# STAT 538 Lecture 3
January 21, 2011

**Unconstrained optimization – Part II**

©Marina Meilă
mmp@stat.washington.edu

**Reading:** For mathematical background, Appendix A of Boyd & Vandenberghe (BV). For Unconstrained optimization proper, BV Ch 9. The notes follow mostly Bertsekas (B) Chapter 1. For the line minimization methods, look at B (ch 1), at "Numerical recipes" (NR) chapter 10. NW (Ch 3, 6) is also useful reading for both line minimization and Quasi-Newton.

## 1 Multidimensional minimization. The choice of direction (Continued)

### 1.1 The Conjugate gradient method

Conjugate Gradient Algorithm

```
    $x^{k+1} = x^k + \alpha^k d^k$ where $\alpha^k$ is chosen by line minimization
1.  $d^0 = -\nabla f(x^0)$
2.  for $k = 1, \ldots n-1$, $d^k = -\nabla f(x^k) + \beta^k d^{k-1}$ with
```

$$\beta^k \;=\; \frac{\nabla f(x^k)^T(\nabla f(x^k) - \nabla f(x^{k-1}))}{\nabla f(x^{k-1})^T \nabla f(x^{k-1})} \tag{1}$$

```
3.  restart from step 1 if $k = n$
```

**Intuition** If $H = \nabla^2 f(x^*) = I$ and $f$ quadratic, then the level curves of $f$ around $x^*$ would be circles. After line minimization along any direction of descent, the next direction of descent will be $\perp$ on the previous one, and in $n$ steps the min is attained.

Let

$$z = H^{1/2}(x - x^*) \tag{2}$$

where the notation $A^{1/2}$ represents the **matrix square root** of the symmetric positive definite matrix $A$, i.e the matrix $B$ such that $B^T B = A$. $B$ is real whenever the eigenvalues of $A$ are non-negative; $B$ is not necessarily symmetric, and it is not unique as multiplying $B$ to the right with an orthogonal matrix produces another square root of $A$.

The usual Taylor approximation of $f$ around the minimimum gives

$$\begin{aligned} f &\approx f(x^*) + (x - x^*)^T H(x - x^*) + \dots && (3)\\ &= f(x^*) + z^T z + \dots && (4)\\ &= f(x^*) + ||z||^2 + \dots && (5)\\ & && (6) \end{aligned}$$

In the new variable $z$, the level curves are (approximately) circles. Let $\tilde{d}_{1:k}$ be orthogonal consecutive directions of descent in the $z$ space. We have then

$$\begin{aligned} z^k - z^{k-1} &= \beta_k \tilde{d}_k && (7)\\ H^{-1/2}(z^k - z^{k-1}) &= \beta_k H^{-1/2}\tilde{d}_k && (8)\\ x^k - x^{k-1} &= \beta_k \underbrace{H^{-1/2}\tilde{d}_k}_{d_k} && (9) \end{aligned}$$

The directions $d^k$ in the above satisfy $d^k H d^j = 0$ for all $k \neq j$ (this is the definition of conjugate directions). Equivalently, $H^{1/2}d^k \perp H^{1/2}d^j$, which means that these directions become orthogonal if we applied the coordinate transformation $H^{-1/2}$. But this transformation is exactly the transformation we need to turn the level curves of $f$ into circles. Hence, we can view the directions $d^k$ as the correct orthogonal direction of descent in a transformed coordinate system. Note that exact line minimization is essential for the conjugate gradient method. It is also essential to restart the method after $n$ steps. In practice one can also restart after less than $n$ steps, especially if loss of conjugacy is suspected (because of e.g inexact line minimization).

The smart thing about the method is that these directions are computed recursively, without explicilty estimating, storing or inverting the $H$ matrix, a big saving in high dimensions.

**Rates of convergence** Practical experience and exact results for quadratic functions suggest that the conjugate gradients method is superlinear. The superlinear convergence is preserved even if the method is restarted after less than $n$ steps. [B]

## 1.2 Quasi-Newton (variable metric) methods

The idea is to approximate the (inverse) Hessian from the differences

$$\begin{aligned} s^k &= x^{k+1} - x^k && (10) \\ y^k &= \nabla f(x^{k+1}) - \nabla f(x^k) && (11) \end{aligned}$$

Below is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, which is considered the best general purpose quasi-Newton method.

