8 Advanced gradient schemes

In Chapter 2 we introduced two canonical approaches to unconstrained minimization, namely, gradient descent and Newton's method. In the current chapter we add to that discussion by fully describing two popular step length rules, both of which provide mathematically provable convergence to a stationary point for the gradient descent algorithm. We then describe *stochastic* (or *iterative*) *gradient descent*, an important extension of the original gradient descent scheme that helps scale the algorithm to very large datasets.

8.1 Fixed step length rules for gradient descent

In the following two sections we discuss two of the most popular ways of automatically determining proper step lengths for each step of a gradient descent run, which we refer to as *step length rules*. In particular we discuss two commonly used rules which guarantee, mathematically speaking, convergence of the gradient descent algorithm to a stationary point*: fixed* and *adaptive* step length rules each of which has practical strengths and weaknesses. While typically providing a conservative (i.e., small) step length that is kept fixed for all iterations, the fixed step length rule discussed first provides both a convenient choice for many of the cost functions described in this book, as well as a benchmark by which to easily test larger fixed values. On the other hand, with the adaptive rule discussed second we adaptively compute the step length at each gradient descent step by using local information from the part of the cost function near the current step. This typically produces larger steps in practice than a fixed rule, meaning fewer steps are necessary for convergence, but the determination of each step requires computation.

8.1.1 Gradient descent and simple quadratic surrogates

Recall from Section 2.2.4 how the second order Taylor series expansion of a cost function g centered at a point $\mathbf{w}^0$,

$$g\left(\mathbf{w}^0\right) + \nabla g\left(\mathbf{w}^0\right)^T\left(\mathbf{w} - \mathbf{w}^0\right) + \frac{1}{2}\left(\mathbf{w} - \mathbf{w}^0\right)^T \nabla^2 g\left(\mathbf{w}^0\right)\left(\mathbf{w} - \mathbf{w}^0\right), \tag{8.1}$$

leads to a well-defined descent step known as Newton's method. This is indeed the most natural quadratic approximation to g at $\mathbf{w}^0$ that is available to us. However, as detailed

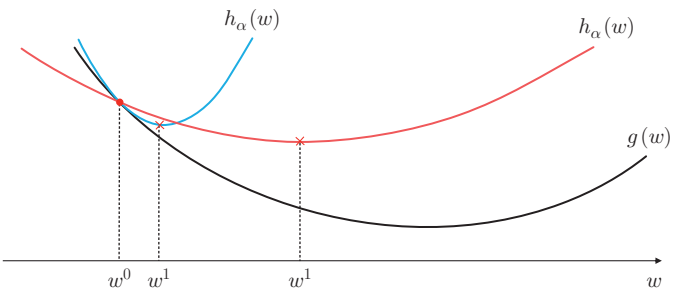

Fig. 8.1 Two quadratic functions approximating the function g around $\mathbf{w}^0$ given by (8.2). The value of α is larger with the red quadratic than with the blue one.

in that section, there are potential difficulties in storing and even calculating the Hessian matrix $\nabla^2 g\left(\mathbf{w}^0\right)$ for large scale problems. Still the idea of minimizing a function g by "hopping down" the stationary points of quadratic approximations (also referred to as *surrogates*), as opposed to the linear approximations/surrogates as employed by gradient descent, is a pleasing one with great intuitive appeal. So a natural question is: can we replace the Hessian with a simpler quadratic term and produce an effective descent algorithm?

For example, we may consider the following simple quadratic function:

$$h_\alpha\left(\mathbf{w}\right) = g\left(\mathbf{w}^0\right) + \nabla g\left(\mathbf{w}^0\right)^T\left(\mathbf{w} - \mathbf{w}^0\right) + \frac{1}{2\alpha}\left\|\mathbf{w} - \mathbf{w}^0\right\|_2^2, \tag{8.2}$$

where $\alpha > 0$. This is just the second order Taylor series of g around $\mathbf{w}^0$ where we have replaced the Hessian $\nabla^2 g\left(\mathbf{w}^0\right)$ with the simple diagonal matrix $\frac{1}{\alpha}\mathbf{I}_{N\times N}$. This kind of quadratic is illustrated in Fig. 8.1 for two values of α. Note that the larger the α the wider the associated quadratic becomes. Also, when $\mathbf{w} = \mathbf{w}^0$ the last two terms on the right hand side of (8.2) disappear and we have $h_\alpha\left(\mathbf{w}^0\right) = g\left(\mathbf{w}^0\right)$.

Our simple quadratic surrogate h_α has a unique global minimum which may be found by checking the first order condition (see Section 2.1.2) by setting its gradient to zero,

$$\nabla h_\alpha\left(\mathbf{w}\right) = \nabla g\left(\mathbf{w}^0\right) + \frac{1}{\alpha}\left(\mathbf{w} - \mathbf{w}^0\right) = \mathbf{0}, \tag{8.3}$$

and solving for $\mathbf{w}$. Doing this we can compute the minimizer of h_α, which we call $\mathbf{w}^1$, as

$$\mathbf{w}^1 = \mathbf{w}^0 - \alpha\nabla g\left(\mathbf{w}^0\right). \tag{8.4}$$

Note the minimizer of the quadratic in Equation (8.2) is precisely a gradient descent step at $\mathbf{w}^0$ with a step length of α. Therefore our attempt at replacing the Hessian with a very simple quadratic, and locating the minimum of that quadratic, does not lead to a new descent method but to the familiar gradient descent step. If we continue taking steps in this manner the kth update is found as the minimum of the simple quadratic surrogate associated with the previous update $\mathbf{w}^{k-1}$,

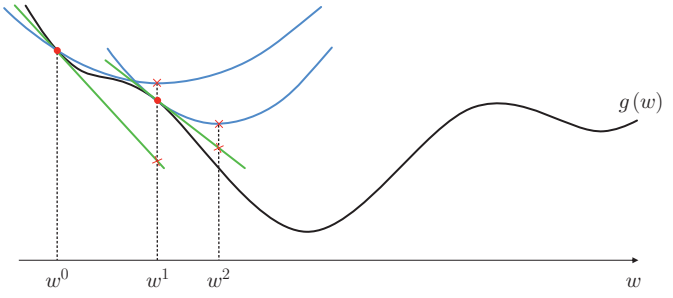

Fig. 8.2 Gradient descent can be viewed simultaneously as using either linear or simple quadratic surrogates to find a stationary point of g. At each step the associated step length defines both how far along the linear surrogate we move before hopping back onto the function g, and at the same time the width of the simple quadratic surrogate which we minimize to reach the same point on g.

$$h_\alpha(\mathbf{w}) = g\left(\mathbf{w}^{k-1}\right) + \nabla g\left(\mathbf{w}^{k-1}\right)^T\left(\mathbf{w} - \mathbf{w}^{k-1}\right) + \frac{1}{2\alpha}\left\|\mathbf{w} - \mathbf{w}^{k-1}\right\|_2^2, \tag{8.5}$$

where the minimum is given as the kth gradient descent step

$$\mathbf{w}^k = \mathbf{w}^{k-1} - \alpha\nabla g\left(\mathbf{w}^{k-1}\right). \tag{8.6}$$

Therefore, as illustrated in Fig. 8.2, we can interpret gradient descent as a method that uses linear surrogates or simultaneously one that uses simple fixed curvature quadratic surrogates to locate a stationary point of g. The chosen step length at the kth iteration then determines how far along the linear surrogate we move, or equivalently the width of the quadratic we minimize, in order to reach the next point on g.

Using the simple quadratic perspective of gradient descent we can naturally wonder if, akin to the operation of Newton's method (see e.g., Fig. 2.11), for a given cost function g we can design a step length rule such that we can "hop down" the minima of the associated quadratic surrogates to reach a stationary point of g. As we describe in the remainder of this and the next section, we absolutely can.

8.1.2 Functions with bounded curvature and optimally conservative step length rules

What would happen if we chose a small enough step length α so that the curvature of the associated quadratic surrogate, which would be fixed at each step of gradient descent, matched the greatest curvature of the underlying cost function itself? As illustrated in Fig. 8.3, this would force not only the minimum of each quadratic surrogate to lie *above* the cost function, but the entire quadratic surrogate itself.[1] While this is a conservative choice of step length (and by conservative, we mean small) we refer to it as "optimally conservative" because we can actually compute the maximum curvature of every cost

[1] It is easy to show that the simple quadratic surrogate with $\alpha = \frac{1}{L}$, where L is defined as in Equation (8.50), around $\mathbf{w}^{k-1}$ given by

$$h_{\frac{1}{L}}(\mathbf{w}) = g\left(\mathbf{w}^{k-1}\right) + \nabla g\left(\mathbf{w}^{k-1}\right)^T\left(\mathbf{w} - \mathbf{w}^{k-1}\right) + \frac{L}{2}\left\|\mathbf{w} - \mathbf{w}^{k-1}\right\|_2^2, \tag{8.7}$$

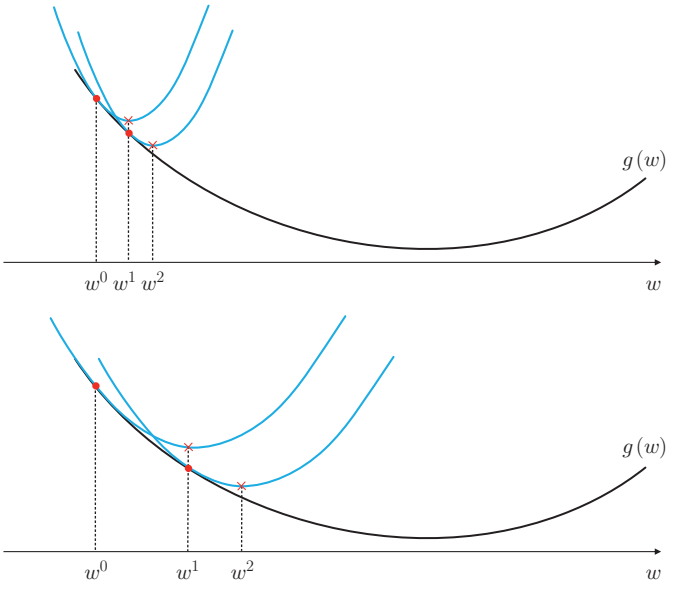

Fig. 8.3 (top panel) Too conservative a fixed step length leads to smaller descent steps. (bottom panel) Another conservative fixed step length where the curvature of its associated quadratic just matches the greatest curvature of the hypothetical cost function while still lying entirely above the function. Such a step length is referred to as "optimally conservative." Note that while the underlying cost here is drawn convex this applies to non-convex cost functions as well, whose greatest curvature could be negative, i.e., on a concave part of the function.

function described in this book (or a reasonable approximation of it) analytically. Therefore this choice of step length can be very convenient in practice and, moreover, using this step length the gradient descent procedure is guaranteed (mathematically speaking) to converge to a stationary point of g (see Section 8.4.1 for further details).

