コーディングだけじゃなく、

学習も助けてくれる Codex！

Codex Meetup Osaka #1

@SORA_LNV

SORA

X: @SORA_LNV

NVIDIA大好きな技術よわよわ学生

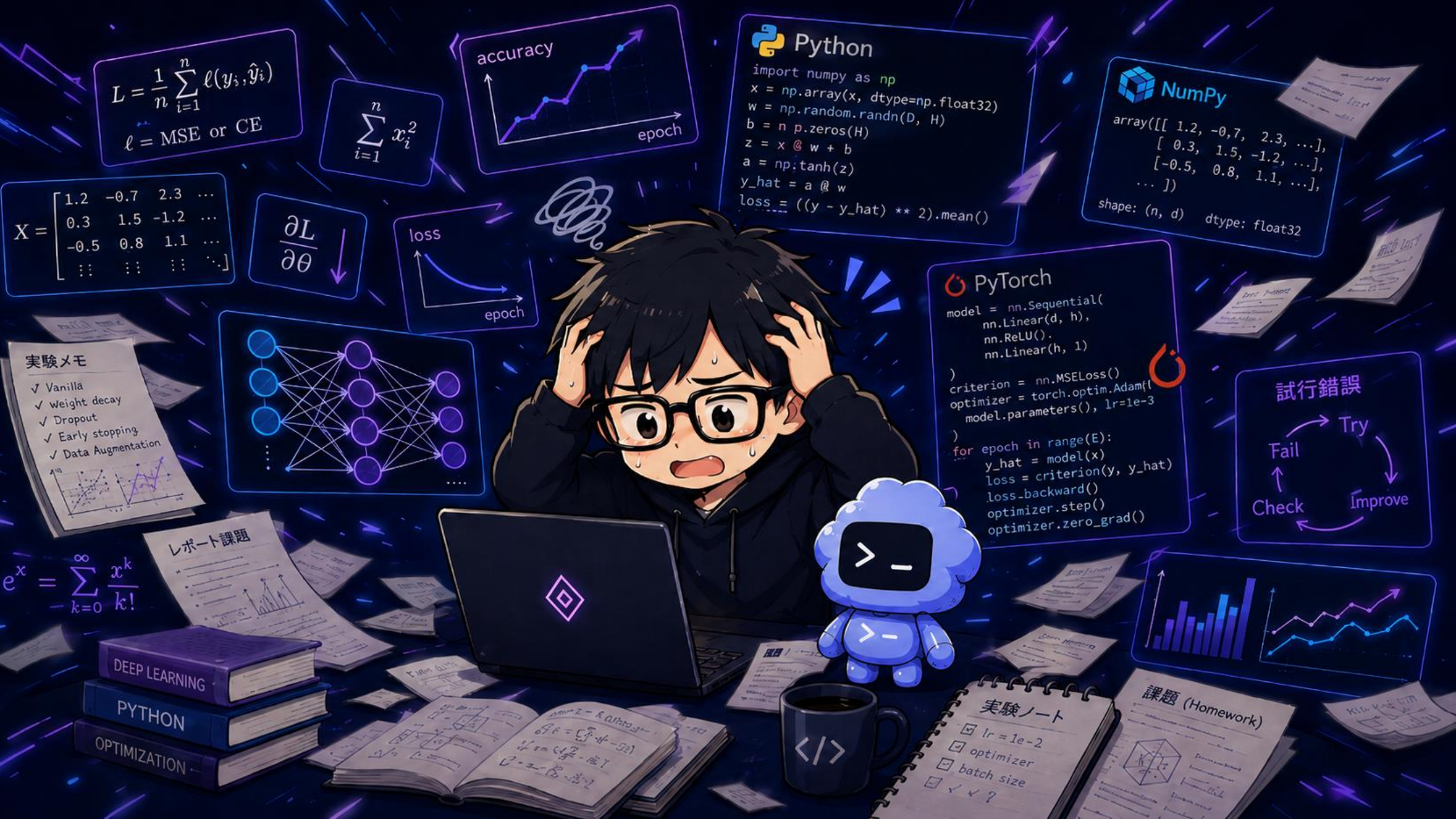
ごまあざらしかわいい

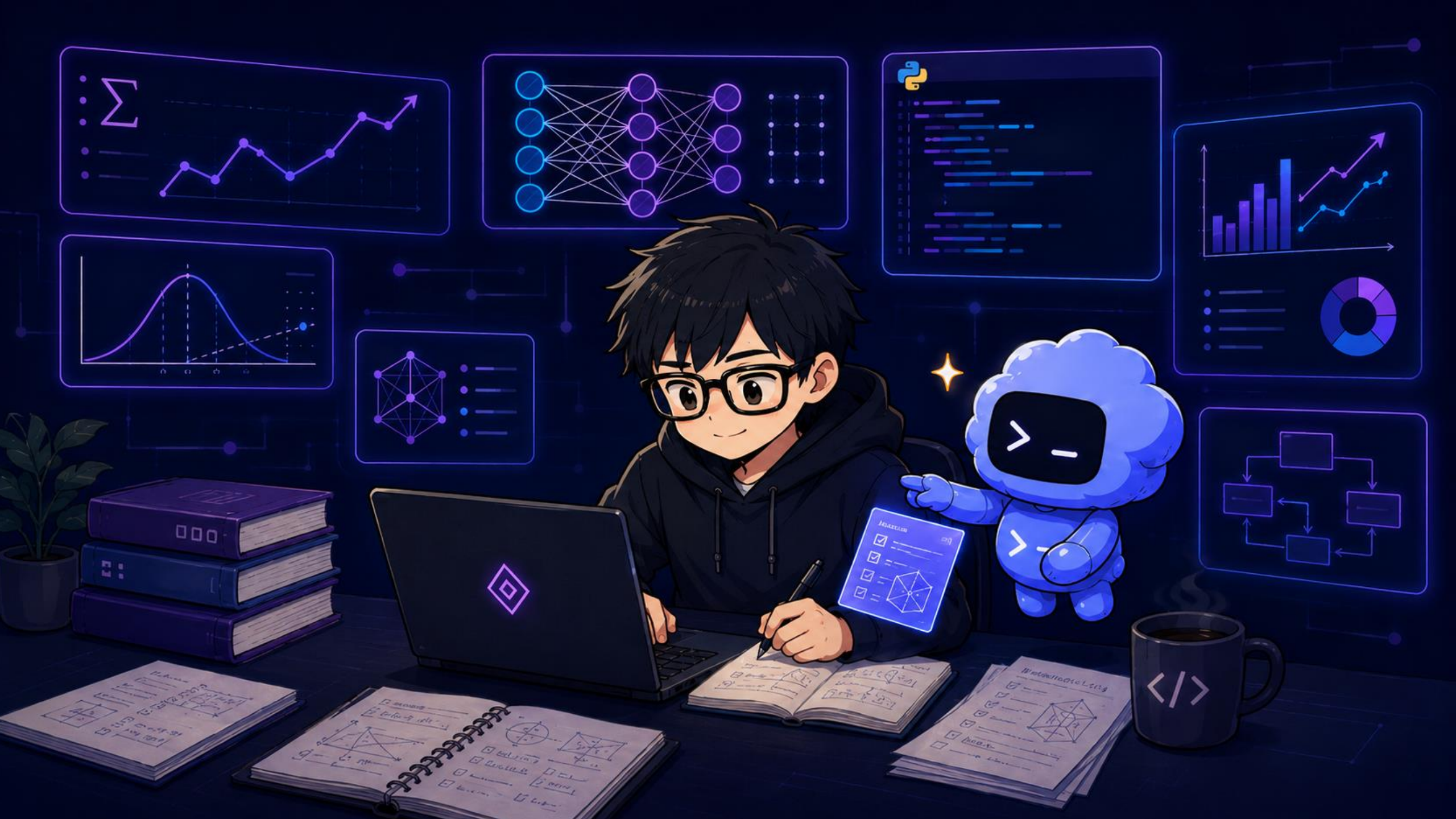
今日のテーマ

Codexは、
コード生成だけの道具ではない

- 学習環境を整える
- 理解しやすい形に変換する
- 復習できる資料を作る
- 考えるための相棒として使う

✧ 難しい講義でも、**学習の回転数**を上げられる ✧


$L = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \hat{y}_i)$
ℓ = MSE or CE
$\sum_{i=1}^{n} x_i^2$
accuracy
epoch
Python
import numpy as np
x = np.array(x, dtype=np.float32)
w = np.random.randn(D, H)
b = n p.zeros(H)
z = x @ w + b
a = np.tanh(z)
y_hat = a @ w
loss = ((y - y_hat) ** 2).mean()
NumPy
array([[1.2, -0,7, 2.3, ...],
[0.3, 1.5, -1.2, ...],
[-0.5, 0.8, 1.1, ...],
...])
shape: (n, d) dtype: float32
$X = \begin{bmatrix} 1.2 & -0.7 & 2.3 & \cdots \\ 0.3 & 1.5 & -1.2 & \cdots \\ -0.5 & 0.8 & 1.1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$