BFGS Direction Update

1. Start with any $D^0 \succ 0$
2. For $k = 1, 2, \ldots$ update

$$\begin{aligned} \text{Set } \rho^k &= (s^k)^T y^k && (12) \\ D^{k+1} &= (I - \rho^k s^k (y^k)^T) D^{k-1} (I - \rho^k y^k (s^k)^T) + \rho^k s^k (s^k)^T && (13) \end{aligned}$$

Below is the update equation for the Broyden family of methods, which depends on a parameter $\xi$. The parameter $\xi^k \in [0,1]$ typically, but values outside this interval have been studied too; for $\xi^k = 1$ one obtains the BFGS.

$$\begin{aligned} D^{k+1} &= D^k + \frac{s^k (s^k)^T}{(s^k)^T y^k} - \frac{D^k y^k (y^k)^T D^k}{(y^k)^T D^k y^k} + \xi^k \tau^k v^k v^{kT} && (14) \\ v^k &= \frac{s^k}{(s^k)^T y^k} - \frac{D^k y^k}{\tau^k} && (15) \\ \tau^k &= (y^k)^T D^k y^k && (16) \end{aligned}$$

**Proposition 1** *If $f$ is quadratic and we run the quasi-Newton method with line minimization (or Wolfe conditions) for $n$ steps, then*
*(1) the vectors $d^0, d^1, \ldots d^{n-1}$ are $A$-conjugate*
*(2) $D^n = A^{-1}$*

**Advantages and disadvantages** The main disadvantage of the quasi-Newton methods compared to the conjugate directions methods is their large computational complexity ($n^2$ versus $n$). The advantage is that the method is robust to inexact line minimization, and that it does not need restarts. In addition, near the optimum, the directions quasi-Newton generates are conjugate and so it is equivalent to conjugate gradients and Newton-Raphson and thus converges very fast.

**Intuition about BFGS**
**First Idea** If the function is nearly quadratic and the step sizes are small (i.e. if we are near an optimum), then the equation below holds approximatively

$$y^k = \nabla^2 f(x^{k+1}) s^k \tag{17}$$

The above is called the *secant equation*, and sometimes quasi-Newton methods are called *matrix secant methods.* From (17) it follows that, given $n$ linearly independent pairs $(s^k, y^k)$ one can approximate the (inverse) Hessian by

$$\nabla^2 f(x^n)^{-1} \approx [s^0\, s^1\, \dots\, s^{n-1}][y^0\, y^1\, \dots\, y^{n-1}]^{-1} \tag{18}$$

Quasi-Newton methods use this idea to recursively construct an approximation to $\nabla^2 f^{-1}$.

**The second idea** is to use the previous approximation $D^{k-1}$ to construct $D^k$ recursively. $D^{k-1}$ already satisfies the secant equation (17) for $k \leftarrow k-n+1, \dots k-1$ therefore we need to only impose the last linear constraint. In addition, we shall ask that $D^k$ is "close" to $D^{k-1}$ in the weighted Frobenius norm

$$||A||_W^2 = ||W^{-1/2} A W^{-1/2}||_F^2 = \sum_{i,j} (W^{-1/2} A W^{-1/2})_{ij}^2 \tag{19}$$

The matrix $W$ is chosen so as to make the norm invariant to coordinate changes. In BFGS $W$ is chosen to be the average Hessian $\int_0^1 \nabla^2 f(x^k + ts^k)dt = \bar{H}$ (note that this is only for theoretical purposes – one does not actually compute it!) and sets

$$D^k = \text{argmin}_D\, ||D - D^{k-1}||_{\bar{H}}^2 \text{ s.t. } D \text{ symmetric and } D^k y^k = s^k \tag{20}$$

The unique solution to this problem is the BFGS $D^k$ given in equation (13).

**Third idea** One can make BFGS even more efficient if one settles for an approximation. The resulting algorithm is called L-BFGS (Limited Memory

BFGS) and achieves two things: first, it approximates $D^k$ by a rank $m$ matrix, with $m < n$. Such a matrix can be expressed as

$$D^k \;=\; M^k (M^k)^T, \;\; M^k \in \mathbb{R}^{n\times m}$$

using only $m \times n$ space and computation instead of $n^2$. Second, L-BFGS updates the direction $d^k = D^k \nabla f(x^k)$ directly. *See Resources page article by Galen Andrew or NW for more details.*

## 1.3 Convergence and rates for gradient based methods

We say that a direction (sequence) is **gradient related**, if for any sequence $(x^k)_k$ such that $x^k \longrightarrow x$ with $x$ non-stationary, the corresponding sequence of directions $(d^k)_k$ is bounded and satisfies

$$\operatorname*{limsup}_{k\longrightarrow\infty} \nabla f(x^k)^T d^k \;<\; 0 \tag{21}$$

This condition ensures that the search directions do not become, in the limit, orthogonal to the gradient, nor 0, unless we approach a stationary point.