To define this step length formally, recall from Section 2.1.3 that the curvature of a function g is encoded in its *second derivative* when g takes in scalar input w, and more generally its matrix of second derivates or *Hessian* when g takes in vector valued input $\mathbf{w}$. More formally, if g has globally bounded curvature then there must exist an $L > 0$ that bounds its second derivative above and below in the case of a scalar input function

indeed lies completely above the function g at all points as shown in Fig. 8.3. Writing out the *first order Taylor's formula* for g centered at $\mathbf{w}^{k-1}$, we have

$$g(\mathbf{w}) = g\left(\mathbf{w}^{k-1}\right) + \nabla g\left(\mathbf{w}^{k-1}\right)^T\left(\mathbf{w} - \mathbf{w}^{k-1}\right) + \frac{1}{2}\left(\mathbf{w} - \mathbf{w}^{k-1}\right)^T \nabla^2 g(\mathbf{c})\left(\mathbf{w} - \mathbf{w}^{k-1}\right), \tag{8.8}$$

where $\mathbf{c}$ is a point on the line segment connecting $\mathbf{w}$ and $\mathbf{w}^{k-1}$. Since $\nabla^2 g \preceq L\mathbf{I}_{N\times N}$ we can bound the right hand side of (8.8) by replacing $\nabla^2 g(\mathbf{c})$ with $L\mathbf{I}_{N\times N}$, giving

$$g(\mathbf{w}) \leq g\left(\mathbf{w}^{k-1}\right) + \nabla g\left(\mathbf{w}^{k-1}\right)^T\left(\mathbf{w} - \mathbf{w}^{k-1}\right) + \frac{L}{2}\|\mathbf{w} - \mathbf{w}^{k-1}\|_2^2 = h_{\frac{1}{L}}(\mathbf{w}), \tag{8.9}$$

which indeed holds for all $\mathbf{w}$.

Table 8.1 Common cost functions and corresponding Lipschitz constants for each cost where the optimally conservative fixed step length rule is given as $\alpha = \frac{1}{L}$. Note that the regularizer can be added to any cost function in the middle column and the resulting Lipschitz constant is the sum of the two Lipschitz constants listed here.

Cost function	Form of cost function	Lipschitz constant
Least Squares regression	$\sum_{p=1}^{P}\left(\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}} - y_p\right)^2$	$L = 2\left\|\tilde{\mathbf{X}}\right\|_2^2$
Softmax cost/logistic regression	$\sum_{p=1}^{P}\log\left(1 + e^{-y_p\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}}}\right)$	$L = \frac{1}{4}\left\|\tilde{\mathbf{X}}\right\|_2^2$
Squared margin/soft-margin SVMs	$\sum_{p=1}^{P}\max^2\left(0, 1 - y_p\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}}\right)$	$L = 2\left\|\tilde{\mathbf{X}}\right\|_2^2$
Multiclass softmax	$\sum_{c=1}^{C}\sum_{p\in\Omega_c}\log\left(1 + \sum_{\substack{j=1\\ j\neq c}}^{C} e^{\tilde{\mathbf{x}}_p^T\left(\tilde{\mathbf{w}}_j - \tilde{\mathbf{w}}_c\right)}\right)$	$L = \frac{C}{4}\left\|\tilde{\mathbf{X}}\right\|_2^2$
ℓ_2-regularizer	$\lambda\left\|\mathbf{w}\right\|_2^2$	$L = 2\lambda$

$$-L \leq \frac{\partial^2}{\partial w^2}g\left(w\right) \leq L, \tag{8.10}$$

or bounds the eigenvalues of its Hessian in the general case of vector input

$$-L\mathbf{I}_{N\times N} \preceq \nabla^2 g\left(\mathbf{w}\right) \preceq L\mathbf{I}_{N\times N}. \tag{8.11}$$

For square symmetric matrices $\mathbf{A}$ and $\mathbf{B}$ the notation $\mathbf{A} \preceq \mathbf{B}$ is shorthand for saying that each eigenvalue of $\mathbf{A}$ is smaller than or equal to the corresponding eigenvalue of $\mathbf{B}$. When described in this mathematical way, functions satisfying the above for finite values of L are said to have "bounded curvature" or equivalently to have a "Lipschitz continuous gradient[2]" with Lipschitz constant L.

As mentioned, all of the cost functions discussed in this book have computable maximum curvature (or some estimation of it) including the Least Squares costs, the squared margin hinge and soft-margin SVM costs, as well as two-class and multiclass soft-margin perceptrons. For convenience, in Table 8.1 we provide a complete list of Lipschitz constants for these cost functions (one can find associated calculations producing these constants in Section 8.5). Note here[3] that we write each cost function using the compact vector notation commonly used throughout the book (see e.g., Examples 4.1 and 4.2). Also recall that the notation $\left\|\tilde{\mathbf{X}}\right\|_2^2$ refers to the so-called "spectral norm" of

[2] In rare instances where g is only once differentiable but not twice (e.g., for the squared margin cost), it is said to have a Lipschitz continuous gradient if $\frac{\left\|\nabla g(\mathbf{w}) - \nabla g(\mathbf{v})\right\|_2}{\left\|\mathbf{w}-\mathbf{v}\right\|_2} \leq L$ for any $\mathbf{v}$ and $\mathbf{w}$ in its domain.

[3] Also note that the results shown here can be easily generalized to pair with fixed basis feature transformations, but while cost functions paired with (deep) neural network features also typically have bounded curvature, explicitly computing Lipschitz constants for them becomes very challenging as the number of layers increases due to the difficulty in gradient/Hessian computations (as one can see by noting the difficulty in simply computing the gradient in such instances, as in Section 7.2). Therefore the Lipschitz constants reported here do not extend to the use of multilayer network basis features.

the matrix $\tilde{\mathbf{X}}$ and denotes the largest eigenvalue of $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$ (which is always the largest eigenvalue of $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$ as well).

8.1.3 How to use the conservative fixed step length rule

The conservatively optimal Lipschitz constant step length rule will always work "right out of the box" to produce a convergent gradient descent scheme, therefore it can be a very convenient rule to use in practice. However, as its name implies and as described previously, it is indeed a conservative rule by nature.[4]

Therefore in practice, if one has the resources, one should use the rule as a benchmark to search for larger convergence-forcing fixed step length rules. In other words, with the Lipschitz constant step length $\alpha = \frac{1}{L}$ calculated one can easily test larger step lengths of the form $\alpha = t \cdot \frac{1}{L}$ for any constant $t > 1$.

> The conservatively optimal step length rule is convenient both because it works "right out of the box," and because it provides a benchmark for trying larger fixed step length values in practice.

Depending on both the cost function and dataset, values of t ranging from 1 to large values like 100 can work well in practice. For convenience we give the complete gradient descent algorithm with this sort of fixed step length in Algorithm 8.1.

Algorithm 8.1 Gradient descent with fixed step length based on a conservatively optimal fixed base.

Input: function g with Lipschitz continuous gradient, and initial point $\mathbf{w}^0$
$k = 1$
Find the smallest L such that $-L\mathbf{I} \preceq \nabla^2 g(\mathbf{w}) \preceq L\mathbf{I}$ for all $\mathbf{w}$ in the domain of g
Choose a constant $t \geq 1$
Set $\alpha = t \cdot \frac{1}{L}$
Repeat until stopping condition is met:
    $\mathbf{w}^k = \mathbf{w}^{k-1} - \alpha \nabla g\left(\mathbf{w}^{k-1}\right)$
    $k \leftarrow k + 1$

Example 8.1 Conservative fixed rate base comparisons

In Fig. 8.4 we show the result of employing several fixed step length rules using the conservatively optimal Lipschitz base for minimizing the softmax cost/logistic regression,

[4] Moreover, in many instances the Lipschitz constants shown in Table 8.1 are themselves conservative estimates of the true maximum curvature of a cost function (that is, a *larger* than ideal estimate) due to necessary mathematical inequalities involved in their derivation (see Section 8.5 for details).

