(Same as steepest descent if di is residual ri)

So all we need is a set of A orthogonal search directions di:

(Gram-Schmidt conjugation)

Pick n linearly independent vectors ui, then di is ui minus its component that are not A-orthogonal to the previous u0,…ui-1.

黑色的钟表

中度可信度描述已自动生成, and beta is:

文本, 信件

描述已自动生成

图片包含 图示

描述已自动生成

Takes O(n^3) operations and need to keep previous direction in memory.

(equivalent to gaussian elimination when ui are the unit axis vectors).

The number of matrix-vector products per iteration can be reduced to one by using a recurrence to find the residual:

文本

描述已自动生成

1. Note:

The error term can also be seen as a linear combination of the directions and ei is the initial error e0 cutted out the previous directions:

白板上的文字

描述已自动生成

So new residual is orthogonal to the previous directions di:

文本

描述已自动生成

So, the residual is also orthogonal to the previous ui vectors that are used to construct di. (because ui and di span the same space)

不同形状的钟表

描述已自动生成 文本, 信件

描述已自动生成

In addition: (derived from the summation)

文本, 信件

描述已自动生成

1. Conjugate gradients:

Set the search direction di as conjugate of the residuals ri.

We can simply beta: 文本, 信件

描述已自动生成(ui is now ri), to:

图示, 文本

描述已自动生成

First line using r(i+1) = r(i) – a(i)Ad(i),

Second line to third using r(i)Tr(j) = 0 if i != j. (residuals are orthogonal to previous ones)

So all the other terms is not needed because when j < i-1 are all 0.

So betaij is now beta(i), and using the formula for alpha:

文本, 信件

描述已自动生成

, second line using d(i-1)r(i-1) = u(i-1)r(i-1) (sec 4) and u(i-1) is r(i-1).

In conclusion:

表格

中度可信度描述已自动生成

1. eigenvectors for A does not rotate when A is applied to. (Ax = cx)

So for iterative application of A, xi reaches 0 or infinity depending on c.

All vectors beside eigenvectors also does. (think of linear combination of the eigen vectors).

So important: spectral radius of A = max|eigenvalue| (greater than 1 or not).

1. Eigenvalue of pd matrix are all positive:

Ax = cx 🡪 xTAx = cxTx 🡪 xTAx > 0 by definition, xTx >0, so c >0.

1. Nonlinear conjugate gradient:
2. The residual is set to the negation of gradient, and search directions are computed using conjugate of residual as CG.

图示

描述已自动生成

1. However, alpha can not be easily calculated in the line search.

Calculate by setting new residual orthogonal to search direction.

文本, 信件

描述已自动生成

1. Many different ways to get beta:

文本

描述已自动生成

24/06:

1. Convolution of function f with kernel g evaluated at x: (flip g at y-axis, then slide)

(no flip is cross correlation, of course some kernel are symmetric at y-axis)

(the flip makes the convolution associative, meaning fgh = f(gh). )

Integral(inf) {f(x-t) g(t)} dt 🡪 each t in g should multiply x-t in f

=Integral(inf) {f(t) g(x-t)} dt 🡪 each t in f should multiply x-t in g (like moving the centre of g to x for convolution, to get the actual value of g to multiply f(t), the x-axis of g is the distance from x to t, x-t, because g is flipped for convolution.)

1. Derivative of convolution is convolution with derivative of kernel:

图示, 文本, 示意图

中度可信度描述已自动生成图形用户界面, 文本

描述已自动生成

For prdpt:

文本, 信件

描述已自动生成

To get the hessian of the convolved render,

Need to convolve with hessian of the gaussian.

文本

描述已自动生成

The calculation is done for all pairs of thetas in the hessian matrix [11,12; 21,22…]

For now, since the implemented backward should only get the incoming gradient of 1(since this is the last ‘layer’), and pytorch does not seem to support a hessian extension of the autograd.function, an extra function will be written to carry out the hessian computation only.

1. Variance reduction:
2. Finite difference for hessian:

Subscripts itself denotes scalar entry for i/j th element. Subscript +/- vector x means a vector with entry i/j.

First derivative:

1. Forward:
2. First derivative secant:

For faster calculations, , meaning that first derivative can be calculated at once.

1. Central:

Rearrange:

1. For CG: preconditioning and other ways to improve.
2. BFGS and Quasi-newton methods.
3. More ways for optimization:
4. Report :
5. Since smoothFn is already gradient in the forward pass, we need to calculate its second derivative.
6. Cannot compare to numdifftools hessian with all hessian calculations.

Can only calculate with first derivative secant

1. Successful with hessian epsilon=1e-3, tol=1e-7, NR\_tol=1e-5, recompte=10, with NR line search and FR beta update.

01/07:

Meeting points:

For the simple problem:

1. gradient 2. Blurred gradient 3. Analytical hessian 4. Fd hessian

For the box example or harder problem:

For the Michael gif problem.

Plot the variance against the number of samples

Expected value of newtons method update

Powell methods, and other methods to approximate to get differentiable hessian inverse

Basically study details of hessian in normal optimization, rendering optimization, blurred, non-blurred optimization, and others

Things to test:

For an example problem where the cost function is known and the hessian can be derived:

Using ground truth f’’

Using diagonal of f’’

Using finite difference

Using secant method to approximate

Using second derivative of gaussian

Missed:

Why do I need the whole hessian for line search?

general convergence of steepest descent: sec6.2

Convergence analysis of cg: sec 9

Preconditioning

Variance reduction

Using FR vs PR

Other ways to approximate hessian

2nd order for noisy functions

Learn bfgs and nonlinear cg

P10, toy problem, no blur, all starting point, all parameters