**Proposition 2** *A direction $d^k = -D^k \nabla f(x^k)$ is gradient related if for every eigenvalue $\lambda$ of $D^k$ we have*

$$0 \;<\; m \;\leq \lambda \;\leq M$$

*for some constants $m, M$.*

The first question is: do the methods described here converge to stationary points (or minima) of $f$? We shall see that the answer is yes, given that the direction choice is gradient related (always so for steepest descent) modulo some stronger or weaker assumptions. The "safest" methods (assuming the least) require line minimization, while having a constant step size requires the strongest assumptions.

**Proposition 3 Convergence of line search** *Let $x^{k+1} = x^k + \alpha^k d^k$ be a sequence generated by a gradient method with $\{d^k\}$ gradient related and $\alpha^k$ generated by the line minimization rule, truncated line minimization or the Armijo rule. Then every limit point of $\{x^k\}$ is a stationary point.*

**Proposition 4 Convergence of constant step methods** *Let $x^{k+1} = x^k + \alpha^k d^k$ be a sequence generated by a gradient method with $\{d^k\}$ gradient related. Assume that for some constant $L > 0$ we have*

$$||\nabla f(x) - \nabla f(y)|| \leq L||x - y|| \quad \text{for all } x, y \in \mathbb{R}^n \text{ (i.e. } \nabla f \text{ is Lipschitz)} \tag{22}$$

*and that for all $k$ we have $d^k \neq 0$ and for some fixed $\epsilon > 0$*

$$\epsilon \ \leq \ \alpha^k \ \leq \ (2 - \epsilon)\bar{\alpha}^k \ \ \textit{where}\ \bar{\alpha}^k \ = \ \frac{|\nabla f(x^k)^T d^k|}{L||d^k||^2}. \tag{23}$$

*Then every limit point of $\{x^k\}$ is a stationary point.*

**Proposition 5 Convergence for diminishing step size** *Let $x^{k+1} = x^k + \alpha^k d^k$ be a sequence generated by a gradient method with $\{d^k\}$ gradient related and $\alpha^k \to 0$, $\sum_k \alpha^k = \infty$. Assume that for some constant $L > 0$ we have*

$$||\nabla f(x) - \nabla f(y)|| \leq L||x - y|| \quad \text{for all } x, y \in \mathbb{R}^n \text{ (i.e. } \nabla f \text{ is Lipschitz)} \tag{24}$$

*and that there exist $c_1, c_2 > 0$ so that for all $k$ we have*

$$c_1||d^k||^2 \ \leq ||\nabla f(x^k)||^2 \ \leq \ c_2(-\nabla f(x^k))^T d^k \tag{25}$$

*Then either $f(x^k) \to -\infty$ or $f(x^k)$ converges to a finite value and $\nabla f(x^k) \to 0$. (Consequently, every limit point of $\{x^k\}$ is a stationary point for $f$).*

These results guarantee that the values $x^k$ for the above methods converge to a unique stationary point under fairly mild conditions, given below.

**Proposition 6 Capture theorem** *Let $f$ be continuously differentiable, let $\{x^k\}$ be a sequence generated by a gradient method so that all its limit points are stationary points of $f$. Assume there exist $s, c > 0$ so that*

$$\alpha^k \ \leq \ s, \quad ||d^k|| \leq c||\nabla f(x^k)||, \quad f(x^{k+1}) \ \leq \ f(x^k) \tag{26}$$

*Then, for any isolated stationary point $x^*$ which is a local minimum, there is an open set $S$ so that, if $x^{\bar{k}} \in S$ for some $\bar{k}$, then $x^k \in S$ for all $k > \bar{k}$ and $x^k \to x^*$.*

Now we shall see some results about convergence rates. Here, the conclusions are that steepest descent is linear (at best), while Newton-Raphson and the methods that attempt to approximate the Hessian are typically superlinear.

