Chapter 1

Image Restoration Using GMRF's

The previous chapters have presented various models such as the 2D AR model and the GMRF, but so far we have not seen how these models can be used to solve imaging problems. In this chapter, we show how these models can be used in imaging applications via an example of image restoration. Although our example is somewhat simplified from a practical perspective, it will serve as a basis for presenting and developing basic methods that are useful in a wide variety of imaging problems.

Let $X \in \Re^N$ be an N pixel image that we would like to measure, and let $Y \in \Re^N$ be the noisy measurements given by

$$Y = AX + W \tag{1.1}$$

where A is a nonsingular $N \times N$ matrix, and W is a vector of i.i.d. Gaussian noise with $N(0, \sigma^2)$.

This type of problem occurs commonly in imaging applications where Y consists of noisy and blurred scanner measurements and and X is the restored image that we would like to recover. For example, the matrix A may represent the blurring operation of a linear space-invariant filter. So we have that

$$y_s = \sum_{r \in S} a_{s-r} x_r + w_s$$

where S is the set of all pixel locations. In this case, the x and y are typically chosen to be the respective images in raster order, and A is a Toeplize-block-Toeplitz matrix. In fact, with a little generalization, the model of (1.1) can be used to represent important problems in image reconstruction that occur in applications such as tomographic reconstruction or transmission or emission tomography, or even astronomical imaging.

Using the model of (1.1), the conditional distribution of Y given X is

$$\begin{aligned} p_{Y|X}(y|x) &= \prod_{s\in S} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2\sigma^2}(y_s - \sum_{r\in S} A_{sr}x_s)^2\right\} \\ &= \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{s\in S}(y_s - \sum_{r\in S} A_{sr}x_s)^2\right\} \\ &= \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left\{-\frac{1}{2\sigma^2}||y - Ax||^2\right\} \end{aligned} \tag{1.2}$$

where S is the set of pixel indices for the image.

1.1 Maximum Likelihood and Bayesian Estimators

One approach to estimation of X is to use the maximum likelihood (ML) estimator. In this case, X is assumed to be a deterministic quantity, and the ML estimate is given by

$$\begin{aligned} \hat{x}_{ML} &= \arg\max_{x\in\Re^N} \log p_{Y|X}(y|x) \\ &= \arg\max_{x\in\Re^N} \left\{-\frac{1}{2\sigma^2}||y - Ax||^2 - \frac{N}{2}\log(2\pi\sigma^2)\right\} \\ &= \arg\min_{x\in\Re^N} ||y - Ax||^2 \\ &= A^{-1}y \ . \end{aligned}$$

However, this solution has a number of problems associated with it. First, if A has very small eigenvalues, the inversion of A maybe unstable. This can happen if the blurring filter transfer function is nearly zero at certain frequencies. In this case, the matrix inversion corresponds to applying a very high amplification of these suppressed frequencies in order to restore them. However, even if $A = I$, the identity matrix, the ML estimate is unsatisfying since it says that the best restortion of a noisy image is to simply accept the measured noise with no further processing. Intuitively, we might expect that some type of smoothing or other processing might be able to reduce the noise without excessively degrading the image detail.

An alternative approach is to assume that X is random to use the Bayesian estimation framework. The approach has the advantage that one can incorporate knowledge about the probable behavior of X; however, its disadvantage

is that it requires an accurate model be selected that captures the characteristics of X. For this example, we will use a zero mean homogeneous Gaussian Markov random field (GMRF) model for X. In this case, X is a zero-mean jointly Gaussian random process, so its distribution has the form

$$p_X(x) = \frac{1}{(2\pi\sigma_x^2)^{N/2}}|B|^{1/2} \exp\left\{-\frac{1}{2\sigma_x^2}x^t B x\right\} \tag{1.3}$$

where B is an invertible symmetric matrix. Since X is a homogeneous GMRF, the entrees of B must have the form

$$B_{i,j} = \delta_{i-j} - g_{i-j}$$

where g_{i-j} is the non-causal FIR prediction filter associated with the GMRF. Using this assumption, we can compute the posterior distribution $p_{X|Y}(x|y)$. From Bayes rule, we know that

