**Neural Networks for Signal Processing-I**

**Prof. Shayan Srinivasa Garani**

**Department of Electronic Systems Engineering**

**Indian Institute of Science, Bengaluru**

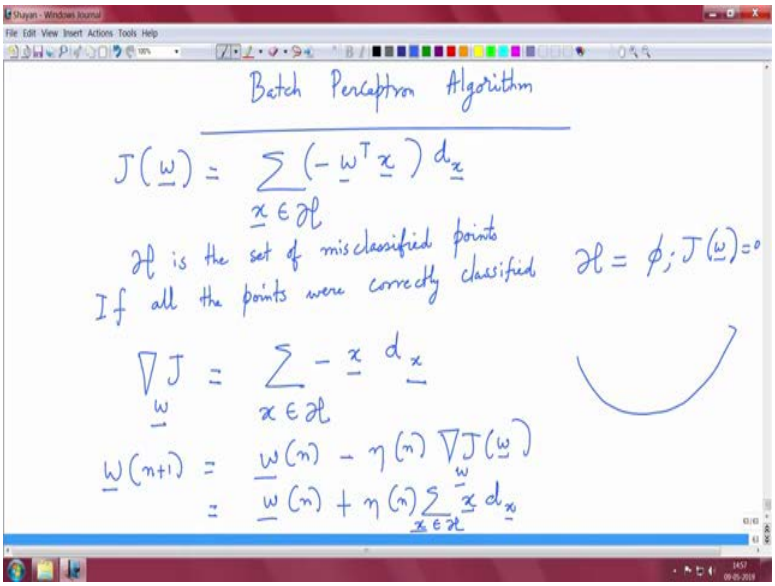
**Lecture - 10**

**Batch Perceptron Algorithm**

So far, we've explored the perceptron algorithm, which operates in an online fashion. This means we process each data point sequentially, adjusting the weight vector based on whether the point lies on one side of the hyperplane or the other.

Now, consider a scenario where we group all misclassified data points together and update the weight vector in a batch mode. Instead of adjusting the weight after each individual point, we update it once based on all misclassified points collectively. This approach introduces another algorithm known as the Batch Perceptron Algorithm.

(Refer Slide Time: 01:19)




Let's delve into the cost function for this approach, denoted as J(w), where w represents the weight vector. This function is defined as the sum over all points x that are misclassified, denoted by H, where H is the set of misclassified points. The cost function is formulated as $J(w) = \sum_{x \in H} -w^\top x \cdot d$, where d corresponds to the desired classification of the vector x.

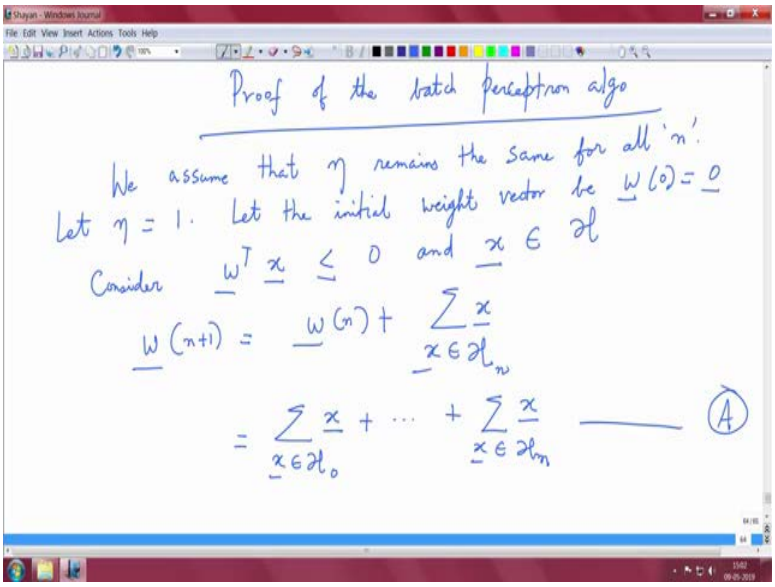
To elaborate, if $w^\top x > 0$, then x belongs to class $c_1$ with d = 1. Conversely, if $w^\top x \leq 0$, x belongs to class $c_2$ with d = -1. In the case of misclassification, where $w^\top x$ and d have opposite signs, the product $-w^\top x \cdot d$ accumulates a positive cost in the function J.

