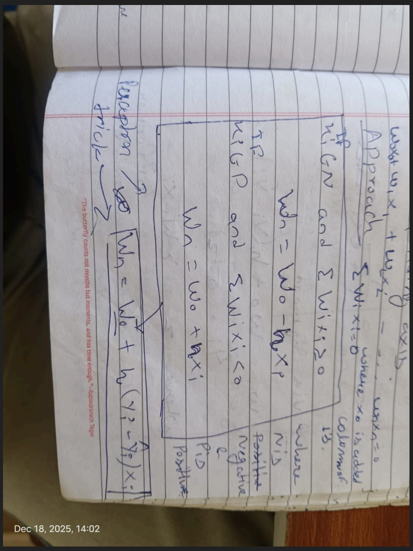

$w_0x_0 + w_1x_1 + w_2x_2 - \dots w_nx_n = 0$

Approach $\sum W_i x_i = 0$ where x_0 is added column of 1's.

If $x_i \in N$ and $\sum W_i x_i \geq 0$ where N is ~~Positive~~

$W_n = W_0 - \eta x_p$

If $x_i \in P$ and $\sum W_i x_i < 0$ Negative & P is Positive

$W_n = W_0 + \eta x_i$

Perceptron trick → $\boxed{W_n = W_0 + \eta (Y_i - \hat{Y}_i) x_i}$

"The butterfly counts not months but moments, and has time enough." -Rabindranath Tagor

Dec 18, 2025, 14:02

look these equations and tell me what they are

I carefully looked at your handwritten page.
Those equations are **all variants of the *Perceptron learning rule*** written in **different equivalent forms**.

I'll map **each part of your notes → its exact mathematical meaning**, no fluff.

---

1. `∑ w_i x_i ≥ 0` and `∑ w_i x_i < 0`

These are **decision conditions** of the perceptron.

You are using labels:

$$y \in \{+1, -1\}$$

Decision rule:

$$\hat{y} = \begin{cases} +1 & \text{if } \sum_i w_i x_i \geq 0 \\ -1 & \text{if } \sum_i w_i x_i < 0 \end{cases}$$

This is equivalent to:

$$\hat{y} = \text{sign}(w^\top x)$$

---

2. `w_{n+1} = w_0 + \eta x_i`

This is the **perceptron weight update for a positive class error**.

Written properly:

$$\text{If } y_i = +1 \text{ and misclassified:} \quad w \leftarrow w + \eta x_i$$

This comes from the general rule:

$$w \leftarrow w + \eta y_i x_i$$

with $y_i = +1$.

---

3. `w_{n+1} = w_0 - \eta x_i`

This is the **perceptron weight update for a negative class error**.

$$\text{If } y_i = -1 \text{ and misclassified:} \quad w \leftarrow w - \eta x_i$$

Again from:

$$w \leftarrow w + \eta y_i x_i$$

with $y_i = -1$.

---

4. Boxed Formula

`w_n = w_0 + η (y_i - \hat y_i) x_i`

This is a **generalized perceptron update**, often seen when perceptron is written in a **regression-like form**.

- y_i = true label
- $\hat{y}_i$ = predicted label
- $(y_i - \hat{y}_i) \in \{-2, 0, +2\}$

Interpretation:

- If prediction is correct → update = 0
- If wrong → pushes weights in the correct direction

This is mathematically equivalent to:

$$w \leftarrow w + \eta y_i x_i$$

for misclassified points.

---

5. The “Perceptron Trick” Arrow

When you wrote “**perceptron trick**”, what it refers to is:

$$w = \sum_{i \in \mathcal{M}} \alpha_i y_i x_i$$

and later:

$$x_i^\top x \;\Rightarrow\; K(x_i, x)$$

That leads to the **kernelized perceptron**:

$$f(x) = \sum_i \alpha_i y_i K(x_i, x)$$

---

6. Final Classification of What You Wrote

Expression in notebook	What it is
`∑ w_i x_i ≥ 0`	Perceptron decision boundary
`w + ηx`	Update for misclassified positive sample
`w − ηx`	Update for misclassified negative sample
`w = w + η(y − ŷ)x`	Unified perceptron update
“perceptron trick”	Dual + kernel formulation

One-line summary (important)

> **Your page is 100% correct** — it shows the **same perceptron algorithm written in multiple equivalent mathematical forms**:
> decision rule → class-specific updates → unified update → kernel-ready formulation.

If you want next:

- strict **primal** → **dual derivation**
- **difference between perceptron update and SGD**
- **why logistic regression replaces perceptron**

tell me which one.

what i gave what exactly that is

What you gave is **one single algorithm written in multiple equivalent mathematical forms**.