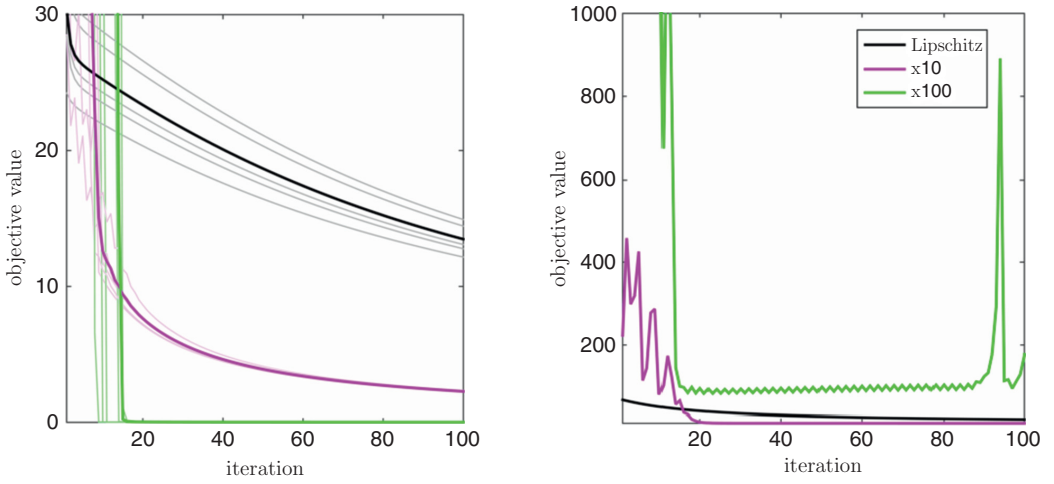

Fig. 8.4 The objective value resulting from the first 100 iterations of gradient descent applied to minimizing the softmax cost over two simple two-class datasets (see text for further details). Three constant step size rules were employed, with five runs for each (shown in lighter colors) as well as their average (shown in darker versions of the matching color): the gradient Lipschitz constant $\alpha = \frac{1}{L}$ (guaranteed to force convergence, shown in black) was used as a base, along with $\alpha = 10 \cdot \frac{1}{L}$ and $\alpha = 100 \cdot \frac{1}{L}$ (shown in magenta and green respectively). For the first dataset (left panel) both of the larger step lengths produce faster convergence than the base, with the largest providing extremely rapid convergence for this dataset. With the second dataset (right panel) the medium step length produces the best result, with the largest step length producing a divergent sequence of gradient steps.

employing both two-class datasets of $P = 100$ points each first shown in Fig. 4.3. In particular, gradient descent was run using three fixed step length rules: the Lipschitz step length $\alpha = \frac{1}{L}$ (where L is as shown in Table 8.1), as well as $\alpha = 10 \cdot \frac{1}{L}$ and $\alpha = 100 \cdot \frac{1}{L}$. Shown in the figure (left panel) is the objective or cost function value for the first 100 iterations of five runs with each step length (shown in light black, magenta, and green respectively), as well as their average shown in bolder versions of the colors.

With the first dataset (left panel), both fixed step length rules larger than the Lipschitz base produce more rapid convergence, with the step length 100 times that of the Lipschitz base producing extremely rapid convergence. Conversely, with the second dataset (right panel), only the medium step length rule produces more rapid convergence than the Lipschitz base, with the 100 times rate producing a divergent sequence of steps.

8.2 Adaptive step length rules for gradient descent

We have just seen how gradient descent can be thought of as a geometric minimization technique that, akin to Newton's method, works by hopping down the global minima of simple quadratic surrogates towards a function's minimum. In this section we continue

this geometric intuition in order to develop a commonly used adaptive step length rule for gradient descent, which is a convenient and well-performing alternative to the fixed step length rule previously described.

8.2.1 Adaptive step length rule via backtracking line search

Using the quadratic surrogate perspective of gradient descent, we can now construct a very simple yet powerful and generally applicable method for adaptively determining the appropriate step length for gradient descent at each iteration. Recall that in the previous section we saw how the kth gradient descent step is given as the global minimizer of the simple quadratic surrogate h_α given in Equation (8.5). Note that if α is chosen in a way that the minimum of h_α lies above $g\left(\mathbf{w}^k\right)$ we have, using the definition of h_α and plugging in $\mathbf{w}^k = \mathbf{w}^{k-1} - \alpha\nabla g\left(\mathbf{w}^{k-1}\right)$ for $\mathbf{w}$,

$$g\left(\mathbf{w}^k\right) \leq g\left(\mathbf{w}^{k-1}\right) + \nabla g\left(\mathbf{w}^{k-1}\right)^T\left(\mathbf{w}^k - \mathbf{w}^{k-1}\right) + \frac{1}{2\alpha}\left\|\mathbf{w}^k - \mathbf{w}^{k-1}\right\|_2^2. \tag{8.12}$$

Simplifying[5] the right hand side gives

$$g\left(\mathbf{w}^k\right) \leq g\left(\mathbf{w}^{k-1}\right) - \frac{\alpha}{2}\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2. \tag{8.13}$$

Note that as long as we have not reached a minimum of g, the term $\frac{\alpha}{2}\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2$ is always positive and we have descent at each step $g\left(\mathbf{w}^k\right) < g\left(\mathbf{w}^{k-1}\right)$. While this conclusion was based on our assumption that the global minimum of h_α lay above g, we can in fact conclude the converse as well. That is, if the inequality in (8.13) holds for an $\alpha > 0$ then the minimum of the associated quadratic h_α lies above g, and the related gradient descent step decreases the objective value, i.e., $g\left(\mathbf{w}^k\right) < g\left(\mathbf{w}^{k-1}\right)$.

Therefore the inequality in (8.13) can be used as a tool, referred to as *backtracking line search*, for finding an appropriate step length α at each step in performing gradient descent (which leads to a provably convergent sequence of gradient steps to a stationary point of g, see Section 8.4.2 for further details). That is, we can test a range of decreasing values for the learning rate until we find one that satisfies the inequality, or equivalently a simple quadratic surrogate whose minimum lies above the corresponding point on g, as illustrated in Fig. 8.5.

One common way of performing this search is to initialize a step length $\alpha > 0$ and check that the desired inequality,

$$g\left(\mathbf{w}^{k-1} - \alpha\nabla g\left(\mathbf{w}^{k-1}\right)\right) \leq g\left(\mathbf{w}^{k-1}\right) - \frac{\alpha}{2}\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2, \tag{8.14}$$

holds. If it does not, then we multiply α by some number $t \in (0, 1)$, set $\alpha \longleftarrow t\alpha$, and try again until the inequality is satisfied. Note that the larger t is set the more fine grained

[5] Making the substitution $\mathbf{w}^k = \mathbf{w}^{k-1} - \alpha\nabla g\left(\mathbf{w}^{k-1}\right)$ the right hand side becomes $g\left(\mathbf{w}^{k-1}\right)$ $-\alpha\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2 + \frac{\alpha}{2}\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2 = g\left(\mathbf{w}^{k-1}\right) - \frac{\alpha}{2}\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2$.

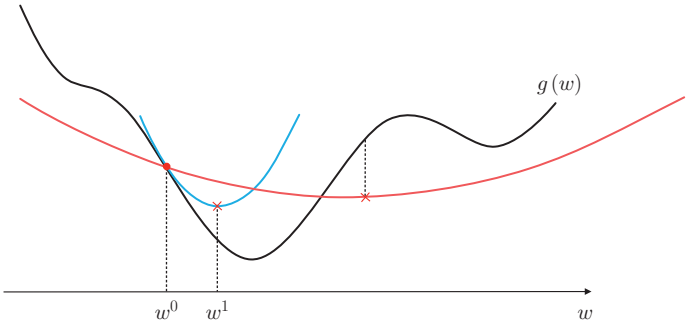

Fig. 8.5 A geometric illustration of backtracking line search. We begin with a relatively large initial guess for the step length, which generates the larger red quadratic, whose minimum may not lie above g. The guess is then adjusted downwards until the minimum of the associated quadratic (in blue) lies above the function.

the search will be. Also the terms $g\left(\mathbf{w}^{k-1}\right)$ and $\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2$ in (8.14) need only be computed a single time, making the procedure very efficient, and the same initial α and t can be used at each gradient descent step.

Furthermore, this sequence can be shown to be mathematically convergent to a stationary point of g, as detailed in Section 8.4.2. For convenience the backtracking line search rule is summarized in Algorithm 8.2.

Algorithm 8.2 Gradient descent with backtracking line search

Input: starting point $\mathbf{w}^0$, damping factor $t \in (0, 1)$, and initial $\alpha > 0$

$k = 1$

Repeat until stopping condition is met:

$\alpha_k = \alpha$

While $g\left(\mathbf{w}^{k-1} - \alpha_k \nabla g\left(\mathbf{w}^{k-1}\right)\right) > g\left(\mathbf{w}^{k-1}\right) - \frac{\alpha_k}{2}\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2$

$\alpha_k \longleftarrow t\alpha_k$

End while

$\mathbf{w}^k = \mathbf{w}^{k-1} - \alpha_k \nabla g\left(\mathbf{w}^{k-1}\right)$

$k \leftarrow k + 1$

8.2.2 How to use the adaptive step length rule

Like the optimally conservative fixed step length, backtracking line search is a convenient rule for determining step lengths at each iteration of gradient descent that works right out of the box. Furthermore, because each step length is determined using the local curvature of g, the backtracking step length will typically be superior (i.e., larger) than that of the conservative fixed step length described in Section 8.1. However each gradient step using backtracking line search, compared to using the fixed step length rule, typically includes higher computational cost due to the search for a proper step length.

The adaptive step length rule works right out of the box, and tends to produce large step lengths at each iteration. However, each step length must be actively computed.

Due to this tradeoff it is difficult to judge universally which rule (conservative or adaptive) works best in practice, and both are widely used. The choice of diminishing parameter $t \in (0, 1)$ in Algorithm 8.2 provides a tradeoff between computation and step size with backtracking line search. The larger the diminishing parameter is set the more careful is the search for each step length (leading to more computation) but the larger will be the final step length chosen, while the converse holds for smaller values of t.