If all the points were correctly classified, naturally, our set of misclassified points H would be empty. Thus, the error would be zero, and the cost function J(w) would evaluate to zero. However, when misclassifications occur, we accumulate a positive cost.

To address this, we compute the gradient of the cost function with respect to the weight vector w. The gradient, derived from the sum of $-x \cdot d$ over all misclassified points x, guides us towards adjusting the weight vector. This update strategy is inspired by the principles of gradient descent, aiming to find an optimal point that minimizes the cost function. While the specific cost function here may not be quadratic, the approach mirrors the balance-seeking process, where iterative adjustments lead towards an optimal solution.

What I can do is adjust my weight at time step n+1 to be my weight at time step n - $\eta(n)$ times the gradient of the cost function with respect to the weight. This leads to the update rule where the weight at time n+1 is the weight at time n + $\eta(n)$ times the gradient, which we replace with the sum of x times the desired output d(x) for all x in the misclassified set.

(Refer Slide Time: 07:04)




This forms the basis of the batch perceptron algorithm, and we will soon see how this update rule ensures correct adjustments. The idea is straightforward, and if we have understood the

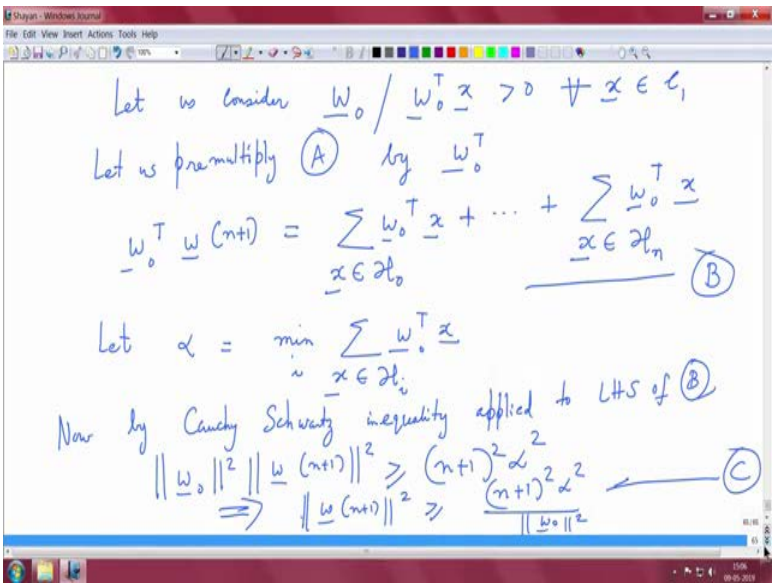
proof of convergence for the perceptron algorithm, the proof for the batch perceptron algorithm follows a similar path. We will revisit this for thoroughness and completeness.

Let's delve into the proof of the batch perceptron algorithm. Now, we assume that the parameter $\eta$ remains consistent for all iterations n, but it's important to distinguish the time step n in the batch perceptron algorithm from that in the perceptron algorithm. In the perceptron algorithm, which operates online, each data point corresponds to a specific time step n. However, in the batch perceptron algorithm, we aggregate all misclassified data points, compute a cost function, and then evaluate the gradient of that function. Hence, it operates in epochs, where n denotes the epoch.

Although we refer to it as time step n, it's more akin to an epoch because it involves grouping all misclassified points and computing the derivative. Let's assume $\eta$ remains fixed, similar to what we did previously. Let the initial weight vector be $w_0$, which is the all-zero vector. Now, consider the scenario where $w^T x \leq 0$ for x belonging to the set of misclassified samples from class $c_1$.

So, when w at time step n + 1 is updated and w at time step n $\leq$ 0, this indicates a misclassification. Now, w at time step n+1 is adjusted by summing over all x belonging to the misclassified set of points at time step n. Let's denote this misclassified set as $H_n$. Now, let's expand this recursively, similar to how we approached recursion earlier. This involves summing over all misclassified data points from time step 0 to n.