**Proposition 7** [B] *Assume that $\alpha^k$ is chosen by the line minimization rule and that $x^k \to x^*$ with $\nabla f(x^*) = 0$, $\nabla^2 f(x^*) \succ 0$. Let $Q^k = \sqrt{D^k}\nabla^2 f(x^k)\sqrt{D^k}$ and let $m^k, M^k$ be the smallest, respectively largest eigenvalue or $Q^k$ (assuming that $D^k, \nabla^2 f(x^k) \succ 0$). Then,*

$$\alpha^k_{opt} = \frac{2}{M^k + m^k} \tag{27}$$

*and*

$$\limsup_{k\to\infty} \frac{f(x^{k+1}) - f(x^*)}{f(x^k) - f(x^*)} \leq \limsup_{k\to\infty} \left(\frac{M^k - m^k}{M^k + m^k}\right)^2 \tag{28}$$

One can show that the above bound is tight for steepest descent.

**Proposition 8 (B)** *The Newton-Raphson algorithm converges superlinearly Assume $x^k \to x^*$, $\nabla f(x^*) = 0$, $\nabla^2 f(x^*) \succ 0$ and*

$$\lim_{k\to\infty} \frac{||d^k + \nabla^2 f(x^k)^{-1}\nabla f(x^k)||}{||\nabla f(x^k)||} = 0 \tag{29}$$

*and $\alpha^k$ is chosen by the Armijo rule, with $s = 1$, $\sigma < 1/2$. Then,*

$$\lim_{k\to\infty} \frac{||x^{k+1} - x^*||}{||x^k - x^*||} = 0 \tag{30}$$

## 2 Noisy gradient and no gradient methods

These are the "cheap and slow" methods which can however be useful too. One should not confuse "theoretically slow" with "slow in practice" and on some problems the former is true of the simpler methods but the latter is not. On other occasions, these methods perform well because they make fewer assumptions about the smoothness of the surface $f(x)$.

## 2.1 Stochastic gradient methods

This class of methods typically includes steepest descent methods with diminishing step size. It is assumed that the gradient is computed with some error, that has 0 mean and bounded variance.

An important example of such a case is the fitting of a model to data. Let $\mathcal{D}_N$ be an i.i.d sample of size $n$ from an unknown distribution. Denote by $f(\theta) = -\frac{1}{N} \ln P(\mathcal{D}_N|\theta)$ the negative log-likelihood to be minimized. Because the sample is i.i.d., $f(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \ln p(x^i|\theta)$. Since $f$ is a sum, so will be the gradient:

$$\nabla f(\theta) \;=\; -\frac{1}{N}\sum_{i=1}^{N} \frac{\frac{\partial p(x^i|\theta)}{\partial \theta}}{p(x^i|\theta)} \tag{31}$$

If $N$ is large, something desirable from the statistical point of view, then the computation of the gradient is very costly (linear in $N$). A practical option is to take

$$d^k \;=\; \frac{\frac{\partial p(x^i|\theta)}{\partial \theta}}{p(x^i|\theta)} \tag{32}$$

where $x^i$ is a randomly sampled point from $\mathcal{D}_N$. Let $P_X$ be the true data distribution and $\hat{P}_X$ the empirical distribution induced by the sample $\mathcal{D}_N$. Note that the direction $d^k$ satisfies $E_{\hat{P}}[d^k] = \nabla f$.

More remarkably, this method can naturally be adapted to on-line learning; i.e situations where data come one by one, and are not stored but used immediately to update the parameters, then discarded. Then $x^i \sim P_X$, $f = E_P[-\ln p(X|\theta)]$ (we denote by $p$ the model, and by $P$ the true data distribution), and $d^i = -\nabla_\theta \ln p(x^i|\theta)$. It can be easily seen that $E_P[d^i] = \nabla f$. Also, in many cases, it can be shown that the variance of $d^k$ is bounded.

This example points out that:

- the function $f$ and its gradient $\nabla f$ are both expensive to evaluate, because they are proportional to the potentially large sample size $N$
- therefore, we want to avoid not only the $\nabla^2 f$ computation, but also the $\nabla f$ computation and even the line search which entails repeated evaluations of $f$
- on the other hand, evaluating a noisy version of the gradient is (assumed to be) fast.

Thus, it pays to take many imprecise steps, instead of few but very computationally demanding precise steps. It remains to see if such a method can effectively find a minimum.

It has been proved under various technical conditions[1] that stochastic gradient methods converge to the true optimum if the step sizes $\alpha^k$ satisfy $\sum_k \alpha^k = \infty$, $\sum_k (\alpha^k)^2 < \infty$ (the latter implies $\alpha^k \to 0$) and the noise variance is bounded. Essentially, for convergence the $\alpha^k$'s should decrease like $\frac{1}{k}$. Note however that typically in practice the decrease needs to be **very slow**, almost constant e.g $\frac{1}{b+k/c}$ with $b, c$ large numbers.