$$\log p_{X|Y}(x|y) = \log p_{Y|X}(y|x) + \log p_X(x) - \log p_Y(y) \ .$$

However, this expression is difficult to evaluate directly because calculation of the term $p_Y(y)$ requires the evaluation of a difficult integral. Therefore, we will take a different tact, and simply evaluate $\log p_{X|Y}(x|y)$ as a function of x, ignoring any dependence on y. This means that our solution will be correct within a unknown multiplicative constant, but we can compute that constant because we know that $p_{X|Y}(x|y)$ must integrate to 1. So using the expressions of (1.2) and (1.3), we have that

$$\log p_{X|Y}(x|y) = -\frac{1}{2\sigma^2}||y - Ax||^2 - \frac{1}{2\sigma_x^2}x^t B x + c(y) \tag{1.4}$$

where $c(y)$ is some complex function of y. Since this is a quadratic function of x, we can express this function in the form

$$\log p_{X|Y}(x|y) = -\frac{1}{2}(x-\mu)^t R^{-1}(x-\mu) + c'(y) \tag{1.5}$$

where $c'(y)$ is a new function of y and μ and R are given by

$$\begin{aligned} \mu(y) &= \left(A^t A + \frac{\sigma^2}{\sigma_x^2}B\right)^{-1} A^t y \\ R &= \left(\frac{1}{\sigma^2}A^t A + \frac{1}{\sigma_x^2}B\right)^{-1} \ . \end{aligned}$$

The correctness of (1.5) is easily verified by taking the first and second gradients (i.e. the Hessian) of each expression with respect to x, and showing that they are equal. From this, we see that the expression for the posterior distribution has the form

$$p_{X|Y}(x|y) = \frac{1}{z(y)} \exp\left\{-\frac{1}{2}(x-\mu(y))^t R^{-1}(x-\mu(y))\right\}$$

where $z(y)$ is an unknown function of y. However, since we know that $\int p_{X|Y}(x|y)dx = 1$, we know that the normalizing constant must be same one used for any multivariate Gaussian distribution. So we have the final result that

$$p_{X|Y}(x|y) = \frac{1}{(2\pi)^{N/2}}|R|^{-1/2} \exp\left\{-\frac{1}{2}(x-\mu(y))^t R^{-1}(x-\mu(y))\right\} \ , \qquad (1.6)$$

and we see that the conditional distribution of X given Y is normal with mean $\mu(y)$ and covariance R.

Using the posterior distribution of (1.6), we can compute some typical Bayesian estimators. Notice that the MMSE estimator is given by the conditional mean,

$$\hat{x}_{MMSE} = E\left[X|Y=y\right] = \mu(y)$$

and the maximum a posterior estimate is given by the conditional mode

$$\begin{aligned}
\hat{x}_{MAP} &= \arg\max_{x\in\Re^N} \log p_{X|Y}(x|y) && (1.7)\\
&= \arg\max_{x\in\Re^N} \left\{-\frac{1}{2}(x-\mu(y))^t R^{-1}(x-\mu(y))\right\} && (1.8)\\
&= \arg\min_{x\in\Re^N} \left\{\frac{1}{2}(x-\mu(y))^t R^{-1}(x-\mu(y))\right\} && (1.9)\\
&= \mu(y) && (1.10)
\end{aligned}$$

So for this case, the MMSE and MAP estimates are the same. In fact, this is always the case when all the random quantities in an estimation problem are jointly Gaussian. However, we will find that the MMSE and MAP are generally not the same when the distributions are non-Gaussian. This will be important later since we will see that non-Gaussian distributions are better models of image features such as edges.

