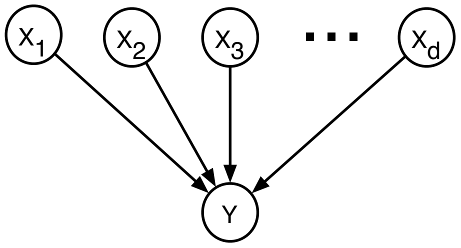
CSE 250a. Assignment 4

Out: *Tue Oct 27*
Due: *Thu Nov 3*

4.1 Gradient-based learning



Consider the belief network shown above with binary random variable $Y \in \{0, 1\}$ and conditional probability table (CPT):

$$P(Y\!=\!1|x_1, x_2, \ldots, x_d) \;=\; f\left(\sum_{i=1}^{d} w_i x_i\right) \;=\; f(\vec{w}\cdot\vec{x})\,.$$

Here, we assume that the weights $w_i \in \Re$ are real-valued parameters to be estimated from data and that $f\!:\!\Re \rightarrow [0,1]$ is a differentiable function that maps its real-valued argument to the unit interval.

Consider a data set of i.i.d. examples $\{(\vec{x}_t, y_t)\}_{t=1}^{T}$ where $\vec{x}_t = (x_{1t}, x_{2t}, \ldots, x_{dt})$ denotes the observed vector of values from the t^{th} example for the network's inputs. Also, as shorthand, let $p_t = P\left(Y\!=\!1|\vec{x}_t\right)$.

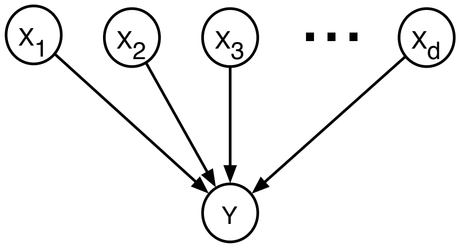
(a) Show that the gradient of the conditional log-likelihood $\mathcal{L} = \sum_t \log P(y_t|\vec{x}_t)$ is given by:

$$\frac{\partial \mathcal{L}}{\partial w_i} \;=\; \sum_{t=1}^{T}\left[\frac{f'(\vec{w}\cdot\vec{x}_t)}{p_t(1-p_t)}\right](y_t - p_t)x_{it},$$

where $f'(z)$ denotes the first derivative of $f(z)$. Intuitively, this result shows that the differences between observed values y_t and predictions p_t appear as error signals for learning.

(b) Show that the general result in part (a) reduces to the result in lecture when $f(z) = \left[1+e^{-z}\right]^{-1}$ is the sigmoid function.

4.2 Multinomial logistic regression



A simple generalization of logistic regression is to predict a discrete (but non-binary) label $y \in \{1, 2, \ldots, c\}$ from a real-valued vector $\vec{x} \in \mathcal{R}^d$. Here c is the number of classes. For the belief network shown above, consider the following parameterized conditional probability table (CPT):

$$P(Y\!=\!i|\vec{X}\!=\!\vec{x}) \;=\; \frac{e^{\vec{w}_i\cdot\vec{x}}}{\sum_{j=1}^{c} e^{\vec{w}_j\cdot\vec{x}}}.$$

The parameters of this CPT are the weight vectors $\vec{w}_i$ which must be learned for each possible label. The denominator normalizes the distribution so that the elements of the CPT sum to one.

Consider a data set of T examples $\{(\vec{x}_t, y_t)\}_{t=1}^{T}$, which you can assume to be identically, independently distributed (i.i.d.). As shorthand, let $y_{it} \in \{0, 1\}$ denote the $c\times T$ matrix of target assignments with elements

$$y_{it} \;=\; \begin{cases} 1 & \text{if} \quad y_t = i, \\ 0 & \text{otherwise.} \end{cases}$$

Also, let $p_{it} \in [0, 1]$ denote the conditional probability that the model classifies the tth example by the ith possible label:

$$p_{it} \;=\; \frac{e^{\vec{w}_i\cdot\vec{x}_t}}{\sum_{j=1}^{c} e^{\vec{w}_j\cdot\vec{x}_t}}.$$

The weight vectors can be obtained by maximum likelihood estimation using gradient ascent. Show that the gradient of the conditional log-likelihood $\mathcal{L} = \sum_t \log P(y_t|\vec{x}_t)$ is given by:

$$\frac{\partial\mathcal{L}}{\partial\vec{w}_i} \;=\; \sum_t (y_{it} - p_{it})\,\vec{x}_t.$$

Again, this result shows that the differences between observed values y_{it} and predictions p_{it} appear as error signals for learning.