$\frac{\partial L}{\partial \theta}$
loss
epoch
PyTorch
model = nn.Sequential(
nn.Linear(d, h),
nn.ReLU(),
nn.Linear(h, 1)
)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(
model.parameters(), lr=1e-3
)
for epoch in range(E):
y_hat = model(x)
loss = criterion(y, y_hat)
loss.backward()
optimizer.step()
optimizer.zero_grad()
実験メモ
✓ Vanilla
✓ weight decay
✓ Dropout
✓ Early stopping
✓ Data Augmentation
試行錯誤
Try
Fail
Check
Improve
レポート課題
$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$
DEEP LEARNING
PYTHON
OPTIMIZATION
実験ノート
lr = 1e-2
optimizer
batch size
課題 (Homework)



なぜこの使い方をしたのか

受講中の深層学習系の講義が、想像以上に手強かった

情報量が多い

分野も用語も多くて、
どこから手をつければ
いいかわからない…

数式・コード・図が一気に出てくる

数式もコードも図も
一度に出てきて、
頭が追いつかない…

理解する前に、整理で疲れる

理解する前に整理で
エネルギーを使い切って、
学ぶ体力が削られる…

“理解する前の面倒”を減らしたかった

Codexで作った学習前処理スキル

講義ノートを自分用に整える skill

1
原本は必ず保存
原本ディレクトリ

2
派生ノートを作成
jp_
c_

3
環境差分を吸収