While these results are general and old, with the advent of very large data sets, interest in stochastic gradient for fitting statistical models has been revived and new results specifical to this context are advancing the understanding of this method.

In particular, it is assumed that (i) $E[d^k] = \nabla f(x^k)$, (ii) $Var\, d^k \leq G^2$, a known constant, and for the proofs, that (iii) $f$ is $\lambda$-strongly convex. It is not assumed that $f$ is differentiable (in this case $\nabla f$ above has the meaning of a *subgradient*, defined later in the context of convex functions).

If there are constraints on $\theta$, we additionally assume that the *projection* $\Pi_{\mathcal{A}}\theta = \operatorname*{argmin}_{\theta' \in \mathcal{A}} ||\theta - \theta'||$ can be computed efficiently (e.g. $\theta \succeq 0$).

```
Stochastic Gradient Descent (SGD)
Input $G^2$ upper bound on the variance, $\lambda$, $c > 1/2$ a constant giving the
    step-size, [optional $\alpha \in (0,1]$, $K$=total number steps]
for $k = 1, 2, \ldots K$
  1. get $d^k$
     If in a statistical learning task, we sample a point $x^i$ at random from
     $\mathcal{D}$ and compute $d^k$ based on it. Alternatively, we can sample a random
     permutation of $\mathcal{D}$ and pick the points sequentially from it.
  2. update $\theta$
        $\theta^{k+1} \leftarrow \theta^k - \frac{c}{\lambda k} d^k$        (33)
  3. if $k > (1-\alpha)K$ accumulate $\bar{\theta} \to \bar{\theta} + \theta^k$
```

[1]These results are best known under the name of Robbins-Munro theory of *stochastic approximation.*

Average
$$\bar\theta \leftarrow \frac{\bar\theta}{\alpha K} \tag{34}$$
**Output** $\bar\theta$ (or optionally $\theta^K$)

Thus, the algorithm takes steps proportional to $1/k$ and averages the last $\alpha$ fraction of steps. When $\alpha = 1$, the average of all steps is taken, but (as the next theorem shows) theoretical convergence is faster when $\alpha < 1$, i.e. when the initial values are discarded. Practically, one has noticed that no averaging, i.e. taking the last sample $\theta^K$ has also good convergence properties, but (as expected) larger variance than the averaged $\bar\theta$.

**Proposition 9** *Assume $f$ is $\lambda$-strongly convex, $\theta^*$ is the true optimum of $f$, and that assumptions (i,ii) above hold as well. Then, after $K$ steps of the* SGD *algorithm*

*1.*
$$E[||\theta^K - \theta^*||^2] \le c'\frac{G^2}{\lambda^2 K} \tag{35}$$
*where $c'$ is a constant that depends on $c$ but not on $G^2, \lambda, T$.*

*2. if $\alpha = 1$ and the problem is smooth, or if $\alpha < 1$ and any problem, smooth or non-smooth,*
$$E[||f(\bar\theta) - f(\theta^*)||] \le \frac{c''}{\sqrt{K}} \tag{36}$$
*where $c''$ is a constant that may depend on $c, \alpha, \lambda, G^2$ and other parameters, but not $K$.*

[To add: relation to Fisher information]

Stochastic gradient and analog techniques are widely used in machine learning: training of neural networks, reinforcement learning (the TD-$\lambda$ and Q-learning procedures are stochastic gradient methods), speedup of boosting.

[Exercise: Prove that least-squares linear regression is also (under mild conditions) a strongly convex problem. Find $\lambda$ and derive the SGD algorithm to solve it. Consider if/when the SGD algorithm would be faster than computing the closed form solution.]

## 2.2 Example: Linear classification with hinge loss

The following example is a classic in statistical learning. We will examine it in two formulation. The first is an example of a problem where $\lambda$ is known, and the SGD theory from above applies. The algorithm has been shown to be empirically effective as well as simple.

In the second instance, the above theory does not apply, but another theorem demonstrates some of the properties of the resulting classifier.