Add a figure showing why the MAP and MSEE estimates are the same for Gaussian distributions.**

So it seems that our problem is solved with the ubiquitous estimate

$$\hat{x} = \left(A^t A + \lambda B\right)^{-1} A^t y \tag{1.11}$$

where we define $\lambda = \frac{\sigma^2}{\sigma_x^2}$. However, while the solution of (1.11) is mathematically appealing it is not very practical for most applications because the dimension of the matrix to be inverted is enormous. For example, if N equals 1 million pixels, then the matrix to be inverted has 10^{12} elements. If each element is stored with a 64 bit float, then the matrix requires approximately 8 terra bytes of storage! Clearly we will need another approach.

1.2 Optimization for Computing the MAP Estimate

In order to make the computation of the MAP estimate more tractable, we need to go back to the first principles of its computation. The MAP estimate of X given Y given by

$$\begin{aligned}
\hat{x}_{MAP} &= \arg\max_{x\in\Re^N} p_{X|Y}(x|y) \\
&= \arg\max_{x\in\Re^N} \left\{\log p_{X|Y}(x|y)\right\} \\
&= \arg\max_{x\in\Re^N} \left\{\log p_{Y|X}(y|x) + \log p_X(x) - \log p_Y(y)\right\} \\
&= \arg\min_{x\in\Re^N} \left\{-\log p_{Y|X}(y|x) - \log p_X(x)\right\} \ .
\end{aligned} \tag{1.12}$$

Substituting the expressions from (1.2) and (1.3) into (1.12), yields the equations

$$\begin{aligned}
\hat{x}_{MAP} &= \arg\min_{x\in\Re^N} \left\{\frac{1}{2\sigma^2}||y - Ax||^2 + \frac{N}{2}\log(2\pi\sigma^2)\right. \\
&\qquad \left. + \frac{1}{2\sigma_x^2}x^t Bx + \frac{N}{2}\log(2\pi\sigma_x^2|B|^{-1/N})\right\} \\
&= \arg\min_{x\in\Re^N} \left\{\frac{1}{2\sigma^2}||y - Ax||^2 + \frac{1}{2\sigma_x^2}x^t Bx\right\}
\end{aligned}$$

So we have the final expression for the MAP estimate in terms of an explicit optimization

$$\hat{x}_{MAP} = \arg\min_{x\in\Re^N} \left\{||y - Ax||^2 + \lambda x^t Bx\right\} \tag{1.13}$$

where $\lambda = \sigma^2/\sigma_x^2$.

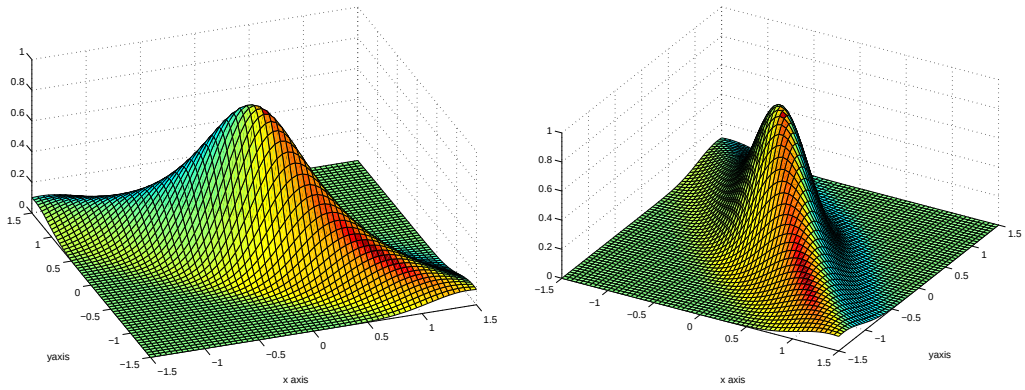


Figure 1.1: 3D rendering of the example non-quadratic cost function from two different views. The analytical form of the function is $\frac{1}{(x-y)^2+1}\exp(-4(x+y)^2)$.

In equation (1.13), we see that, in this case, MAP estimation is equivalent to optimization of a quadratic function. The key insight is that while the direct inversion of (1.11) provides an exact mathematical solution, it requires an enormous amount of computation. Alternatively, numerical optimization methods can provide an approximate solution to the optimization of (1.13) that can be computed much less expensively, but at the cost of numerical approximation. Ironically, the "exact" solution of (1.11) can actually yield numerically less precise solutions than optimization methods because of the cumulative effects of round-off error.