if文でパス分岐

4
一時ファイルを整理
.study_ops

ミスを指摘するたびに、
skill の精度が上がる

英日混在ノートを、読みやすい日本語版にする

Before

英日混在 / 読みにくい

日本語説明

正則化は、重みが大きくなりすぎることを抑えて、未知データに対する性能を安定させる工夫です。

English Translation

Regularization discourages excessively large weights and helps the model perform more stably on unseen data.

Code

```
# L2正則化: 重みwが大きくなりすぎるほどペナルティを足す
mse = ((y_pred - y_true) ** 2).mean()
l2_penalty = weight_decay * (w ** 2).sum()
loss = mse + l2_penalty

# lossを小さくする方向を自動微分で計算する
loss.backward()
```

After

日本語中心 / 読み返しやすい

日本語版

正則化は、重みが大きくなりすぎることを抑えて、未知データでも安定して使えるようにする工夫です。

Code

```
# L2正則化: 大きすぎる重みにペナルティを加える
mse = ((y_pred - y_true) ** 2).mean()
l2_penalty = weight_decay * (w ** 2).sum()
loss = mse + l2_penalty

# lossを小さくする方向を自動微分で求める
loss.backward()
```

英語部分を削除し、日本語だけにすることで読みやすくしました。

4つのルール

- 英語を機械的に消すだけでは危険
- 数式内の英語、URL、略称は残す
- Markdown / LaTeX / 表 / 画像を見て判断
- “読むためのノート” に再構成する