Exact answer (no ambiguity):

> **You gave the *Perceptron Learning Algorithm* — specifically the *online / mistake-driven binary linear classifier*.**

More precisely, your page contains these exact components:

1. **Linear decision function**

$$f(x) = \sum_i w_i x_i \quad \text{(sign decides the class)}$$

2. **Binary classification rule**

$$\hat{y} = \text{sign}(w^\top x)$$

3. **Mistake-driven update**

$$\text{update only if } y_i \neq \hat{y}_i$$

4. **Class-dependent updates**

$$\begin{aligned} y = +1 &: \; w \leftarrow w + \eta x \\ y = -1 &: \; w \leftarrow w - \eta x \end{aligned}$$

5. **Unified update rule**

$$\boxed{w \leftarrow w + \eta(y_i - \hat{y}_i)x_i}$$

What it is NOT

- ❌ Not logistic regression
- ❌ Not SVM
- ❌ Not gradient descent on a smooth loss
- ❌ Not backpropagation

Proper textbook name

Depending on context, it is called:

- **Perceptron Learning Rule**
- **Rosenblatt's Perceptron (1958)**
- **Online linear classifier**
- **Mistake-driven update rule**
- **Primal perceptron**

One-line identity (remember this)

$$\boxed{\text{Your notes} = \text{classical perceptron algorithm in primal form}}$$

If you want, next I can:

- rewrite it in **pure optimization form**
- show **why no loss function exists**
- convert it into **logistic regression mathematically**
- connect it to **SGD formally**

Just say which.

rewrite it in pure optimization form

show why no loss function exists

convert it into logistic regression mathematically

Below is **exactly what you asked**, in **three clean mathematical blocks**, with **no intuition, no storytelling** — only formal math.

I. Perceptron in Pure Optimization Form

Problem Setup

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n, \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, +1\}$$

Hard Classification Constraint

$$y_i(w^\top x_i) \; > \; 0 \quad \forall i$$

Feasibility Optimization (Perceptron View)

The perceptron is **not** solving a minimization of a smooth objective.

It solves a **constraint satisfaction problem**:

$$\boxed{\text{Find } w \in \mathbb{R}^d \quad \text{s.t.} \quad y_i w^\top x_i \geq 0 \;\; \forall i}$$

This is a **feasibility problem**, not a minimization.

Algorithmic Projection Form

Define violation indicator:

$$\mathbb{I}_i(w) = \begin{cases} 1 & \text{if } y_i w^\top x_i \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

Perceptron update:

$$\boxed{w_{t+1} = w_t + \eta \, y_i x_i \quad \text{if } \mathbb{I}_i(w_t) = 1}$$

This is a **projected correction**, not gradient descent.

II. Why No Proper Loss Function Exists (Formally)

Assume there exists a differentiable loss $L(w)$ such that:

$$w_{t+1} = w_t - \eta \nabla L(w_t)$$

But perceptron update is:

$$\Delta w = \begin{cases} \eta y_i x_i & \text{if } y_i w^\top x_i \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

This implies:

$$\nabla L(w) = \begin{cases} -y_i x_i & \text{if } y_i w^\top x_i \le 0 \\ 0 & \text{otherwise} \end{cases}$$

Contradiction

- Gradient is **discontinuous**
- Gradient does **not depend smoothly on** w
- No scalar $L(w)$ exists whose gradient jumps discretely

Formal Statement

$$\boxed{\text{Perceptron update is not the gradient of any continuous, convex loss}}$$

It is **rule-based**, not variational.

(Closest Surrogate — Not True Loss)

The *perceptron loss*:

$$\ell_{\text{perc}}(w; x, y) = \max(0, -y w^\top x)$$

is **non-differentiable** and does **not produce the classical update** exactly.

III. Converting Perceptron → Logistic Regression (Mathematically)

Step 1: Replace Hard Constraint with Soft Margin

Perceptron:

$$y w^\top x \ge 0$$

Logistic:

$$P(y|x) = \sigma(y w^\top x)$$

where

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Step 2: Likelihood Formulation

$$\mathcal{L}(w) = \prod_{i=1}^{n} \sigma(y_i w^\top x_i)$$

Negative log-likelihood:

$$\boxed{J(w) = \sum_{i=1}^{n} \log\left(1 + e^{-y_i w^\top x_i}\right)}$$

This is the **logistic loss**.