Appendix A presents some basic facts about convex functions and their use in optimization. Using the results of this appendix, we can see that the cost function to be optimized

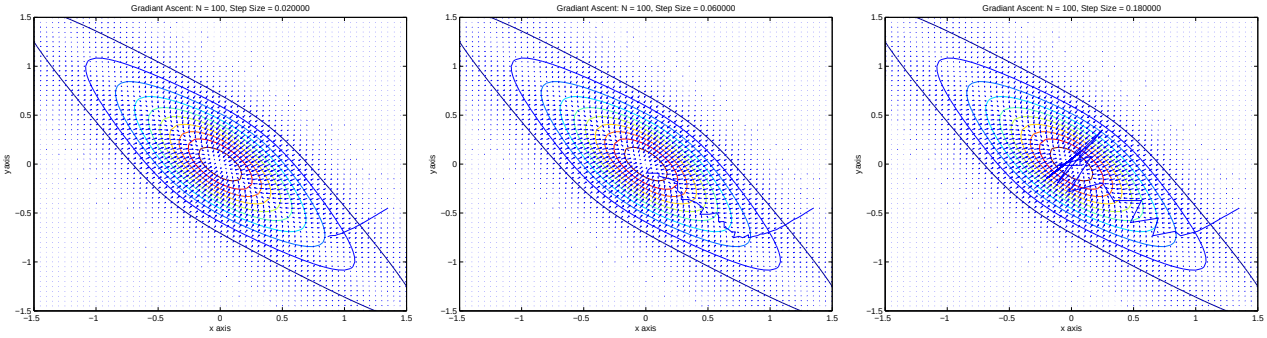
$$f(x) = ||y - Ax||^2 + \lambda x^t B x$$

is strictly convex and has a unique global minimum at $\mu(y)$ which is a solution to the equation

$$\nabla f(x)\big|_{x=\hat{x}} = 0 \ .$$

1.3 Gradient Decent Optimization

A natural approach to optimization of a continuously differentiable function is to start at an initial value $x^{(0)}$, and then incrementally move in the opposite direction of the gradient. Intuitively, the gradient is the direction of the functions most rapid increase, so one would expect moving in the opposite



(a) (b) (c)

Figure 1.2: 2D contour plots of the trajectory of gradient decent optimization using a step size of a) $\alpha = 0.02$; b) $\alpha = 0.06$; c) $\alpha = 0.18$. Notice that the convergence with small α is slow, and the convergence with large α is unstable.

direction to be an effective method of minimizing the function. This approach to optimization is generally referred to as **gradient decent** optimization or **Jacobi** iterations, and it is commonly used in a variety of applications.

The general form of the gradient decent iterations is given by

$$x^{(k+1)} \;=\; x^{(k)} - \alpha[\nabla f(x^{(k)})]^t$$

where $x^{(k)}$ is the current image estimate, $x^{(k+1)}$ is the updated image, and α is a user selected step size. For the image restoration problem of (1.13), the gradient of $f(x)$ is given by

$$\nabla f(x) = -2(y - Ax)^t A + 2\lambda x^t B \;;$$

so the gradient decent update equation is given by

$$x^{(k+1)} = x^{(k)} + \alpha[A^t(y - Ax^{(k)}) - \lambda B x^{(k)}] \tag{1.14}$$

where α is scaled by $1/2$. Rearranging terms, we have the recursion

$$x^{(k+1)} = (I - \alpha[A^t A + \lambda B])x^{(k)} + \alpha A^t y \;. \tag{1.15}$$

An important special case of (1.15) occurs when A represents a linear space-invariant filter a_s. When A is a space-invariant filter, then it is Toeplitz-block-Toeplitz; the operation Ax is equivalent to 2D convolution $a_s * x_s$; and multiplication by the transpose $A^t y$ is equivalent to 2D convolution with the space-reversed impulse response $a_{-s} * y_s$. In this case, the gradient decent update equation of (1.14) has the form

$$x_s^{(k+1)} = x_s^{(k)} + \alpha[a_{-s} * (y_s - a_s * x_s^{(k)}) + \lambda(\delta_s - g_s)x_s^{(k)}] \;. \tag{1.16}$$

Here you can see that each iteration of gradient decent is computed through the application of LSI filters, so as long as the filter impulse response is spatially limited, the computation is not excessive.