Goodnotesで
読みやすいPDFも作る

iPadで読みやすく、書き込みやすい形にする

- ダークモードでPDF化
- 数式はKaTeXで綺麗に表示
- 不要な表示を減らす
- ページまたぎを減らし、余白も確保

宿題では、答えではなく試行錯誤を支えてもらう

頼まないこと ✕

- 答えを書いて
- そのまま埋めて

よく頼むこと ✓

- この問題は何を理解させたい？
- 今の考え方は合ってる？
- 答えは言わず、ヒントだけ教えて
- 次に見るべきポイントを教えて

“先生や詳しい先輩に相談する感覚”で使う

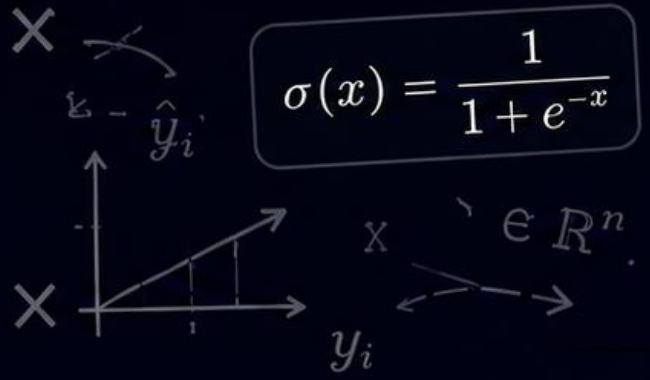
数式とコードのつながりが見えない

数式

$$y = \sum_{i=1}^{n} w_i x_i + b$$

$$\frac{dL}{dw} = \frac{1}{m} \sum_{i=1}^{m} x(y_i - \hat{y}_i)$$

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



なんでこの数式が
このコードになるの？

コード

```python
def predict(x, w, b):
    y = 0.0
    for i in range(n):
        y += w[i] * x[i]
    y += b
    return y
```

```python
def sigmoid(x):
    return 1 / (1 + math.exp(-x))
```

ImageGenで、理解が進む図を作る

抽象的な概念ほど、最初の1枚が理解の入口になる

例：プロンプト（お願いの例）	出力イメージ（例）
この概念を 初心者向けの図にして	
数式の流れを ストーリーっぽく見せて	
理解しづらい部分を キャラクター化して	

百聞は一見にしかず、を学習に持ち込める

復習用の問題も作ってもらえる

答えをもらうのではなく、理解確認のために使う

元ノート・講義範囲

講義ノート（例）

- 機械学習の概要
- 教師あり学習とは
- 回帰と分類の違い
- 活性化関数の役割
- 勾配降下法の流れ

...

復習用アウトプット

穴埋め問題

次の文章の（　　）を
埋めてください。

・教師あり学習は、
（　）を使ってモデルを学習する。
・回帰は（　　）、
・分類は（　　）を予測する。
...

小テスト

次のうち正しいものを
1つ選びなさい。

Q. 勾配降下法の目的は？
A. 最適な重みを見つける
B. データを増やす
C. モデルを可視化する
D. 学習時間を短くする
...

間違えやすい所チェック

次の内容は正しい？
誤っている場合は、
正しい表現に直しなさい。

「活性化関数は、学習の
結果そのものを決める。」

...

この範囲から穴埋め問題を作って

答えは最後にまとめて

まずは自分で解ける形にして

復習の回転数を上げやすい

学習のボトルネックが変わる

面倒な前処理より、**理解そのものに集中**しやすくなる

昔の詰まりどころ

- 環境構築
- エラー対応
- ノート整理
- 実験結果の整理

ボトルネックの
シフト

今の問い

- 何を理解したいか
- どこがわかっていないか
- どの実験を信じるか
- 自分の言葉で
  説明できるか

☆ 楽になるというより、**勉強の密度が上がる**

答えをもらうより、理解を確かめる

使い方次第で、学びの質が変わる

やりがち

- ✕ 答えだけ先に見る
  考える機会を失う
- ✕ 埋めてもらって終わる
  思考のプロセスが残らない
- ✕ そのままコピペで済ませる
  理解しているか分からない

おすすめ

- ✓ まず自分で考える
  思考の筋力を育てる
- ✓ 答えは最後に確認
  理解しているかをチェック
- ✓ ヒントだけもらう
  考える手がかりにする
- ✓ 間違えやすい所を確かめる
  弱点の把握につながる

自分で解ける形を保つと、理解確認に使いやすい

今日のまとめ

Codexは、学習サポートAIとしてもかなり強い

- 学習環境を整える
- 理解しやすい形に変換する
- 復習しやすい形に残す
- 試行錯誤を支える
- 図で理解の入口を作る

✦ 学びを前に進めるための**補助線**を引いてくれる ✦

一番伝えたいこと

学ぶ

整える

試す

Codexは、強いエンジニアだけの道具じゃなく、学びたい人の道具にもなる

見える化

復習

勉強にも使ってみると、見え方がかなり変わる

ご清聴ありがとうございました

コーディングだけじゃなく、
学習も助けてくれるCodex！

Codex Meetup Osaka #1

@SORA_LNV