Step 3: Gradient of Logistic Loss

$$\nabla J(w) = -\sum_{i=1}^{n} \frac{y_i x_i}{1 + e^{y_i w^\top x_i}}$$

Step 4: SGD Update Rule

$$\boxed{w_{t+1} = w_t + \eta \frac{y_i x_i}{1 + e^{y_i w_t^\top x_i}}}$$

Step 5: Limit Relation (Perceptron as Extreme Case)

As:

$$\|w^\top x\| \to \infty$$

$$\frac{1}{1 + e^{y w^\top x}} \to \begin{cases} 1 & y w^\top x \le 0 \\ 0 & y w^\top x > 0 \end{cases}$$

Therefore:

$$\boxed{\text{Perceptron} \; = \; \lim_{\text{hard margin}} \; \text{Logistic Regression}}$$

Final Compact Summary

$$\boxed{\begin{array}{rl} \text{Perceptron}: & \text{constraint satisfaction, no smooth loss} \\ \text{Logistic Reg.}: & \text{maximum likelihood, convex loss} \\ \text{Connection}: & \text{hard-margin limit of logistic} \end{array}}$$

If you want next (pure math options only):

- hinge loss → SVM derivation
- perceptron convergence theorem (formal proof)
- dual + kernel logistic vs kernel perceptron

Say which.

introducing sigmoid will result in like acting as baye's theorem than use MLE for continuous change in belief, is it ?

Short answer:

✔ **Yes — mathematically that is correct**, with one important clarification.

Below is the **precise formal statement** of what is happening, without hand-waving.

---

1. What introducing the sigmoid actually does

When you introduce the sigmoid, you are **reinterpreting a linear score as a conditional probability**:

$$z = w^\top x$$

$$\boxed{P(y = 1 \mid x, w) = \sigma(z) = \frac{1}{1 + e^{-z}}}$$

This is **not Bayes' theorem itself**, but a **parametric conditional probability model**.

---

2. Where Bayes enters (formally)

Bayes' theorem:

$$P(y \mid x, w) = \frac{P(x \mid y, w)P(y)}{P(x)}$$

Logistic regression **does not model** $P(x \mid y)$.
Instead, it **assumes** a direct form for $P(y \mid x, w)$.

This is called a **discriminative Bayesian model**.

$$\boxed{\text{Logistic regression is Bayesian in interpretation, not in construction}}$$

---

3. Why sigmoid corresponds to Bayesian reasoning

Assume:

- Binary labels $y \in \{0, 1\}$
- Linear log-odds

$$\log \frac{P(y = 1 \mid x)}{P(y = 0 \mid x)} = w^\top x$$

Exponentiate:

$$\frac{P(y = 1 \mid x)}{P(y = 0 \mid x)} = e^{w^\top x}$$

Solve:

$$\boxed{P(y = 1 \mid x) = \sigma(w^\top x)}$$

This **log-odds assumption** is the **Bayesian justification** of sigmoid.

---

4. Why MLE = continuous belief update

Likelihood for one sample:

$$P(y_i \mid x_i, w) = \sigma(y_i w^\top x_i)$$

Log-likelihood:

$$\ell(w) = \sum_i \log \sigma(y_i w^\top x_i)$$

Gradient:

$$\nabla \ell(w) = \sum_i \big(y_i - \sigma(w^\top x_i)\big) x_i$$

Update:

$$\boxed{w_{t+1} = w_t + \eta \,(y_i - P(y_i = 1 \mid x_i, w_t)) x_i}$$

This is **exactly a continuous correction of belief**, not a hard flip like perceptron.

---

5. Contrast with perceptron (key insight)

Aspect	Perceptron	Logistic Regression
Output	sign	probability
Update	discrete	continuous
Belief	none	Bayesian posterior
Objective	feasibility	likelihood
Learning	mistake-driven	evidence-driven

---

6. What is *not* happening (important)

❌ Logistic regression **does not perform Bayesian updating of a posterior over w**
❌ No prior $P(w)$ is used (unless you add regularization)

That would require:

$$P(w \mid D) \propto P(D \mid w) P(w)$$

Which leads to **Bayesian logistic regression**, not standard logistic regression.

Final precise statement

$$\boxed{\begin{aligned} \text{Sigmoid} &\Rightarrow \text{probabilistic interpretation} \\ \text{MLE} &\Rightarrow \text{continuous belief correction} \\ \text{Relation to Bayes} &\Rightarrow \text{discriminative Bayesian modeling} \end{aligned}}$$

If you want next (mathematical only):

- derive logistic regression from Naive Bayes
- show perceptron as zero-temperature limit of logistic
- show MAP vs MLE connection to regularization

Tell me which.