1.3.1 Convergence Analysis of Gradient Decent

Gradient decent is a simple algorithm, but a key disadvantage is related to the difficulty in selection of the step-size parameter. Fig. 1.1 shows an example of a 2D non-quadratic function to be maximized, which is equivalent to the minimization of the function's negative. Notice that the peak of the function is narrow along one dimension and wide along another. Fig. 1.2 shows the effect of the step-size selection. Too large of a step size results in the unstable convergence shown in Fig. 1.2(c), whereas too small of a step size results in the slow convergence of Fig. 1.2(c). Unfortunately, for high dimensional optimization problems such as (1.13), we will see that their usually is no choice of α which is effective for both high and low spatial frequencies.

To better understand why this is true, we can analyze the convergence behavior of the gradient decent algorithm. Let us assume that the iteration of (1.15) converges to the limit $x^{(\infty)}$, then we can define the error in the calculation of the MAP estimate to be $\epsilon^{(k)} = x^{(k)} - x^{(\infty)}$. In this case, we know that taking the limit of (1.15) results in

$$x^{(\infty)} = (I - \alpha[A^t A + \lambda B])x^{(\infty)} + \alpha A^t y \ . \tag{1.17}$$

Subtracting (1.15) and (1.17) results in

$$\epsilon^{(k+1)} = \left(I - \alpha[A^t A + \lambda B]\right) \epsilon^{(k)} \ . \tag{1.18}$$

We can further simply the structure of (1.18) by using eigenvector analysis of the matrix

$$H \stackrel{\triangle}{=} A^t A + \lambda B \ .$$

Since H is symmetric, we know it can be diagonalized as $H = E\Lambda E^t$ where columns of E are the eigenvectors of H, and $\Lambda = \text{diag}\{h_1, \cdots, h_N\}$ is a diagonal matrix containing the corresponding N eigenvalues of H. Notice that the eigenvalues, h_i, are all strictly positive because the both matrices are positive definite, and $\lambda > 0$. If we define, the transformed error $\tilde{\epsilon}^{(k)} = E^t \epsilon^{(k)}$, then we may then derive the following simple relationship for the decay of

the error in the gradient decent method.

$$\tilde{\epsilon}^{(k)} = (I - \alpha\Lambda)^k \, \tilde{\epsilon}^{(0)} \ . \tag{1.19}$$

Since Λ is diagonal, we can see that the i^{th} component of the error then decays as

$$\tilde{\epsilon}_i^{(k)} = (1 - \alpha h_i)^k \, \tilde{\epsilon}_i^{(0)} \ . \tag{1.20}$$

In order for the convergence of (1.20) to be stable, we must have that $|1 - \alpha h_i| < 1$ for each component i. From this, and the fact that $h_i > 0$, we derive the following stability condition,

$$\alpha < \frac{2}{\max_i\{h_i\}} \ , \tag{1.21}$$

which results in the following bound

$$\max_i\{1 - \alpha h_i\} \geq 1 - \alpha \min_i\{h_i\} > 1 - 2\frac{\min_i\{h_i\}}{\max_i\{h_i\}}$$

The quantity $\kappa(H) = \frac{\max_i\{h_i\}}{\min_i\{h_i\}}$ is known as the conditioning number of the matrix H. The conditioning number is valuable because it quantifies the difficulty of inverting a matrix. Remember, the exact solution to our problem is expressed by the equation (1.11) as $\hat{x} = H^{-1}A^t y$; so our optimization problem is really equivalent to the inversion of H. Using this concept, we can lower bound the rate of convergence of (1.20) as

$$\max_i \left\{ \frac{\tilde{\epsilon}_i^{(k)}}{\tilde{\epsilon}_i^{(0)}} \right\} \ > \ \left(1 - \frac{2}{\kappa(H)} \right)^k \ .$$