Example 8.2 Simple comparison of adaptive and optimal conservative step length rules

In Fig. 8.6 we show the result of 100 iterations of gradient descent using the backtracking line search step size rule as well as the optimally conservative step length rule discussed in Section 8.1. The dataset used here is a two-class dataset consisting of $P = 10\,000$ points (see Example 8.4 for further details), and the cost function used

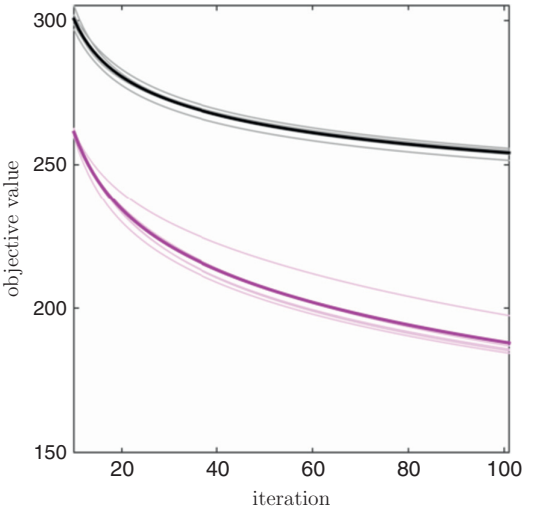

Fig. 8.6 The convergence of gradient descent using conservative fixed and backtracking line search rules on a two class classification dataset (see text for further details). Shown here are the objective values per iteration for each rule, with runs using the backtracking line search and fixed step lengths shown in magenta and black respectively (lighter colored curves indicate values over a single run of gradient descent, with the two darker colored curves showing the respective average value over five runs). As expected the backtracking runs typically display greater decrease per iteration than runs employing the fixed step length rule.

was the softmax cost. Gradient descent was run five times with each step length rule, and the objective value per iteration is shown in the figure (with the backtracking runs in magenta and the runs with fixed step length in black). As expected the adaptive backtracking line search rule, due to its larger individual step lengths, leads to more rapid decrease in the cost function value per iteration.

8.3 Stochastic gradient descent

As the size of a dataset grows, storing it in active memory in order to just compute a gradient becomes challenging if not impossible, making the standard gradient descent scheme particularly ineffective in practice for large datasets. In this section we introduce an extension of gradient descent, known as *stochastic* (or *iterative*) *gradient descent* (see footnote 8), which not only completely overcomes the memory issue but, for large datasets, is also significantly more effective computationally speaking in practice than the standard gradient method. In particular, the stochastic gradient descent provides one of the best ways of dealing with the large datasets often employed in image/audio/text-based learning tasks when paired with both non-convex cost functions (e.g., any regression/classification cost employing a neural network feature map), as well as convex costs where storage is an issue or when the feature space of a dataset is too large to employ Newton's method.

Throughout the section we will discuss minimization of cost functions over a dataset of P points $\left\{\left(\mathbf{x}_p, y_p\right)\right\}_{p=1}^{P}$, where the y_p are continuous in the case of regression, or $y_p \in \{-1, 1\}$ in the case of two class classification (although the method described here can also be applied to multiclass classification as well), and for simplicity we will make use of the compact optimization notation $\tilde{\mathbf{x}}_p = \begin{bmatrix} 1 \\ \mathbf{x}_p \end{bmatrix}$ $\tilde{\mathbf{w}} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}$ introduced in e.g., Examples 4.1 and 4.2.

Finally, note that in what follows one may replace each input data point with any M-dimensional fixed or neural network feature vector (as in Chapters 5 and 6) with no adjustment to the general ideas described here.

8.3.1 Decomposing the gradient

As we have seen, the cost function of a predictive model is written as a sum of individual costs over each data point as

$$g\left(\tilde{\mathbf{w}}\right) = \sum_{p=1}^{P} h\left(\tilde{\mathbf{w}}, \tilde{\mathbf{x}}_p\right). \tag{8.15}$$

For example, the Least Squares regression and softmax perceptron costs each take this form as

$$g\left(\tilde{\mathbf{w}}\right) = \sum_{p=1}^{P} \left(\tilde{\mathbf{x}}_p^T \tilde{\mathbf{w}} - y_p\right)^2, \qquad g\left(\tilde{\mathbf{w}}\right) = \sum_{p=1}^{P} \log\left(1 + e^{-y_p \tilde{\mathbf{x}}_p^T \tilde{\mathbf{w}}}\right), \tag{8.16}$$

where $h\left(\tilde{\mathbf{w}}, \tilde{\mathbf{x}}_p\right) = \left(\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}} - y_p\right)^2$ and $h\left(\tilde{\mathbf{w}}, \tilde{\mathbf{x}}_p\right) = \log\left(1 + e^{-y_p\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}}}\right)$ respectively. Because of this, when we minimize a predictive modeling cost via gradient descent, the gradient itself is a summation of the gradients of each of the P summands. For example, the gradient of the softmax cost may be written as

$$\nabla g\left(\tilde{\mathbf{w}}\right) = \sum_{p=1}^{P} \nabla h\left(\tilde{\mathbf{w}}, \tilde{\mathbf{x}}_p\right) = -\sum_{p=1}^{P} \sigma\left(-y_p\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}}\right) y_p\tilde{\mathbf{x}}_p, \tag{8.17}$$

where in this instance $\nabla h\left(\tilde{\mathbf{w}}, \tilde{\mathbf{x}}_p\right) = -\sigma\left(-y_p\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}}\right) y_p\tilde{\mathbf{x}}_p$ with $\sigma(\cdot)$ being the logistic sigmoid function (as first defined in Section 3.3.1). When minimizing any predictive modeling cost via gradient descent we can therefore think about the kth gradient descent step in terms of these individual gradients as

$$\tilde{\mathbf{w}}^k = \tilde{\mathbf{w}}^{k-1} - \alpha_k \nabla g\left(\tilde{\mathbf{w}}^{k-1}\right) = \tilde{\mathbf{w}}^{k-1} - \alpha_k \sum_{p=1}^{P} \nabla h\left(\tilde{\mathbf{w}}^{k-1}, \tilde{\mathbf{x}}_p\right), \tag{8.18}$$

where α_k is an appropriately chosen step length such as those discussed in the previous sections. As described in the introduction to this section, for large datasets/values of P this gradient can be difficult or even impossible to produce given memory limitations.

Given this memory issue and the fact that the gradient decomposes over each data point, it is natural to ask if, in place of a single gradient step over the entire dataset, whether or not we can instead take a sequence of P shorter gradient steps in each data point individually. In other words, instead of taking a single full gradient step as in Equation (8.18), at the kth iteration of the procedure, will taking P smaller gradient steps in each data point similarly lead to a properly convergent method (that is, a method convergent to a stationary point of the cost function $g\left(\tilde{\mathbf{w}}\right)$)? If this were the case then we could resolve the memory problem discussed in the introduction to this section, as data would need only to be loaded into active memory a single point at a time.

Indeed with the appropriate choice of step length this procedure, called stochastic gradient descent, is provably convergent (for a formal proof see Section 8.4.3).

8.3.2 The stochastic gradient descent iteration

More formally, the analog of the kth iteration of the full gradient scheme shown in Equation (8.18) consists of P sequential point-wise gradient steps written as

$$\tilde{\mathbf{w}}^{k,p} = \tilde{\mathbf{w}}^{k,p-1} - \alpha_k \nabla h\left(\tilde{\mathbf{w}}^{k,p-1}, \tilde{\mathbf{x}}_p\right) \qquad p = 1 \ldots P. \tag{8.19}$$

In analogy with the kth full gradient step, here we have used the double superscript $\tilde{\mathbf{w}}^{k,p}$ which reads "the pth individual gradient step of the kth stochastic gradient descent iteration." In this notation the initial point of the kth iteration is written as $\tilde{\mathbf{w}}^{k,0}$, the corresponding sequence of P individual gradient steps as $\left\{\tilde{\mathbf{w}}^{k,1}, \tilde{\mathbf{w}}^{k,2} \ldots, \tilde{\mathbf{w}}^{k,P}\right\}$, and the final output of the kth iteration (i.e., the Pth stochastic step) as $\tilde{\mathbf{w}}^{k,P} = \tilde{\mathbf{w}}^{k+1,0}$. After

completing the kth iteration we perform the $(k+1)$th iteration by cycling through the data in precisely the same order, taking individual gradient steps for $p = 1 \dots P$.

To reaffirm the vocabulary being used here, with the standard gradient descent we use "step" and "iteration" interchangeably, i.e., each iteration consists of one full gradient step in all P data points simultaneously as shown in Equation (8.18). Conversely, with the stochastic method we refer to a single "iteration" as consisting of all P individual gradient steps, one in each data point, executed sequentially for $p = 1 \dots P$ as shown in Equation (8.19).

Example 8.3 Comparing stochastic and standard gradient descent on a simulated dataset

As an example, in Fig. 8.7 we show the result of applying 25 iterations of the standard gradient descent method with a fixed conservative step length (discussed in Section 8.1) shown in black, and an adaptively chosen one at each iteration (discussed in Section 8.2) shown in magenta, as well as the stochastic gradient descent method shown in green (we discuss the choice of step length for the iterative gradient method in the next section).

