CS 189 Introduction to Machine Learning
Spring 2018 gs2

1 Descent

(a) Consider the setup of gradient descent: $x_{t+1} = x_t - \alpha \nabla f(x)$. Recall that for a function $f : \mathbb{R}^n \to \mathbb{R}$ that is convex and differentiable, and which is additionally L-smooth:
$f(y) \leq f(x) + \nabla f(x)^T (y-x) + \frac{L}{2}\|x-y\|_2^2$ for any x, y.
Then if we run gradient descent for k iterations with a fixed step size $\alpha = 1/L$, it will yield a solution $f(x^{(k)})$ which satisfies:

$f(x^{(k)}) - f(x^*) \leq \frac{\|x^{(0)} - x^*\|_2^2}{2\alpha k}$

What is the convergence rate? Hint: how many iterations are required to obtain $f(x^{(k)}) - f(x^*) \leq \varepsilon$?

(b) Now, consider the case of strong convexity, that is:
$f(y) \geq f(x) + \nabla f(x)^T (y-x) + \frac{\mu}{2}\|x-y\|_2^2$

How does assuming strong convexity impact the convergence rate? That is, how many iterations are required to obtain a value of x that is ε away from x^* if we run gradient descent for k iterations with a constant stepsize $\alpha = \frac{1}{L}$?
Hint: For x_{t+1}, the norm squared error is bounded by: $\|x_{t+1} - x^*\|_2^2 \leq (1 - \frac{\mu}{L})\|x_t - x^*\|_2^2$

(c) Above, we have seen how Gradient Descent works for certain functions. However,evaluating the gradient on all data points is expensive. This is one reason we use Stochastic Gradient Descent in practice. Specifically, recall that in SGD instead of using the full gradient, we use a noisy estimate of it. Specifically, we assume:

$$\tilde{g}(x) = \nabla(x) + \varepsilon \sim N(0, \Sigma)$$

We will now explore the role of this noise. For concreteness let $f(w) = \frac{1}{2}||Xw - y||_2^2$ i.e. we are trying to optimize the OLS loss function using SGD.

Recall that in pratice we usually choose η to be small (Discuss with the people around you why we want this. One reason can be see by considering what happens to the norm of the gradient as we approach the optimal value). That being said, there are also reasons to want high learning rates. To see this, consider the extreme case of $\eta = 0$. Then, we have $w^1 = w^0 - 0 = w^0$ and thus we never advance from our initial case. Formally show a stonger version of this by proving a lower bound on η if we want to make at least l progress on the kth step i.e. we want $||w^{k+1} - w^k||_2^2 \geq l$

HINT: If we prove an upper bound on $||w^{k+1} - w^k||_2^2$, we can then lower bound that upper bound with l.

2 Backprop: *Adapted from Fall 2015 Final Exam*

Suppose we have a neural network that takes in $P = 2$ input features, has $H = 4$ hidden units, and $N = 1$ output.