(Refer Slide Time: 11:21)



The set $H_n$ varies over time, which is a crucial distinction in the convergence proof of the batch perceptron algorithm. Unlike the perceptron convergence proofs that consider individual sample points, here we focus on the evolving set of misclassified points, necessitating a clear distinction.

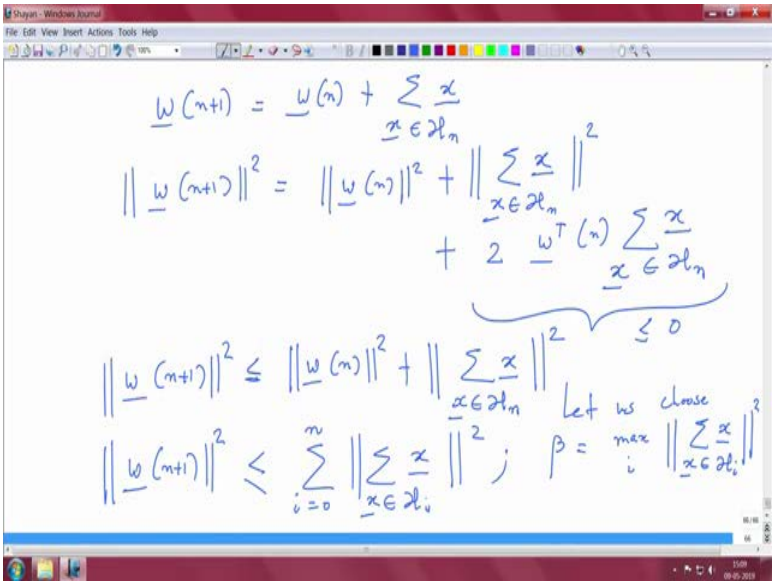
Now, let's consider the existence of $w_0$ (or $w_{optimal}$, or $w^*$, whichever notation suits, I'll use $w_0$ with the subscript). Suppose $w_0$ is such that $w_0^T x > 0$ for all x belonging to class $c_1$. Let's pre-multiply the equation $w_0^T w_{n+1}$ (let's denote this as Equation A).

Pre-multiplying Equation A by $w_0^T$, we get $w_0^T w_{n+1} = \sum w_0^T x$ for all x belonging to the set $H_0$, + ..., summing over $w_0^T x$ for x in $H_n$. From here, we proceed with the same straightforward steps. Let's assume $\alpha$ is the minimum of these sums. $\alpha$ is the minimum over all i of $\sum w_0^T x$ for x belonging to some set $H_i$ in the process of this update.

Now, applying Cauchy-Schwarz inequality to the left-hand side of Equation B, we have $|w_0|^2 \cdot |w_{n+1}|^2 \geq (n+1)^2 \alpha^2$.

Rearranging these terms, we get $|w_{n+1}|^2 \geq \frac{(n+1)^2 \alpha^2}{|w_0|^2}$. Let's denote this as Equation C. We proceed straightforwardly as in the previous proof, but we need to carefully consider how the sets evolve with each epoch.

(Refer Slide Time: 16:06)



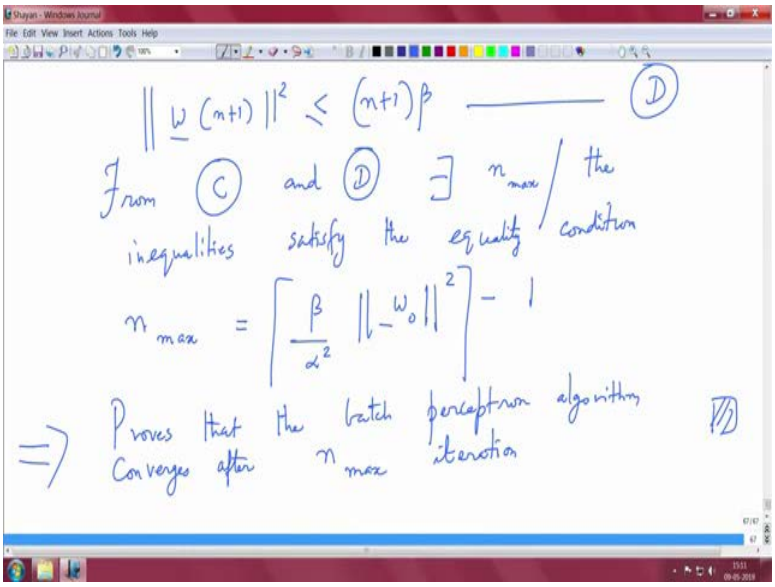

Now, considering its recursion, $w_{n+1} = w_n + \sum_{x \in H_n} x$, we take the norm squared on both