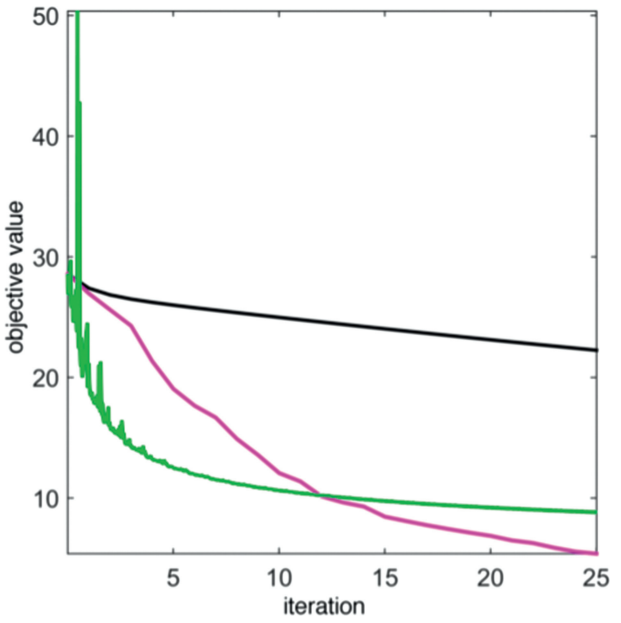

Fig. 8.7 The objective value of the first 25 iterations of stochastic gradient descent (shown in green), compared with standard gradient descent with conservative fixed step length (black) and adaptively chosen step length (magenta) (all three algorithms used the same initialization). Each iteration of the stochastic gradient method consists of P sequential gradient steps, one in each of the data points, as shown in Equation (8.19). Note how the stochastic method converges rapidly in the first few iterations, outperforming both standard gradient runs, when far from a point of convergence. See text for further details.

The softmax cost function is minimized here in order to perform logistic regression on the dataset[6] of $P = 100$ points first shown in the left panel of Fig. 4.3.

In Fig. 8.7 we show the objective value at all iterations of the algorithm. For the stochastic gradient method in each case we show these values for each step of the algorithm over all 25 of its iterations, for a total of 2500 individual steps (since $P = 100$ and 25 iterations were performed). Interestingly, we can see that while the stochastic gradient method has roughly the same computational cost as the standard gradient method, it actually outperforms the fixed step length run and, at least for the first 12 iterations, the adaptive run as well.

8.3.3 The value of stochastic gradient descent

The result of the previous example is indicative of a major computational advantage of stochastic gradient descent: when far from a point of convergence the stochastic method tends in practice to progress much faster towards a solution compared to standard gradient descent schemes. Because moderately accurate solutions (provided by a moderate amount of minimization of a cost function) tend to perform reasonably well in machine learning applications, and because with large datasets a random initialization will tend to lie far from a convergent point, substantial *empirical* evidence (see e.g., [18, 21] and references therein) suggests that stochastic gradient descent is often far more effective in practice (than standard gradient descent) for dealing with large datasets.[7] In many such instances even a single iteration (i.e., one gradient step through each point of the dataset) can provide a good solution.

Stochastic gradient descent tends to work extremely well with large datasets.

Example 8.4 Stochastic gradient descent performed on a large dataset

In Fig. 8.8 we show the result of 100 iterations of gradient descent on a $P = 10\,000$ two class classification dataset on which the softmax cost was used to perform logistic regression. This data was generated for the task of face detection (see Example 1.4)

[6] When applying stochastic gradient descent it is common practice to first randomize the order of the data prior to running the algorithm. In practice this has been found to improve convergence of the method, and is done with the data in this example.

[7] Note that due to the manner in which MATLAB/Octave currently performs loops an implementation of stochastic gradient descent can run slowly. However, in other programming languages like C++ or Python this is not a problem.

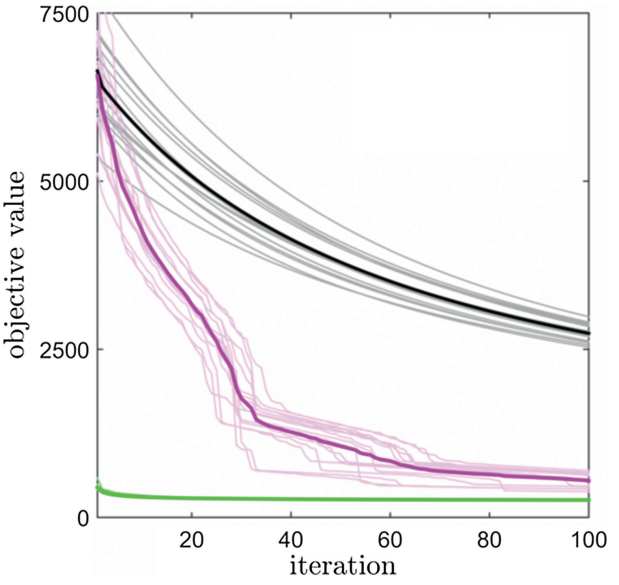

Fig. 8.8 The objective value of the first 100 iterations of stochastic gradient descent (shown in green), compared with standard gradient descent with conservatively optimal fixed step length (black) and adaptively chosen step length (magenta). In this instance a real two class classification dataset is used consisting of $P = 10\,000$ points (see text for further details). 15 runs of each method are shown as the lighter colored curves, with their averages shown in bold. Here the stochastic gradient descent scheme is massively superior to both standard gradient methods in terms of the rapidity of its convergence.

and consists of 3000 facial images, the remainder consisting of examples of non-face images.

Shown in the figure is the objective or cost function value of two runs of the standard gradient descent scheme, the first using the conservatively optimal fixed step length and the second using the adaptive rule, along with the corresponding cost function value of the stochastic gradient procedure. Here the results of 15 runs of each method are shown in lighter colors, where in each instance a shared random initialization is used by all three methods, and the average over all runs of each method is highlighted as a darker curve. We can see that the stochastic gradient descent scheme is massively superior on this large dataset in terms of the rapidity of its convergence when compared to the standard gradient method.

8.3.4 Step length rules for stochastic gradient descent

By slightly extending the convergence-forcing mechanism used in determining a step size rule for standard gradient descent one can conclude that a *diminishing* step size can similarly guarantee mathematically the convergence of the stochastic gradient method (see Section 8.4.3 for a formal derivation). More precisely, a step size rule satisfying

the following two requirements is guaranteed to cause the stochastic gradient descent procedure to converge to a stationary point:

(1) The step size must diminish as the number of iterations increases: $\alpha_k \longrightarrow 0$ as $k \longrightarrow \infty$.

(2) The sum of the step sizes is not finite: i.e., $\sum_{k=1}^{\infty} \alpha_k = \infty$.

Common choices of step size in practice with the iterative gradient method include $\alpha_k = \frac{1}{k}$ and $\alpha_k = \frac{1}{\sqrt{k}}$, or variations of these (see e.g., [21] for further information about how to choose particular variations of these step lengths in practice). The former of these rules, $\alpha_k = \frac{1}{k}$, was used in both examples shown in Fig. 8.7 and 8.8.

8.3.5 How to use the stochastic gradient method in practice

Even though a diminishing step length mathematically ensures convergence, like the optimal conservative fixed step length rules discussed for standard gradient descent in Section 8.1, the diminishing step length rule for the stochastic gradient method is *conservative in nature*. Again as with the standard gradient this does not reduce the utility of the diminishing step length rule, it is indeed very useful as it can always be counted on to work right out of the box in practice, and it is therefore commonly used.

However, be aware that in practice one may successfully use other step length rules such as fixed step lengths that, while they do not ensure convergence theoretically, work very well in practice. For example, fixed step lengths (tuned properly on a given dataset) are also commonly used in practice with the stochastic gradient method.

Example 8.5 Comparison of diminishing versus (tuned) fixed step lengths on a large dataset

In Fig. 8.9 we show several runs of the stochastic gradient method to minimize the softmax cost for the dataset of $P = 10\,000$ two-class data points first described in Example 8.4. In particular, we compare the result of $k = 100$ total iterations using three distinct step-size rules: two diminishing step sizes, $\alpha_k = \frac{1}{k}$ and $\alpha_k = \frac{1}{\sqrt{k}}$, and the constant step size of $\alpha_k = \frac{1}{3}$ for all k. We run stochastic gradient descent with each step size rule five times each, in each instance providing the same random initialization to each version of the algorithm. As can be seen in the figure, where we show the objective value at each iteration of the stochastic method for runs of all three step length choices, the run with $\alpha_k = \frac{1}{\sqrt{k}}$ provides better performance than $\alpha_k = \frac{1}{k}$, but the constant step size $\alpha_k = \frac{1}{3}$ runs outperform both diminishing rules overall.

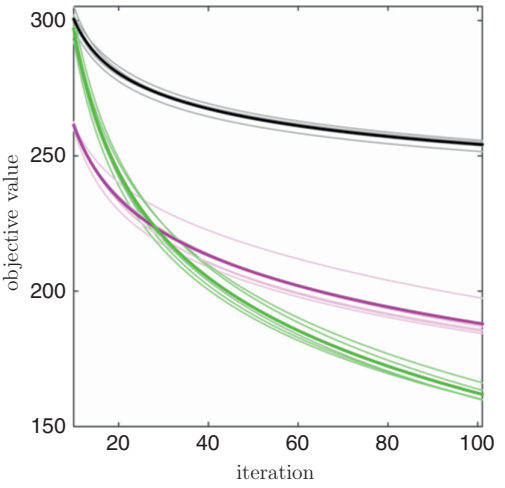

Fig. 8.9 The objective value of the first $k = 1$–100 iterations of three versions of stochastic gradient descent with diminishing step-size rules, $\alpha_k = \frac{1}{k}$ and $\alpha_k = \frac{1}{\sqrt{k}}$, and a constant step size $\alpha_k = \frac{1}{3}$ for all k (shown in light black, magenta, and green respectively, with the average of the runs shown in bold of each color). The two-class classification dataset used here consists of $P = 10\,000$ data points (see text for further details). In this instance the constant step size-runs tend to outperform those governed by the provably convergent diminishing step-size rules.

8.4 Convergence proofs for gradient descent schemes

To set the stage for the material of this section, it will be helpful to briefly point out the specific set of mild conditions satisfied by all of the cost functions we aim to minimize in this book, as these conditions are relied upon explicitly in the upcoming convergence proofs. These three basic conditions are listed below:

(1) They have piecewise-differentiable first derivative.