We start with a definition: Assume that $y \in \{\pm 1\}$ (binary classification) and $f(x) \in \mathbb{R}$, so that the classifier outputs $\hat{y}(x) = \operatorname{sgn} f(x)$. The **hinge loss** for classification is the following loss function

$$L_h(y, f(x)) \;=\; \begin{cases} 0 & \text{if}\, yf(x) \geq 1 \\ 1 - yf(x) & \text{if}\, yf(x) < 1 \end{cases} \;=\; [yf(x) - 1]_- \tag{37}$$

In words, an error is penalized linearly by how far $f(x)$ is in the "wrong direction" to which we add a penalty even for correctly classified examples if the **margin** $yf(x)$ is below 1.

We will fit the linear classifier

$$f(x) \;=\; w^T x \tag{38}$$

using this loss function.

In this section will make the simplifying assumption that the data $\mathcal{D} = \{(x^i, y^i)\}_{i=1:N}$ are **linearly separable**, i.e. there exists a $w^*$ that classifies the sample with no error. Note that in general this $w^*$ is *not unique.*

**Linear Support Vector Machine formulation.** The optimization problem is a regularized one:

$$\min_w \frac{1}{N} \sum_i L_h(y^i, w^T x^i) + \frac{\lambda}{2} ||w||^2 \tag{39}$$

with $\lambda > 0$ a regularization parameter chosen by the user. The reason for introducing this regularization will be explained later. For now, it suffices to notice that the non-quadratic loss term is linear (with unknown slope) and therefore the function $f$ is by definition $\lambda$-strongly convex.

The stochastic part of the gradient is

$$\frac{\partial L_h}{\partial w} = \begin{cases} y^i x^i & \text{if } i \text{ ``error''} \\ 0 & \text{if } i \text{ ``correct''} \end{cases} \tag{40}$$

where "correct" means that $y^i f(x^i) > 1$. Thus, the SGD algorithm becomes

```
**Initialize** with $w^0 = 0$, $\bar{w} = 0$
Iterate for $k = 1, 2, \dots K$
  1. Pick a random $i$ in $1 : N$ (or typically, pick $i$ from a random permutation of $\mathcal{D}$, until $\mathcal{D}$ is exhausted, then repeat with a new random permutation).
  2.
     $$d^k = \lambda w^k - \mathbf{1}_{i\text{"}error\text{"}} y^i x^i \tag{41}$$
  3.
     $$w^{k+1} = w^k - \frac{c}{\lambda k}\left(\lambda w^k - \mathbf{1}_{i\text{"}error\text{"}} y^i x^i\right) = w^k(1 - c/k) + \frac{c}{\lambda k} y^i x^i \mathbf{1}_{i\text{"}error\text{"}} \tag{42}$$
  4. $\bar{w} \leftarrow \bar{w} + w^{k+1}$
**Output** $\bar{w}$
```

**The Perceptron Algorithm.** The linear classifier (38) has been known since the early machine learning day under the name of *perceptron*. The following algorithm also dates from those days. There is no regularization, and the margin $-1$ is dropped from the hinge loss. Thus, finding a classifier for the dataset $\mathcal{D}$ is formulated as

$$\min_w \frac{1}{N} \sum_{i=1}^{N} [y^i w^T x^i]_- \tag{43}$$

If the data is linearly separable, as we assume for now, the minimum value of $f$ is known to be 0; what interests is the (non-unique!) $w$ that attains this minimum.

```
PERCEPTRON ALGORITHM
**Initialize** $w = 0$
For $k = 1, 2, \dots$
```

1. pick a data point $i$, calculate $f(x^i) = w^T x^i$
2. if $i$ is a mistake (i.e. $y^k w^T x^i \leq 0$)

$$w \leftarrow w + y^i x^i \tag{44}$$

until no more mistakes are made.
**Output** $w$

**Proposition 10** *The number of mistakes made by the* PERCEPTRON *algorithm is bounded by* $\frac{1}{\gamma^2}$ *where*

$$\gamma = \min_{\mathcal{D}} \frac{|(w^*)^T x^i|}{||x^i|| ||w^*||} \tag{45}$$

*with* $w^*$ *any linear separator of the data.*

**Exercise:** it makes sense to take $w^*$ to be the separator that maximizes $\gamma$, since this gives the tightest bound. Find what this $w^*$ is and what the interpretation of $\gamma$ should be in this case. Assume that $||x^i|| = 1$ for all $i$.