4.3 Convergence of gradient descent

One way to gain intuition for gradient descent is to analyze its behavior in simple settings. For a one-dimensional function $f(x)$ over the real line, gradient descent takes the form:

$$x_{n+1} \;=\; x_n - \eta f'(x_n).$$

(a) Consider minimizing the function $f(x) = \frac{\alpha}{2}(x - x_*)^2$ by gradient descent, where $\alpha > 0$. (In this case, we know that the minimum occurs at $x = x_*$; our goal is to analyze the rate of convergence to this minimum.) Derive an expression for the error $\varepsilon_n = x_n - x_*$ at the n^{th} iteration in terms of the initial error ε_0 and the step size $\eta > 0$.

(b) For what values of the step size η does the update rule converge to the minimum at x_*? What step size leads to the fastest convergence, and how is it related to $f''(x_n)$?

In practice, the gradient descent learning rule is often modified to dampen oscillations at the end of the learning procedure. A common variant of gradient descent involves adding a so-called *momentum* term:

$$\vec{x}_{n+1} = \vec{x}_n - \eta \nabla f + \beta \left(\vec{x}_n - \vec{x}_{n-1}\right),$$

where $\beta > 0$. Intuitively, the name arises because the optimization continues of its own momentum (stepping in the same direction as its previous update) even when the gradient vanishes. In one dimension, this learning rule simplifies to:

$$x_{n+1} \;=\; x_n - \eta f'(x_n) + \beta(x_n - x_{n-1}).$$

(c) Consider minimizing the quadratic function in part (a) by gradient descent with a momentum term. Again, let $\varepsilon_n = x_n - x_*$ denote the error at the nth iteration. Show that the error in this case satisfies the recursion relation:

$$\varepsilon_{n+1} = (1 - \alpha\eta + \beta)\varepsilon_n - \beta\varepsilon_{n-1}.$$

(d) Suppose that the second derivative $f''(x^*)$ is given by $\alpha = 1$, the learning rate by $\eta = \frac{4}{9}$, and the momentum parameter by $\beta = \frac{1}{9}$. Show that one solution to the recursion in part (c) is given by:

$$\varepsilon_n = \lambda^n \varepsilon_0,$$

where ε_0 is the initial error and λ is a numerical constant to be determined. (Other solutions are also possible, depending on the way that the momentum term is defined at time $t = 0$; do not concern yourself with this.) How does this rate of convergence compare to that of gradient descent with the same learning rate ($\eta = \frac{4}{9}$) but no momentum parameter ($\beta = 0$)?

4.4 Newton's method

One way to gain intuition for Newton's method is to analyze its behavior in simple settings. For a twice-differentiable function $f(x)$ over the real line, Newton's method takes the form:

$$x_{n+1} \;=\; x_n - \frac{f'(x_n)}{f''(x_n)}.$$

(a) Consider the polynomial function $f(x) = (x - x_*)^{2p}$ for positive integers p, whose minimum occurs at $x = x_*$. Suppose that Newton's method is used to minimize this function, starting from some initial estimate x_0. Derive an expression for the error $\varepsilon_n = |x_n - x_*|$ at the n^{th} iteration in terms of the initial error ε_0.

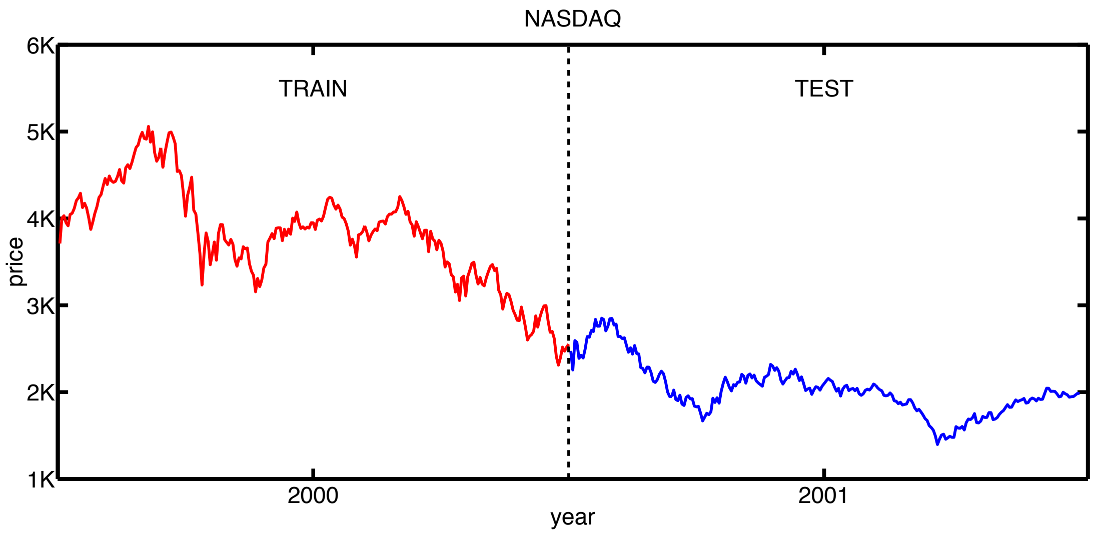
(b) For the function in part (a), how many iterations of Newton's method are required to reduce the initial error by a constant factor $\delta < 1$, such that $\varepsilon_n \leq \delta\varepsilon_0$? Starting from your previous answer, show that $n \geq (2p-1)\log(1/\delta)$ iterations are sufficient. (Hint: use the inequality that $\log z \leq z-1$ for $z > 0$.)