(2) They are bounded from below, i.e., they never take on values at $-\infty$.

(3) They have bounded curvature.

While the first condition is specific to the set of cost functions we discuss, in particular so that we include the squared margin perceptron which is not smooth and indeed has piecewise differentiable first derivative while the other costs are completely smooth, the latter two conditions are very common assumptions made in the study of mathematical optimization more generally.

8.4.1 Convergence of gradient descent with Lipschitz constant fixed step length

With the gradient of g being Lipschitz continuous with constant L, from Section 8.1 we know that at the kth iteration of gradient descent we have a corresponding quadratic upper bound on g of the form

$$g(\mathbf{w}) \leq g\left(\mathbf{w}^{k-1}\right) + \nabla g\left(\mathbf{w}^{k-1}\right)^T\left(\mathbf{w} - \mathbf{w}^{k-1}\right) + \frac{L}{2}\|\mathbf{w} - \mathbf{w}^{k-1}\|_2^2, \tag{8.20}$$

where indeed this inequality holds for all $\mathbf{w}$ in the domain of g. Now plugging the form of the gradient step $\mathbf{w}^k = \mathbf{w}^{k-1} - \frac{1}{L}\nabla g\left(\mathbf{w}^{k-1}\right)$ into the above and simplifying gives

$$g\left(\mathbf{w}^k\right) \leq g\left(\mathbf{w}^{k-1}\right) - \frac{1}{2L}\|\nabla g\left(\mathbf{w}^{k-1}\right)\|_2^2, \tag{8.21}$$

which, since $\|\nabla g\left(\mathbf{w}^{k-1}\right)\|_2^2 \geq 0$, indeed shows that the sequence of gradient steps with conservative fixed step length is decreasing. To show that it converges to a stationary point where the gradient vanishes we subtract off $g\left(\mathbf{w}^{k-1}\right)$ from both sides of the above, and sum the result over $k = 1 \ldots K$ giving

$$\sum_{k=1}^{K} g\left(\mathbf{w}^k\right) - g\left(\mathbf{w}^{k-1}\right) = g\left(\mathbf{w}^K\right) - g\left(\mathbf{w}^0\right) \leq -\frac{1}{2L}\sum_{k=1}^{K}\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2. \tag{8.22}$$

Note importantly here that since g is bounded below so too is $g\left(\mathbf{w}^K\right)$ for all K, and this implies that, taking $K \longrightarrow \infty$, we *must* have that

$$\sum_{k=1}^{\infty}\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2 < \infty. \tag{8.23}$$

If this were not the case then we would contradict the assumption that g has a finite lower bound, since Equation (8.22) would say that $g\left(\mathbf{w}^K\right)$ would be negative infinity! Hence the fact that the infinite sum above must be finite implies that as $k \longrightarrow \infty$ we have that

$$\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2 \longrightarrow 0, \tag{8.24}$$

or that the sequence of gradient descent steps with step length determined by the Lipschitz constant of the gradient of g produces a vanishing gradient. Or, in other words, that this sequence indeed converges to a stationary point of g.

Note that we could have made the same argument above using any fixed step length smaller than $\frac{1}{L}$ as well.

8.4.2 Convergence of gradient descent with backtracking line search

With the assumption that g has bounded curvature, stated formally in Section 8.1.2 that g has a Lipschitz continuous gradient with some constant L (even if we cannot calculate

L explicitly), it follows that with a fixed choice of initial step length $\alpha > 0$ and $t \in (0, 1)$ for all gradient descent steps, we can always find an integer n_0 such that

$$t^{n_0}\alpha \leq \frac{1}{L}. \tag{8.25}$$

Thus we always have the lower bound on an adaptively chosen step length $\hat{t} = t^{n_0}\alpha$, meaning formally that the backtracking found step length at the kth gradient descent step will always be larger than this lower bound, i.e.,

$$\alpha_k \geq \hat{t} > 0. \tag{8.26}$$

Now, recall from Section 8.2 that by running the backtracking procedure at the kth gradient step we produce a step length α_k that ensures the associated quadratic upper bound

$$g(\mathbf{w}) \leq g\left(\mathbf{w}^{k-1}\right) + \nabla g\left(\mathbf{w}^{k-1}\right)^T\left(\mathbf{w} - \mathbf{w}^{k-1}\right) + \frac{1}{2\alpha_k}\|\mathbf{w} - \mathbf{w}^{k-1}\|_2^2 \tag{8.27}$$

holds for all $\mathbf{w}$ in the domain of g. Plugging the gradient step $\mathbf{w}^k = \mathbf{w}^{k-1} - \alpha_k \nabla g\left(\mathbf{w}^{k-1}\right)$ into the above and simplifying we have equivalently that

$$g\left(\mathbf{w}^k\right) \leq g\left(\mathbf{w}^{k-1}\right) - \frac{\alpha_k}{2}\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2, \tag{8.28}$$

which indeed shows that that step produces decrease in the cost function. To show that the sequence of gradient steps converges to a stationary point of g we first subtract off $g\left(\mathbf{w}^{k-1}\right)$ and sum the above over $k = 1 \ldots K$ which gives

$$\sum_{k=1}^{K} g\left(\mathbf{w}^k\right) - g\left(\mathbf{w}^{k-1}\right) = g\left(\mathbf{w}^K\right) - g\left(\mathbf{w}^0\right) \leq -\frac{1}{2}\sum_{k=1}^{K}\alpha_k\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2. \tag{8.29}$$

Since g is bounded below so too is $g\left(\mathbf{w}^K\right)$ for all K, and therefore taking $K \longrightarrow \infty$ the above says that we must have

$$\sum_{k=1}^{\infty}\alpha_k\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2 < \infty. \tag{8.30}$$

Now, we know from Equation (8.26) that since each $\alpha_k \geq \hat{t} > 0$ for all k, this implies that we must have that

$$\sum_{k=1}^{\infty}\alpha_k = \infty. \tag{8.31}$$

And this is just fine, because in order for Equation (8.30) to hold under this condition we *must* have that

$$\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2^2 \longrightarrow 0, \tag{8.32}$$

as $k \longrightarrow \infty$, for otherwise Equation (8.30) could not be true. This shows that the sequence of gradient steps determined by backtracking line search converges to a stationary point of g.

8.4.3 Convergence of the stochastic gradient method

To understand what kind of step length rule we will need to mathematically force the stochastic gradient method to converge we first relate the kth iteration of stochastic gradient descent to a full standard gradient step. This is accomplished by unraveling the definition of the gradient iteration given in Equation (8.19), and writing out the kth iteration (note here that we ignore the bias term for ease of exposition) as

$$\begin{aligned}\mathbf{w}^{k,0} &= \mathbf{w}^{k-1,P} = \mathbf{w}^{k-1,P-1} - \alpha_k \nabla h\left(\mathbf{w}^{k-1,P-1}, \mathbf{x}_P\right) \\ &= \ldots = \mathbf{w}^{k-1,0} - \alpha_k \sum_{p=1}^{P} \nabla h\left(\mathbf{w}^{k-1,p-1}, \mathbf{x}_p\right).\end{aligned} \tag{8.33}$$

Next, by adding and subtracting the full gradient at $\mathbf{w}^{k-1,0}$, that is $\nabla g\left(\mathbf{w}^{k-1,0}\right) = \sum_{p=1}^{P} \nabla h\left(\mathbf{w}^{k-1,0}, \mathbf{x}_p\right)$, weighted by the step length α_k, and by referring to

$$\epsilon_k = \sum_{p=2}^{P} \left(\nabla h\left(\mathbf{w}^{k-1,p-1}, \mathbf{x}_p\right) - \nabla h\left(\mathbf{w}^{k-1,0}, \mathbf{x}_p\right)\right), \tag{8.34}$$

the gradient iteration can be rewritten equivalently as

$$\mathbf{w}^{k,0} = \mathbf{w}^{k-1,0} - \alpha_k \left(\nabla g\left(\mathbf{w}^{k-1,0}\right) + \epsilon_k\right) . \tag{8.35}$$

In other words, the above expresses the kth gradient iteration as a standard gradient step, with the additional "error" term ϵ_k. Since we have expressed the kth iteration of the stochastic gradient method in this manner, to simplify notation from here on we remove the redundant second superscript, writing the above more simply as

$$\mathbf{w}^{k} = \mathbf{w}^{k-1} - \alpha_k \left(\nabla g\left(\mathbf{w}^{k-1}\right) + \epsilon_k\right) . \tag{8.36}$$

This can be thought of as a "noisy" gradient step.[8] As we have seen previously, a properly designed step size α_k that forces the gradient $\nabla g\left(\mathbf{w}^{k-1}\right)$ to vanish for large k is that $\sum_{k=1}^{\infty} \alpha_k = \infty$, where each $\alpha_k \leq \frac{1}{L}$. However, in order for the stochastic gradient method to converge to a stationary point of g we will also need the error ϵ_k to vanish.

By analyzing ϵ_k one can show[9] that the norm of the kth error term $\|\epsilon_k\|_2$ is in fact bounded by a constant proportional to the corresponding step length α_k, i.e.,

$$\|\epsilon_k\|_2 \leq \alpha_k S \tag{8.41}$$

[8] The mathematical form of Equation (8.36), as a noisy gradient step, arises more generally in the case where the gradient of a function cannot be effectively computed (i.e., the only gradient calculation available is polluted by noise). In this more general instance, ϵ_k is typically modeled using a random variable, in which case the step defined in Equation (8.36) is referred to as a *stochastic gradient descent*. Because of this similarity in mathematical form the iterative gradient method is also often referred to as stochastic gradient descent, although the error ϵ_k is not random but given explicitly by Equation (8.34).