[TO DO: Proof of Proposition]

## 2.3 Numerical evaluation of the gradient

Sometimes, if the gradient is expensive to compute (and $f(x)$ is presumably not), the gradient can be evaluated by finite differences. Let $u_i$ represent the unit vector of coordinate $i$. The forward difference is

$$\frac{\partial f}{\partial x_i}(x^k) \approx \frac{f(x^k + hu_i) - f(x^k)}{h} \tag{46}$$

and the symmetric difference is

$$\frac{\partial f}{\partial x_i}(x^k) \approx \frac{f(x^k + hu_i) - f(x^k - hu_i)}{2h} \tag{47}$$

The symmetric difference involves twice as much computation, but it is significantly more stable numerically (the rates of converges are respectively $\mathcal{O}(h)$ for the forward method and $\mathcal{O}(h^2)$ for the symmetric method).

It is important to remember that $h$ need not be extremely small (remember the $\epsilon_{machine}$ limit!), but that it is good to choose the best $h$ for each coordinate separately. This is because different coordinate axes may have different measurement units and rates of variation of $\nabla f$, and it may be impossible to find one $h$ which is acceptable for all.

## 2.4 Coordinate descent

Here, the direction of descent $d^k$ is always one of the coordinate axes $u_{i^k}$. Hence $x^{k+1} = x^k + \alpha_k u_{i^k}$. Note that line search is necessary, and that the minimum can be on either side of $x^k$ so $\alpha_k$ can take negative values.

Theoretical and empirical results suggest that coordinate descent has similar convergence properties as the steepest descent method (i.e linear in the best case).

While in a general case coordinate descent is suboptimal, there are several situations when it is worth considering

1. When line minimization can be done analytically. This can save one the often expensive gradient computation.
2. When the coordinate axes affect the function value approximately independently, or (in statistics) when the coordinate axes are uncorrelated. Then minimizing along each axis separately is (nearly) optimal.
3. When there exists a natural grouping of the variables. Then one can optimize one group of variables while keeping the other constant. Again, we hope that the groups are "independent", or that optimizing one group at a time can be done analytically, or it's much easier than computing the gradient w.r.t all variables simultaneously. This idea is the basis of many *alternate minimization* methods, including the well known EM algorithm.

## 2.5 The simplex method

This is a method that does not evaluate derivatives. (Not to be confused with the *Simplex algorithm* of linear programming!) A set of $n-1$ points is maintained, and iterative replacement of the worst of them aims to move the simplex so that it encloses the optimum, and to simultaneously shrink it so that the optimum is bracketed with small tolerance.

```
SIMPLEX METHOD

Initialize $x_1, \dots x_{n+1}$ points in general position (i.e that enclose a non-zero volume)
Denote $x_{max}, f_{max}, f_{min}$ respectively $\underset{i}{\text{argmax}}\, f(x_i), \max_i f(x_i), \min_i f(x_i)$,
and $\hat{x}$ the mean of $\{x_{1:n+1}\} \setminus x_{max}$.
1. Reflection Compute $x_{ref} = \hat{x} + (\hat{x} - x_{max})$ and $f_{ref} = f(x_{ref})$.
2. If $f_{ref} > f_{max}$, $x_{new} = \text{argmin}(f((x_{ref} + \hat{x})/2), f((x_{max} + \hat{x})/2)$ and go to 5
3. If $f_{min} < f_{ref} \le f_{max}$, $x_{new} = x_{ref}$ and go to 5
4. If $f_{min} \ge f_{ref}$ try to expand
   $x_{exp} = x_{ref} + (x_{ref} - \hat{x})$, $x_{new} = \text{argmin}(f_{ref}, f(x_{exp}))$.
5. replace $x_{max}$ with $x_{new}$, then recompute $f_{min}, x_{max}, f_{max}$
```

The algorithm is not guaranteed to converge in all cases, but it can be shown that it works for convex $f$. The above is not the most efficient implementation, it's the one that's easier to read. Consult also NR for this algorithm.

## 3 Stopping criteria

**What we want** is to stop when $e^k \le tol$, where *tol* is some redefined tolerance, and $e^k$ is the current error w.r.t $f$ or $x$. The problem is that, not knowing $x^*$, the error $e^k$ cannot be computed, only estimated. Usually, by estimate, we mean an upper bound on the true error. To note that, if the problem is parameter estimation, we care about the value $x^*$ which represents the unknown parameter. For an optimization problem that is not related to statistical learning, it may be the case that the loss $f(x^*)$ is more important than the actual value of $x^*$.

In what follows, we shall see various recipes for stopping, that depend on progressively less information (and are less accuarate in the same measure).