In many applications, the conditioning number of H can be very large. In this case, the value $1 - 1/\kappa$ is very close to 1, and the convergence of (1.20) is very slow. The basic problem is that it is not possible to choose α so that both modes corresponding to both the large and small eigenvalues of H converge quickly. In general, the modes corresponding to small eigenvalues tend to converge quickly, and if one try to increase the speed of convergence by increasing the size of α, then the convergence can become unstable.

1.3.2 Frequency Analysis of Gradient Decent

The general convergence analysis of gradient decent depends on the eigenvalues of the matrix H. However, in real applications it may be difficult to

(a) (b)

Figure 1.3: 1d plots showing the frequency response of a) $h(\omega)$ and the resulting function b) $p(\omega)$ which determines the convergence rate of gradient decent for a step-size with good low frequency convergence. Values of $p(\omega)$ close to 1 result in slow convergence, so it is clear that for this problem gradient decent has slow high frequency convergence. In fact, this tends to be a general property of gradient decent.

determine these eigenvalues. An important special case occurs when A represents a LSI filter, and in this case a more complete analysis is possible in terms of the frequency response of the update iteration. Ignoring the finite boundaries of the image, we can take the DSFT of (1.16) and rearrange terms to yield

$$x^{(k+1)}(\omega) = \left(1 - \alpha \left[|a(\omega)|^2 + \lambda(1 - g(\omega))\right]\right) x^{(k)}(\omega) + \alpha a^*(\omega) y(\omega) \qquad (1.22)$$

If we further define $\epsilon^{(k)}(\omega)$ to be the DSFT of $\epsilon^{(k)} = x^{(k)} - \hat{x}$, then we can derived the following recursion for the error in the computation of the MAP estimate.

$$\epsilon^{(k+1)}(\omega) = \left(1 - \alpha \left[|a(\omega)|^2 + \lambda(1 - g(\omega))\right]\right) \epsilon^{(k)}(\omega) \qquad (1.23)$$

Using the definition that

$$h_s \stackrel{\triangle}{=} |a(\omega)|^2 + \lambda(1 - g(\omega)) \ ,$$

we have that

$$\epsilon^{(k)}(\omega) = (1 - \alpha h(\omega))^k \, \epsilon^{(0)}(\omega) \ .$$

So here we can see that the values of the frequency transform $h(\omega)$ are essentially the eigenvalues of the tranformation H when the transforms are all LSI. Furthermore, we know that $h(\omega) > 0$ because due to the fact that $1 - g(\omega) > 0$. Therefore, we can derive equivalent conditions for the stable selection of the update parameter

$$\alpha < \frac{2}{\max_\omega\{h(\omega)\}} \ , \qquad (1.24)$$

and we see that the slowest mode has convergence bounded below by

$$\max_\omega \left\{ \frac{\tilde{\epsilon}(\omega)^{(k)}}{\tilde{\epsilon}_\omega^{(0)}} \right\} \ > \ \left(1 - \frac{2}{\kappa(h)}\right)^k \ .$$

where

$$\kappa(h) = \frac{\max_\omega h(\omega)}{\min_\omega h(\omega)} \ .$$

Perhaps here the interpretation of the conditioning number, $\kappa(h)$, is more clear. A larger conditioning number means that the ratio of the maximum and minimum gain in the filter $h(\omega)$ is large. This typically occurs when the blurring filter has strong attenuation of high frequencies, but it can even occur when there is not blurring.