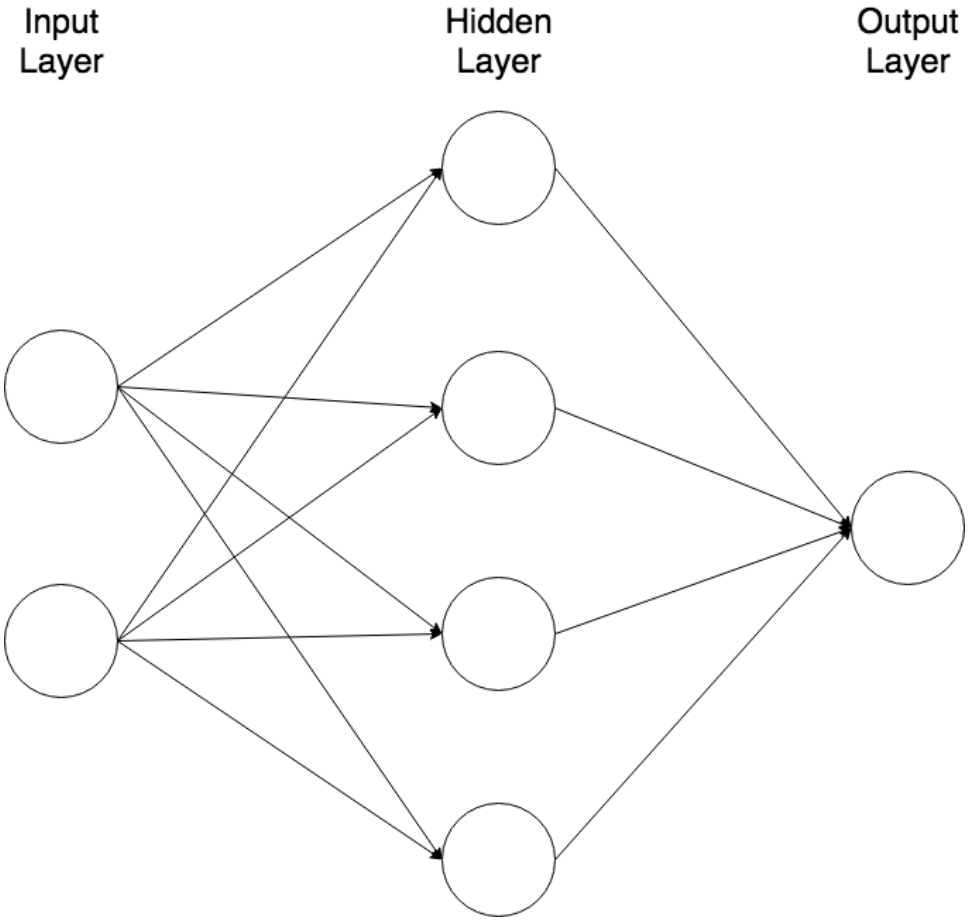


Figure 1: Unlabeled neural network as described above.

$f(x)$ is the activation function in the hidden layer, and $g(x)$ is the activation function in the output layer.

Let's define some notation!
$x \in \mathscr{R}^P$ is a feature vector
$s_j^h = \sum_{i=1}^{P} V_{i,j} x_i$ are inputs to the hidden layer
$h_j = f(s_j^h)$
$s_k^o = \sum_{j=1}^{H} W_{j,k} h_j$ are inputs to the output layer
$z_k = g(s_k^o)$
$V \in \mathscr{R}^{H \times P}$ are weights between input layer and hidden layer
$W \in \mathscr{R}^{N \times H}$ are weights between hidden layer and output layer

(a) On the diagram provided, label each of the pieces of notation defined above in their appropriate places. Focus on understanding the role of each variable in defining your neural net; don't worry too much about the exact details of how to annotate the diagram.

Now, draw in a bias term to both the input layer and the hidden layer and connect it appropriately with the other nodes.

(b) Suppose you want to include a bias term in both the input layer and hidden layer, how many weights will be trained in total now? What will be the dimensions of our weight matrices V and W if we include these bias terms? What if we had $P = p$ input features, l hidden layers each with $H = h$ hidden units, $N = n$ outputs, and bias terms in both the input and hidden layers?

(c) Use the chain rule to expand the following partial derivatives (no need to find the actual derivatives themselves):

$$\frac{\partial L}{\partial W_{j,k}} =$$
$$\frac{\partial L}{\partial V_{i,j}} =$$

(d) The softmax function, or normalized exponential function, is a generalization of the logistic function to handle multiclass classification. The softmax function "squashes" a N-dimensional vector s of arbitrary real values to a K-dimensional vector $\sigma(s)$ of real values in the range [0, 1] that add up to 1. The i-th value corresponds to the probability that the output is class i. The function is given by the following:

$$\sigma(s_j) = \frac{e^{s_j}}{\sum_{k=1}^{N} e^{s_k}} \qquad \text{for } j = 1...N$$

The cross entropy error is commonly paired with softmax activation in the output layer and is defined as:

$$L(z) = \sum_{k=1}^{N} y_k \ln(z_k)$$

Now, let $g(x)$ be the softmax function and let our loss function be the cross entropy loss. Calculate the partial derivative of L with respect to $W_{i,j}$. (Note: For part (c), you were not required to calculate the actual derivative. For this part, you must do so.)