1. *Newton steps* The best estimate of the distance to $x^*$ from the information usually available is the Newton step. Hence, one could use the rule
$$||\nabla^2 f(x^k)^{-1}\nabla f(x^k)|| \le tol \tag{48}$$
Often the above expression is too expensive to compute; you can save time by estimating it only every $L$ iterations, assuming that its values don't change much at one iteration. (This holds for the Newton steps as well, not only for the stopping test, however it's unsafe if the Hessian is not positive definite).
Practically, assuming that the direction of search incorporates a "good" variable scaling, (i.e NOT in steepest descent) one can stop when
$$||d^k|| \le tol \quad OR \quad ||\alpha^k d^k|| \; = \; ||x^{k+1} - x^k|| \le tol \tag{49}$$
The two are pretty much equivalent in the asymptotic regime, as $\alpha^k$ should be approximately 1 in that case. For steepest descent, the latter rule is used too, but it offers no guarantees.
2. *Gradient descent with information on the Hessian*
**Proposition 11** *If the smallest eigenvalue of* $\nabla^2 f(x) \ge m > 0$ *and* $||\nabla f(x)|| \le \epsilon$ *for* $x \in S$ *a neighborhood of* $x^*$*, then*
$$||x - x^*|| \le \frac{\epsilon}{m} \quad f(x) - f(x^*) \le \frac{\epsilon^2}{m} \tag{50}$$
**Proof** By first order Taylor expansion of $\nabla f$
$$\nabla f(x) - \underbrace{\nabla f(x^*)}_{0} \;\approx\; \nabla^2(x)(x - x^*) \tag{51}$$
$$\epsilon \;\ge\; ||\nabla f(x)|| \;\approx\; ||\nabla^2(x)(x - x^*)|| \;\ge\; m||x - x^*|| \tag{52}$$
$$\epsilon/m \;\ge\; ||x - x^*|| \tag{53}$$
Now the first order expansion of $f$ around $x$ gives us
$$f(x^*) \;\approx\; f(x) + \nabla f(x)(x^* - x) \tag{54}$$
$$f(x) - f(x^*) \;\approx\; ||\nabla f(x)(x^* - x)|| \tag{55}$$
$$\le\; ||\nabla f(x)||||x - x^*|| \tag{56}$$
$$\le\; \epsilon\frac{\epsilon}{m} \tag{57}$$

Proposition 11 highlights that in $n$ dimensions, the key to a good approximation of the distance to the optimum is an estimate of the minimum curvature of $f$ around the minimum (together with the assumption that the third and higher derivatives are negligible in the region around the minimum that we are considering). Having this, the local and computable magnitude of the gradient $\nabla f(x)$ gives us an upper bound on the error w.r.t both $x$ and $f$.
Estimating $m$ can be done either from specific knowledge about the problem or numerically; in addition to computing the Hessian, computing its smallest eigenvalue by iterative methods (Lanczos) is proportional to $n^2$.
So, knowing when to stop involves assumptions (or information) about (1) the gradient, (2) the Hessian (its eigenvalues), (3) the continuity of the Hessian. In practice, (1) is available, (2) can be evaluated numerically, (3) is usually assumed or derived from specific knowledge about the problem.
Hence, when $m$ is known, Proposition 11 gives us the desired upper bounds on the error.

3. *Gradient, Coordinate descent, etc with no information about the Hessian* If $m$ is not known, the the following rules of thumb help:
   (a) check that $\nabla^2 f \succ 0$ and, if feasible, that the third derivatives are not too high
   (b) if you care about the value $f(x^*)$, stop when the relative decrease $\left|1 - \frac{f(x^{k+1})}{f(x^k)}\right| < tol'$ with $tol \geq \epsilon_{machine}$. Note that this $tol'$ should be cca 1 order of magnitude smaller than the desired $tol$ (i.e we assume that in 10 steps convergence to $tol$ would be reached).

## 3.1 Stopping stochastic gradient

From results similar to Proposition 9 follows that the number of steps to reach $tol$ is $K = \Omega\left(\frac{1}{tol^2}\right)$. Proposition 9 applied to the special case of learning from a sample where $tol \sim \frac{1}{N}$ suggests that a constant number of passes over the sample $\mathcal{D}$ suffices to reduce the variance $Var\,\theta^k$ to the order of magnitude $\frac{1}{N}$.

Empirically, if we don't know anything about the constants involved, stochastic gradient is stopped when $\theta^k$, the change in the average gradient, becomes lower than a $tol'$.