To better understand this, consider the 1D example when $a(\omega) = 1$, $g_n = \frac{1}{2}\left[\delta_{n-1} + \delta_{n+1}\right]$, and $\lambda = 10$. In this case, there is no blurring and the noise variance is large computed to the variation in x. From this we can calculate that $g(\omega) = 1 - \cos(\omega)$, and $h(\omega) = 1 + 10(1 - \cos(\omega))$, and the maximum and minimum eigenvalues occur at $\omega = \pi$ and $\omega = 0$ respectively, and have the values $h_{max} = 21$ and $h_{min} = 1$. This means that in order to get stable convergence at low frequencies we must choose $\alpha < 2/h_{max}$. In fact, for the best low frequency convergence, we need to choose $\alpha = 1/h_{max} = 1/21$, so we have

$$\epsilon^{(k)}(\omega) = \left(1 - (1 + 10(1 - \cos(\omega))/21)\right)^k \epsilon^{(0)}(\omega) \ .$$

The resulting frequency response of $h(\omega)$ and

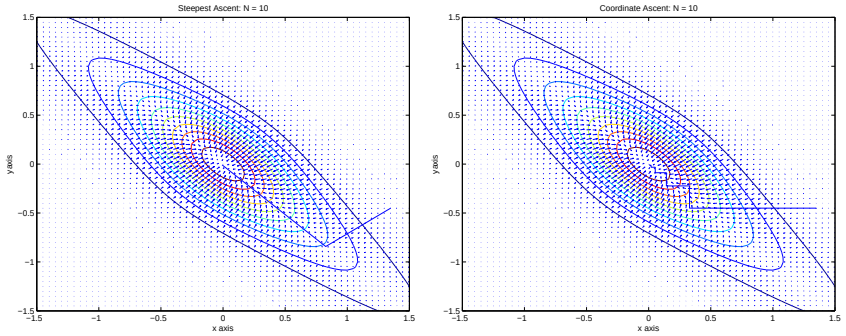
$$p(\omega) = (1 - (1 + 10(1 - \cos(\omega))/21))$$

are shown in Fig. 1.3.

When the function $p(\omega)$ is close to 1, the convergence of gradient decent is slow. Therefore, we can see that for this example, the convergence is slow at the high frequencies. In fact, this tends to be a general property of gradient decent. When solving problem in 2D, this means that the low frequency shading and color tend to converge quickly, but the high frequency detail, such as edges, converge slowly. This is a serious problem in imaging problems because typically edge content is particularly important in the perceived quality of an image.

1.4 Steepest Decent Optimization

One partial solution to the problem of selecting a step size is to choose the value of α at each iteration that minimizes the cost function. This algorithm



(a) (b)

Figure 1.4: 2D contour plots of the trajectory of a) steepest decent optimization and b) iterative coordinate decent optimization. Neither of these algorithms require the choice of a step parameter α.

is called **steepest decent**.

The general form of the steepest decent optimization is given by

$$\begin{aligned} d^{(k)} &= \left[-\nabla f(x^{(k)})\right]^t && (1.25) \\ \alpha^{(k)} &= \arg\min_{\alpha \geq 0} f(x^{(k)} + \alpha d^{(k)}) && (1.26) \\ x^{(k+1)} &= x^{(k)} + \alpha^{(k)} d^{(k)} \ . && (1.27) \end{aligned}$$

So in steepest decent optimization, each gradient step is taken with the step size that achieves the best minimum in that step. This computation of the step size in (1.27) requires a minimization of a 1D function and is known as a **line search**. Typically, it is possible to solve the line search by solving the following equation.

$$\frac{\partial f(x + \alpha d)}{\partial \alpha} = \nabla f(x + \alpha d) d = 0 \qquad (1.28)$$

In some cases, the line search required for steepest decent can be computationally intensive, but for the quadratic optimization we will see that the line search can be computed in closed form with modest computation.

For the image restoration problem of (1.13), the step direction is given by

$$d^{(k)} = 2A^t(y - Ax^{(k)}) - 2\lambda B x^{(k)} \ ; \qquad (1.29)$$

so using this gradient expression in (1.28) results in

$$\alpha^{(k)} = \frac{||d^{(k)}||^2}{2||d^{(k)}||_H^2} \qquad (1.30)$$

where $H = A^t A + \lambda B$, and the notation $||z||_W^2 = z^t W z$ for a positive definite matrix W.