[9] Using the definition of the error, the triangle inequality, and the fact that each h has Lipschitz continuous gradient with constant L_h, we have that

$$\|\epsilon_k\|_2 \leq \sum_{p=2}^{P} \left\| \left(\nabla h\left(\mathbf{w}^{k-1,p-1}, \mathbf{x}_p\right) - \nabla h\left(\mathbf{w}^{k-1,0}, \mathbf{x}_p\right)\right)\right\|_2 \leq L_h \sum_{p=2}^{P} \left\|\mathbf{w}^{k-1,p-1} - \mathbf{w}^{k-1,0}\right\|_2 . \tag{8.37}$$

for some constant S. Therefore, by adding the condition to the step size scheme that $\alpha_k \longrightarrow 0$ as $k \longrightarrow \infty$ we can force the error term $\|\epsilon_k\| \longrightarrow 0$ as well. Altogether then our conditions on the step size for convergence of the stochastic gradient method include that $\sum_{k=1}^{\infty} \alpha_k = \infty$ (which forces the gradient to vanish) and $\alpha_k \longrightarrow 0$ as k grows large (which forces the error to vanish). Any diminishing sequence like e.g., $\alpha_k = \frac{1}{k}$ or $\alpha_k = \frac{1}{\sqrt{k}}$ etc., satisfies such requirements. However, these two kinds of step-size rules are commonly used in practice as they balance our desire to cause both the gradient and error term to vanish (while slower or faster diminishing step size favors one term over the other practically speaking).

8.4.4 Convergence rate of gradient descent for convex functions with fixed step length

Suppose, in addition to g having bounded curvature with Lipschitz constant L, that g is also convex. As illustrated in Fig. 8.10, this implies that for any $\mathbf{w}$, $g(\mathbf{w})$ is majorized by (or lies underneath) a quadratic and minorized by (or lies over) a linear function. With convexity we can, in addition to assuring convergence of gradient descent, more easily calculate a convergence rate of gradient descent.

Now that we know the sequence is decreasing, all that is left is to make sure that the sequence converges to a global minimum of g, say $\mathbf{w}^\star$. We would like to show that $g\left(\mathbf{w}^k\right)$ converges to $g(\mathbf{w}^\star) > -\infty$ as k increases. Looking at the quadratic upper bound on $g\left(\mathbf{w}^i\right)$ centered at $\mathbf{w}^{i-1}$, as in (8.21), we have

$$g\left(\mathbf{w}^k\right) \leq g\left(\mathbf{w}^{k-1}\right) - \frac{1}{2L}\|\nabla g\left(\mathbf{w}^{k-1}\right)\|_2^2. \tag{8.42}$$

By using the definition of the stochastic gradient step we can roll back each $\mathbf{w}^{k-1,p-1}$ to $\mathbf{w}^{k-1,0}$ by expanding first $\mathbf{w}^{k-1,p-1} = \mathbf{w}^{k-1,p-2} - \alpha_k \nabla h\left(\mathbf{w}^{k-1,p-2}, \mathbf{x}_{p-1}\right)$ and then doing the same to $\mathbf{w}^{k-1,p-2}$, etc., giving

$$\mathbf{w}^{k-1,p-1} - \mathbf{w}^{k-1,0} = -\alpha_k \sum_{t=1}^{p-1} \nabla h\left(\mathbf{w}^{k-1,t-1}, \mathbf{x}_t\right). \tag{8.38}$$

Substituting this into the right hand side of Equation (8.37) for each $p = 1 \ldots P$ then gives the bound

$$\|\epsilon_k\|_2 \leq L_h \sum_{p=1}^{P} \left\| \alpha_k \sum_{t=1}^{p} \nabla h\left(\mathbf{w}^{k-1,t-1}, \mathbf{x}_t\right) \right\|_2 \leq L_h \alpha_k \sum_{p=1}^{P} \sum_{t=1}^{p} \left\| \nabla h\left(\mathbf{w}^{k-1,t-1}, \mathbf{x}_t\right) \right\|_2, \tag{8.39}$$

where the right hand side follows by the triangle inequality and the definition of the norm $\|\alpha \mathbf{z}\|_2 = \alpha \|\mathbf{z}\|_2$ when $\alpha \geq 0$. Finally, because we have chosen α_k such that the gradient $\nabla g\left(\mathbf{w}^{k-1}\right)$ will vanish, or in other words that $\left\{\mathbf{w}^{k-1}\right\}_{k=1}^{\infty}$ converges, and because each set of points $\left\{\mathbf{w}^{k-1,t-1}\right\}_{t=1}^{P}$ lies in a neighborhood of $\mathbf{w}^{k-1}$ for each k, we can say that the individual gradients $\left\|\nabla h\left(\mathbf{w}^{k-1,t-1}, \mathbf{x}_t\right)\right\|_2$ are bounded. Hence we can say that there is some fixed constant S such that $L_h \sum_{p=1}^{P} \sum_{t=1}^{p} \left\|\nabla h\left(\mathbf{w}^{k-1,t-1}, \mathbf{x}_t\right)\right\|_2 \leq S$ and therefore conclude that

$$\|\epsilon_k\|_2 \leq \alpha_k S. \tag{8.40}$$

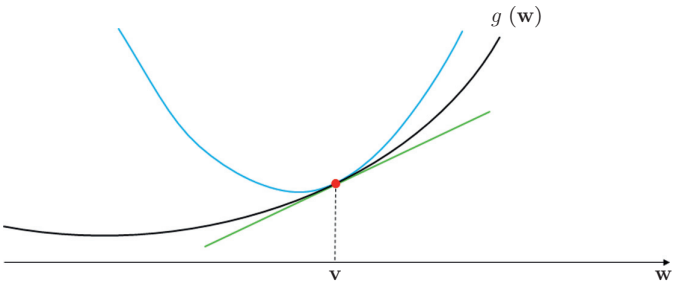

Fig. 8.10 The conservatively optimal quadratic form (in blue) majorizes the convex function g at $\mathbf{v}$, which (since it is convex) is also minorized by a linear function (in green) at each $\mathbf{v}$.

The first order definition of convexity at $\mathbf{w}^\star$gives the inequality

$$g\left(\mathbf{w}\right) + \nabla g\left(\mathbf{w}\right)\left(\mathbf{w}^\star - \mathbf{w}\right) \leq g\left(\mathbf{w}^\star\right). \tag{8.43}$$

Plugging in $\mathbf{w} = \mathbf{w}^{i-1}$ into the right side and rearranging gives

$$g\left(\mathbf{w}^{k-1}\right) \leq g\left(\mathbf{w}^\star\right) + \nabla g\left(\mathbf{w}^{k-1}\right)^T\left(\mathbf{w}^{k-1} - \mathbf{w}^\star\right). \tag{8.44}$$

Now we use this upper bound for $g\left(\mathbf{w}^{i-1}\right)$ on the right hand side of (8.42) to obtain

$$g\left(\mathbf{w}^{k}\right) \leq g\left(\mathbf{w}^\star\right) + \nabla g\left(\mathbf{w}^{k-1}\right)^T\left(\mathbf{w}^{k-1} - \mathbf{w}^\star\right) - \frac{1}{2L}\|\nabla g\left(\mathbf{w}^{k-1}\right)\|_2^2. \tag{8.45}$$

Bringing $g\left(\mathbf{w}^\star\right)$ to the left hand side, and using the fact that $\mathbf{w}^k = \mathbf{w}^{k-1} - \frac{1}{L}\nabla g\left(\mathbf{w}^{k-1}\right)$ implies that $\nabla g\left(\mathbf{w}^{k-1}\right) = L\left(\mathbf{w}^{k-1} - \mathbf{w}^k\right)$, simple rearrangement gives that the above is equivalent to

$$g\left(\mathbf{w}^{k}\right) - g\left(\mathbf{w}^\star\right) \leq \frac{L}{2}\left(\|\mathbf{w}^{k-1} - \mathbf{w}^\star\|_2^2 - \|\mathbf{w}^{k} - \mathbf{w}^\star\|_2^2\right). \tag{8.46}$$

Now averaging both sides over the first k steps gives the corresponding inequality

$$\frac{1}{k}\sum_{i=1}^{k}\left(g\left(\mathbf{w}^{i}\right) - g\left(\mathbf{w}^\star\right)\right) \leq \frac{L}{2k}\left(\|\mathbf{w}^{0} - \mathbf{w}^\star\|_2^2 - \|\mathbf{w}^{k} - \mathbf{w}^\star\|_2^2\right). \tag{8.47}$$

Note that because $\left\{g\left(\mathbf{w}^{i}\right)\right\}_{i=1}^{k}$ is decreasing, the left hand side itself has lower bound given by $g\left(\mathbf{w}^{k}\right) - g\left(\mathbf{w}^\star\right)$, and the right side has upper bound given by $\frac{L}{2k}\|\mathbf{w}^{0} - \mathbf{w}^\star\|_2^2$. Therefore we have

$$g\left(\mathbf{w}^{k}\right) - g\left(\mathbf{w}^\star\right) \leq \frac{L}{2k}\|\mathbf{w}^{0} - \mathbf{w}^\star\|_2^2. \tag{8.48}$$

The right hand side goes to zero as $k \longrightarrow \infty$ with the rate of $\frac{1}{k}$. In other words, to get within $\frac{1}{k}$ of the global optimum takes in the order of k steps.