3 Fisher's Linear Discriminant Analysis

(Inspired by section 4.3.3 of ESL.)

Suppose we have a binary classification problem for which we would like to find a linear estimator for–that is, given a random vector $\vec{X}$, we would like to find predictor $Z = w^\top \vec{X}$, s.t. we maximize $(E[Z|Y=1] - E[Z|Y=0])^2$.

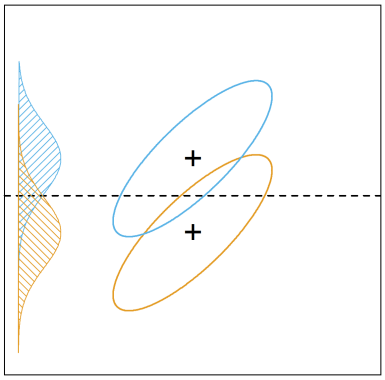
We define:
$\mu_i = E[X|Y=i]$
$\Sigma = (\vec{X} - \mu_i)(\vec{X} - \mu_i)^\top$, where for both classes $i = 0, 1$ we have the same covariance matrix.
$n_0 = n_1$, i.e. the number of data points in each class are the same.

Thus we have

$$\begin{aligned}
\max(E[Z|Y=1] - E[Z|Y=0])^2 &= \max(E[w^\top \vec{X}|Y=1] - E[w^\top \vec{X}|Y=0])^2 \\
&= \max(w^\top (E[\vec{X}|Y=1] - E[\vec{X}|Y=0]))^2 \\
&= \max(w^\top (\mu_1 - \mu_0))^2 \\
&= \max w^\top (\mu_1 - \mu_0)(\mu_1 - \mu_0)^\top w
\end{aligned}$$

(a) Given the above maximization problem, we know that $|w|$ can increase without bound unless we have some kind of normalization. One way we can normalize is to divide by the magnitude of w. What is the result of this normalization?

(b) Is the approach from part (a) equivalent to LDA? Refer to this plot from ESL to explain your conceptual answer. The crosses represent centroids. The ellipses represent an isocontour of the multivariate gaussian corresponding to the covariance matrix of a set of data. The dotted line represents the decision boundary. The bell curves represent a projection of the multivariate gaussians.

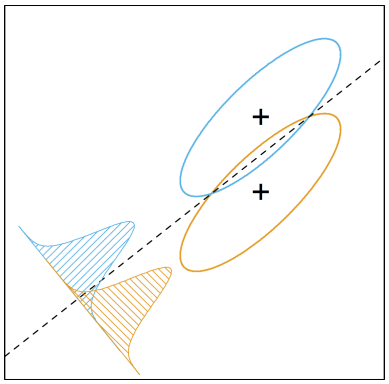


(c) Now, we will try normalizing instead by $\frac{1}{2}\text{var}[Z|Y=1] + \frac{1}{2}\text{var}[Z|Y=0]$. Simplify in terms of Σ and μ_i (refer to the definitions at the beginning of this problem).

(d) Prove that your approach from part (c) is indeed equivalent to the decision boundary of LDA you are familiar with by finding w s.t. $w^\top x = c$, for some constant c. (Take it that solving for w through Lagrange multipliers yields that $w \propto \Sigma^{-1}(\mu_1 - \mu_0)$.)

$(x-\mu_0)^\top \Sigma (x-\mu_0) = (x-\mu_1)^\top \Sigma (x-\mu_1) = \ldots$

(e) Interpret why the approach from parts (c) and (d) are conceptually equivalent to LDA? Refer to this plot from ESL to explain your answer.



4 Logistic Regression

Consider the log-likelihood function for logistic regression

$$\ell(\theta) = \sum_{i=1}^{m} y^{(i)} \ln h(x^{(i)}) + (1 - y^{(i)}) \ln(1 - h(x^{(i)}))$$

Find the Hessian H of this function , and show that for every z, it holds true that

$$z^\top H z \leq 0$$

(Hint: You might want to start by showing that $\sum_i \sum_j z_i x_i x_j z_j = (x^\top z)^2$)