Figure 1.4(a) illustrates the results of steepest decent optimization on the example function. Notice that the convergence is guarrented to be stable because with each iteration the cost function is monotone decreasing.

$$f(x^{(k+1)}) \leq f(x^{(k)})$$

If the cost function $f(x)$ is bounded below, then we know that $\lim_{k\to\infty} f(x^{(k)})$ converges to a limit.

1.5 Coordinate Decent Optimization

As we saw in Section 1.3.2, gradient decent has typically has slow convergence at high spatial frequencies. In practice, most gradient based methods tend to converge faster at low spatial frequencies for many important problems. One method to enhance convergence of high spatial frequencies is the **iterative coordinate decent** (ICD) algorithm, which is also known as the *Gauss-Seidel* algorithm. ICD works by updating each pixel in sequence while keeping the remaining pixels fixed. One full iteration of ICD then consists of a single update for each pixel.

The ICD algorithm is most easily specified using the assignment notation of a programming language. Using this notation, the value of the pixel x_s is updated using the rule

$$x_s \leftarrow \arg\max_{x_s \in \Re} f(x)$$

while the remaining pixels x_r for $r \neq s$ remain unchanged. The result of an ICD update of the pixel s can be compactly expressed as $x + \alpha e_s$ where e_s is a vector that is 1 for element s, and zero otherwise. Using this notation, we have that

$$x_s \leftarrow \arg\max_{\alpha \in \Re} f(x + \alpha e_s) \ .$$

For the image restoration problem of (1.13), the ICD update can be computed by using the quadratic form of the cost function. From this, we have that

$$0 \ = \ \frac{\partial}{\partial \alpha} f(x + \alpha e_s)$$

$$\begin{aligned} &= -2(y - A(x + \alpha e_s))^t A e_s + 2\lambda (x + \alpha e_s)^t B e_s \\ &= -2(y - Ax)^t A e_s + 2\lambda x B e_s + 2\alpha e_s^t (A^t A + \lambda B) e_s \\ &= -2(y - Ax)^t A_{*,s} + 2\lambda x B_{*,s} + 2\alpha [A^t A + \lambda B]_{s,s} \end{aligned}$$

where $A_{*,s}$ represents the s^{th} column of A. Solving for α yields

$$\alpha = \frac{(y - Ax)^t A_{*,s} - \lambda x^t B_{*,s}}{[A^t A + \lambda B]_{s,s}} \ , \tag{1.31}$$

which in turn results in the ICD update equation

$$x_s \leftarrow x_s + \frac{(y - Ax)^t A_{*,s} - \lambda x^t B_{*,s}}{[A^t A + \lambda B]_{s,s}} \ . \tag{1.32}$$

As with gradient decent, an important special case of (1.32) occurs when A represents a linear space-invariant filter a_s. In this case, multiplication by A is equivalent to convolution with a_s; multiplication by A^t is equivalent to convolution with a_{-s}; and the ICD update has the form

$$x_s \leftarrow x_s + \frac{\sum_l a_{l-s} * (y_l - \sum_r a_{l-r} * x_r) - \lambda (x_s - \sum_r g_{s-r} x_r)}{\sum_s a_s + \lambda} \ . \tag{1.33}$$

When the point spread function is $a_s = \delta_s$, then there is not blurring, and the update equation has a simpler form which lends insight into the process.

$$x_s \leftarrow \frac{y_s + \lambda \sum_r g_{s-r} x_r}{1 + \lambda}$$

Notice that in this case, the ICD update is a weighted balance between the measurement, y_s, and the average of the neighboring pixels, $\sum_r g_{s-r} x_r$. When λ is large and the signal-to-noise is low, then the update is more similar to the pixel's neighbors; however, when λ is small and the signal-to-noise is high, then the update is more similar measured data.

1.5.1 Convergence Analysis of Coordinate Decent

1.5.2 Frequency Analysis of Coordinate Decent

1.6 Preconditioned Conjugate Gradient Optimization

Appendix A

Convex Functions

Look at homework problems for items to cover.

Bibliography