sides. $|w_{n+1}|^2$ becomes $|w_n|^2 + \sum_{x \in H_n} |x|^2 + 2w_n^T \sum_{x \in H_n} x$. However, since we assume this quantity is initially less than or equal to 0, $|w_{n+1}|^2$ is less than or equal to $|w_n|^2 + \sum_{x \in H_n} |x|^2$.

Recursively applying this as before, we end up with $|w_{n+1}|^2 \leq \sum_{i=0}^{n} \sum_{x \in H_i} |x|^2$.

Now, let's define a quantity β as the maximum over all i of the norm of $\sum_{x \in H_i} x$, where $x$ belongs to the set of misclassified points at time step i.

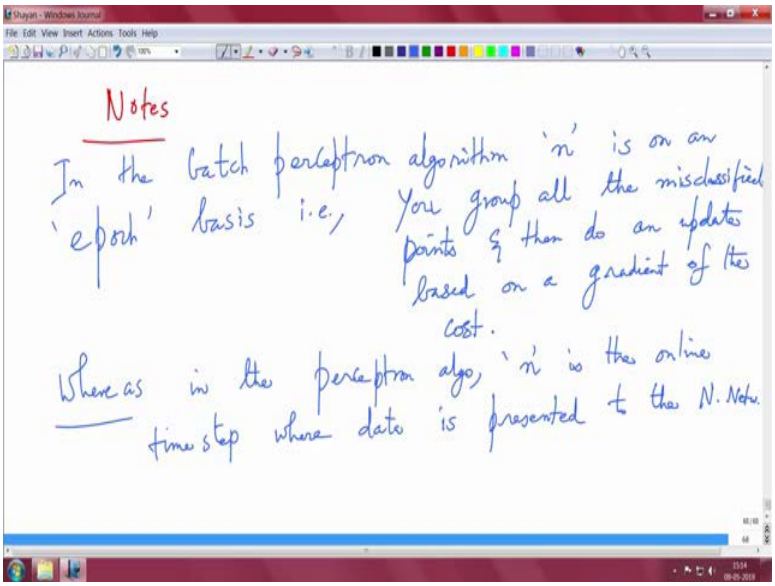
(Refer Slide Time: 19:14)




We arrive at the Euclidean norm squared at time step n+1, which is at most (n+1) times β. Let's denote this equation as D. Now, we have D and an earlier inequality equation C. From C and D, there exists an $n_{max}$, the maximum number of epochs, such that the inequalities satisfy the condition of equality. Calculating this, $n_{max}$ exactly equals $\frac{\beta}{\alpha^2 |w_0|^2 - 1}$, and you can round this quantity to the nearest integer to ensure all conditions are satisfied. This proves that the batch perceptron algorithm converges after $n_{max}$ iterations.

One crucial note to emphasize, from an implementation perspective, marked as Notes, is that in the batch perceptron algorithm, n operates on an epoch basis. Here, all misclassified points are grouped together, and updates are made based on the gradient of the cost function. Contrarily, in the ordinary perceptron algorithm, n represents the online time step where data is sequentially presented to the neural network. This distinction is pivotal, and when coding, careful consideration of these time steps is essential.

(Refer Slide Time: 21:35)




So far, we have covered both the perceptron and the batch perceptron algorithms. The choice between these algorithms depends on the application you are interested in, whether you prefer an online approach or a batch mode basis. It's worth noting that with batch processing, you can parallelize the algorithm, which offers natural advantages. When we delve into discussions about the multilayer perceptron or the backpropagation algorithm, we will explore the broader concepts of batch and online learning, discussing their nuances.