(c) Consider the function $f(x) = x_* \log(x_*/x) - x_* + x$, where $x_* > 0$. Show that the minimum occurs at $x = x_*$, and sketch the function in the region $|x - x_*| < x_*$.

(d) Consider minimizing the function in part (c) by Newton's method. Derive an expression for the relative error $\rho_n = (x_n - x_*)/x_*$ at the n^{th} iteration in terms of the initial relative error ρ_0. Note the rapid convergence (which is typical of Newton's method). For what range of initial values (for x_0) does Newton's method converge to the correct answer?

4.5 Stock market prediction

In this problem, you will apply a simple linear model to predicting the stock market. From the course web site, download the files `nasdaq00.txt` and `nasdaq01.txt`, which contain the NASDAQ indices at the close of business days in 2000 and 2001.



(a) Linear coefficients

How accurately can the index on one day be predicted by a linear combination of the three preceding indices? Using only data from the year 2000, compute the linear coefficients (a_1,a_2,a_3) that maximize the log-likelihood $\mathcal{L} = \sum_t \log P(x_t|x_{t-1}, x_{t-2}, x_{t-3})$, where:

$$P(x_t|x_{t-1}, x_{t-2}, x_{t-3}) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}\Big(x_t - a_1 x_{t-1} - a_2 x_{t-2} - a_3 x_{t-3}\Big)^2\right],$$

and the sum is over business days in the year 2000 (starting from the fourth day).

(b) Mean squared prediction error

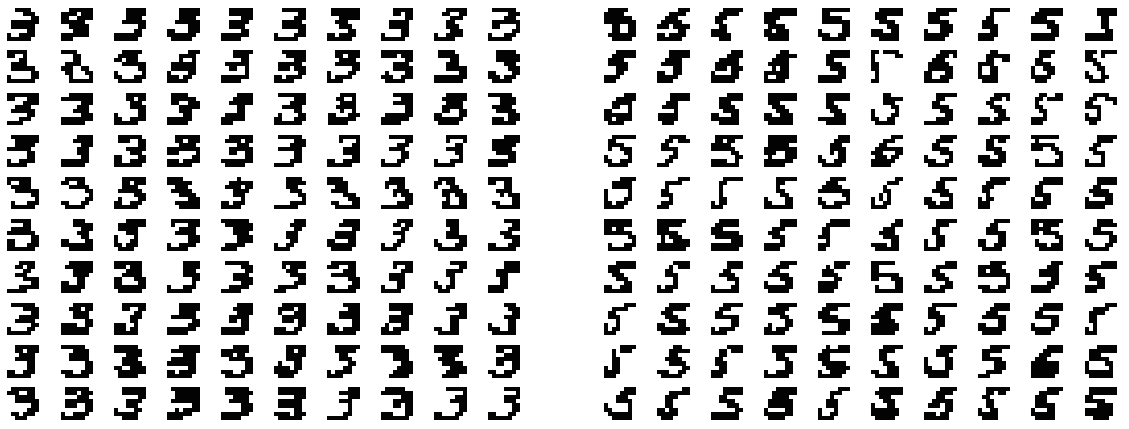
For the coefficients estimated in part (a), compare the model's performance (in terms of mean squared error) on the data from the years 2000 and 2001. Would you recommend this linear model for stock market prediction?

(c) Source code

Turn in a print-out of your source code. You may program in the language of your choice, and you may solve the required system of linear equations either by hand or by using built-in routines (e.g., in Matlab, NumPy).

4.6 Handwritten digit classification

In this problem, you will use logistic regression to classify images of handwritten digits. From the course web site, download the files `train3.txt`, `test3.txt`, `train5.txt`, and `test5.txt`. These files contain data for binary images of handwritten digits. Each image is an 8x8 bitmap represented in the files by one line of text. Some of the examples are shown in the following figure.



(a) Training

Perform a logistic regression (using gradient ascent or Newton's method) on the images in files `train3.txt` and `train5.txt`. Indicate clearly the algorithm used, and provide evidence that it has converged (or nearly converged) by plotting or printing out the log-likelihood on several iterations of the algorithm, as well as the percent error rate on the images in these files. Also, print out the 64 elements of your solution for the weight vector as an 8x8 matrix.

(b) Testing

Use the model learned in part (a) to label the images in the files `test3.txt` and `test5.txt`. Report your percent error rate on these images.

(c) Source code

Turn in a print-out of your source code. Once again, you may program in the language of your choice. You should write your own routines for computing the model's log-likelihood, gradient, and/or Hessian, as well as for updating its weight vector.