8.5 Calculation of computable Lipschitz constants

Here we provide the calculations associated with several of the Lipschitz constants reported in Table 8.1. In particular instances it will be convenient to express curvature in terms of the *curvature function,* which for a general cost function g taking input $\mathbf{w}$ is given as

$$\psi\left(\mathbf{z}\right) = \mathbf{z}^T\left(\nabla^2 g\left(\mathbf{w}\right)\right)\mathbf{z}. \tag{8.49}$$

As was the case with convexity (see Exercise 2.11), the greatest curvature given by the Lipschitz constant L can be defined in terms of the eigenvalues of $\nabla^2 g\left(\mathbf{w}\right)$ in Equation (8.11) or the curvature function above. In particular, the greatest curvature of g is equivalently given by the minimum and maximum values taken on by its associated curvature function on the unit sphere where $\|\mathbf{z}\|_2 = 1$. Formally this is the smallest nonnegative L such that

$$-L \leq \psi\left(\mathbf{z}\right) \leq L \quad \text{for any } \mathbf{z} \text{ where } \|\mathbf{z}\|_2 = 1 \tag{8.50}$$

holds over the domain of g. Here L is precisely the Lipschitz constant defined in Equation (8.11). One can show fairly easily that the two definitions of Lipschitz constant are indeed equivalent (see Exercise 2.11).

Example 8.6 Least Squares for linear regression

The Least Squares cost function for linear regression $g\left(\tilde{\mathbf{w}}\right) = \|\tilde{\mathbf{X}}^T\tilde{\mathbf{w}} - \mathbf{y}\|_2^2$ has an easily calculable gradient and Hessian, given respectively by $\nabla g\left(\tilde{\mathbf{w}}\right) = 2\tilde{\mathbf{X}}\left(\tilde{\mathbf{X}}^T\tilde{\mathbf{w}} - \mathbf{y}\right)$ and $\nabla^2 g\left(\tilde{\mathbf{w}}\right) = 2\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$. Because the Least Squares cost is convex we know that its Hessian is positive semidefinite, so we have that

$$\mathbf{0}_{N+1\times N+1} \preceq \nabla^2 g\left(\tilde{\mathbf{w}}\right) \preceq 2\left\|\tilde{\mathbf{X}}\right\|_2^2 \mathbf{I}_{N+1\times N+1}, \tag{8.51}$$

using the definition of greatest curvature given in Equation (8.50). Therefore the Lipschitz constant is given by the largest eigenvalue of this matrix,

$$L = 2\left\|\tilde{\mathbf{X}}\right\|_2^2. \tag{8.52}$$

Example 8.7 Two class softmax cost

As we saw in Exercise 4.4, the Hessian of the softmax perceptron/convex logistic regression cost function is convex with Hessian given as

$$\nabla^2 g\left(\tilde{\mathbf{w}}\right) = \tilde{\mathbf{X}}\text{diag}\left(\mathbf{r}\right)\tilde{\mathbf{X}}^T, \tag{8.53}$$

where $\mathbf{r} = [r_1 \ldots r_P]^T$ is defined entry-wise as $r_p = \sigma\left(-y_p\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}}\right)\left(1 - \sigma\left(-y_p\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}}\right)\right)$.

Using the fact that $0 < r_n \leq \frac{1}{4}$ and the curvature definition in Equation (8.50) we can say that the following holds for each $\mathbf{z}$ on the unit sphere:

$$\mathbf{z}^T\left(\tilde{\mathbf{X}}\text{diag}\left(\mathbf{r}\right)\tilde{\mathbf{X}}^T\right)\mathbf{z} \leq \frac{1}{4}\mathbf{z}^T\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\mathbf{z}. \tag{8.54}$$

Now, because the right hand side is maximized when $\mathbf{z}$ is the eigenvector of the matrix $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$ associated to its largest eigenvalue, the maximum value attainable by the right hand side above is $\frac{1}{4}\left\|\tilde{\mathbf{X}}\right\|_2^2$ (see exercises). Therefore, altogether we have that

$$\mathbf{0}_{N\times N} \preceq \nabla^2 g\left(\tilde{\mathbf{w}}\right) \preceq \frac{1}{4}\left\|\tilde{\mathbf{X}}\right\|_2^2, \tag{8.55}$$

and therefore we may take as a Lipschitz constant

$$L = \frac{1}{4}\left\|\tilde{\mathbf{X}}\right\|_2^2. \tag{8.56}$$

Example 8.8 Squared margin hinge

The Hessian of the squared margin hinge function $g\left(\tilde{\mathbf{w}}\right) = \sum_{p=1}^{P}\max^2\left(0, 1 - y_p\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}}\right)$ can be written as

$$\nabla^2 g\left(\tilde{\mathbf{w}}\right) = 2\tilde{\mathbf{X}}_s\tilde{\mathbf{X}}_s^T. \tag{8.57}$$

Because the maximum eigenvalue of $\tilde{\mathbf{X}}_s\tilde{\mathbf{X}}_s^T$ is bounded above by that of the matrix $\tilde{\mathbf{X}}\,\tilde{\mathbf{X}}^T$ (see Exercise 8.5), a bound on the maximum eigenvalue of this matrix/the curvature of the cost g is given by

$$L = 2\left\|\tilde{\mathbf{X}}\right\|_2^2. \tag{8.58}$$

Example 8.9 Multiclass softmax

Since the multiclass softmax cost is convex we know that its Hessian, described blockwise in Exercise 4.18, must have all nonnegative eigenalues. Note that the maximum eigenvalue of its cth diagonal block $\nabla_{\tilde{\mathbf{w}}_c\tilde{\mathbf{w}}_c} g = \sum_{p=1}^{P}\frac{e^{\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}}_c}}{\sum_{d=1}^{C}e^{\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}}_d}}\left(1 - \frac{e^{\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}}_c}}{\sum_{d=1}^{C}e^{\tilde{\mathbf{x}}_p^T\tilde{\mathbf{w}}_d}}\right)\mathbf{x}_p\mathbf{x}_p^T$ is, in the same manner as the two class softmax cost in Example 8.7, given by $\frac{1}{4}\left\|\tilde{\mathbf{X}}\right\|_2^2$. Because the maximum eigenvalue of a symmetric block matrix with nonnegative eigenvalues is bounded above by the sum of the maximum eigenvalues of its diagonal blocks [28], an upper bound on the maximum eigenvalue of the Hessian is given by

$$L = \frac{C}{4}\left\|\tilde{\mathbf{X}}\right\|_2^2. \tag{8.59}$$

8.6 Summary

In the first two sections of this chapter we described two rigorous ways for determining step lengths for gradient descent. First in Section 8.1 we introduced the optimally conservative fixed step length rule (based on the notion of the maximum curvature of a cost function). This can be used to produce a fixed step length that, while conservative in nature, will guarantee convergence of gradient descent when applied to every cost function detailed in this book (see Table 8.1).

Next, in Section 8.2 we introduced an adaptive step length rule for gradient descent, another rule that works right out of the box for any cost function. As discussed in Section 8.2.2, the adaptive step length rule requires more computation at each step of gradient descent but typically takes considerably longer steps than a conservative fixed counterpart.

Finally, in Section 8.3 we introduced the stochastic gradient descent method. A natural extension of the standard gradient descent scheme, stochastic gradient descent is especially useful when dealing with large datasets (where, for example, loading the full dataset into memory is challenging or impossible). On large datasets the stochastic gradient method can converge extremely rapidly to a reasonable solution of a problem, even after only a single pass through the dataset.

8.7 Exercises

Section 8.1 exercises

Exercises 8.1 Code up backtracking line search for the squared margin cost

Code up an adaptive step length sub-function for the minimization of the squared margin perceptron and install it into the wrapper detailed in Exercise 4.7, replacing the fixed step length given there. Test your code by running the wrapper, and produce a plot of the objective value decrease at each iteration.

Exercises 8.2 Code up stochastic gradient descent

Reproduce a part of the experiment shown in Example 8.4 by comparing standard and stochastic gradient descent methods on a large two class classification dataset of $P = 10\,000$ points. Employ any cost function (e.g., softmax) to fit to this data and plot the cost value at each iteration from runs of each method, along with the average value of these runs (as in Fig. 8.8).

Exercises 8.3 Code up backtracking line search for the multiclass softmax cost

Code up an adaptive step length sub-function for the minimization of the multiclass softmax perceptron and install it into the wrapper detailed in Exercise 4.15, replacing the fixed step length given there. Test your code by running the wrapper, and produce a plot of the objective value decrease at each iteration.

Exercises 8.4 Alternative formal definition of Lipschitz gradient

An alternative to defining the Lipschitz constant by Equation (8.11) for functions f with Lipschitz continuous gradient is given by

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L \|\mathbf{x} - \mathbf{y}\|_2, \tag{8.60}$$

which follows from the limit definition of a derivative (in defining the Hessian of f). This definition is especially helpful to employ when f has only a single continuous derivative.

Suppose f has Lipschitz continuous gradient with constant J, and g is Lipschitz continuous with constant K, i.e.,

$$\|g(\mathbf{x}) - g(\mathbf{y})\|_2 \leq K \|\mathbf{x} - \mathbf{y}\|_2 \tag{8.61}$$

for all $\mathbf{x}$, $\mathbf{y}$ in the domain of g. Using this definition of Lipschitz continuous gradient show that the composition $f(g)$ also has Lipschitz continuous gradient. What is the corresponding Lipschitz constant?

Exercises 8.5 Verifying Lipschitz constants

Let $\mathbf{a}$ and $\mathbf{b}$ be two $N \times 1$ column vectors.

a) Show that the maximum eigenvalue of $\mathbf{a}\mathbf{a}^T$ is less than or equal to the that of $\mathbf{a}\mathbf{a}^T + \mathbf{b}\mathbf{b}^T$.

b) Use the result of part a) to verify the bound reported in Example 8.8 on the maximum eigenvalue of the soft-margin SVM's Hessian.

c) Use the result of part a) to verify the bound on the maximum eigenvalue of the Hessian of the logistic regression function reported in Example 8.7.