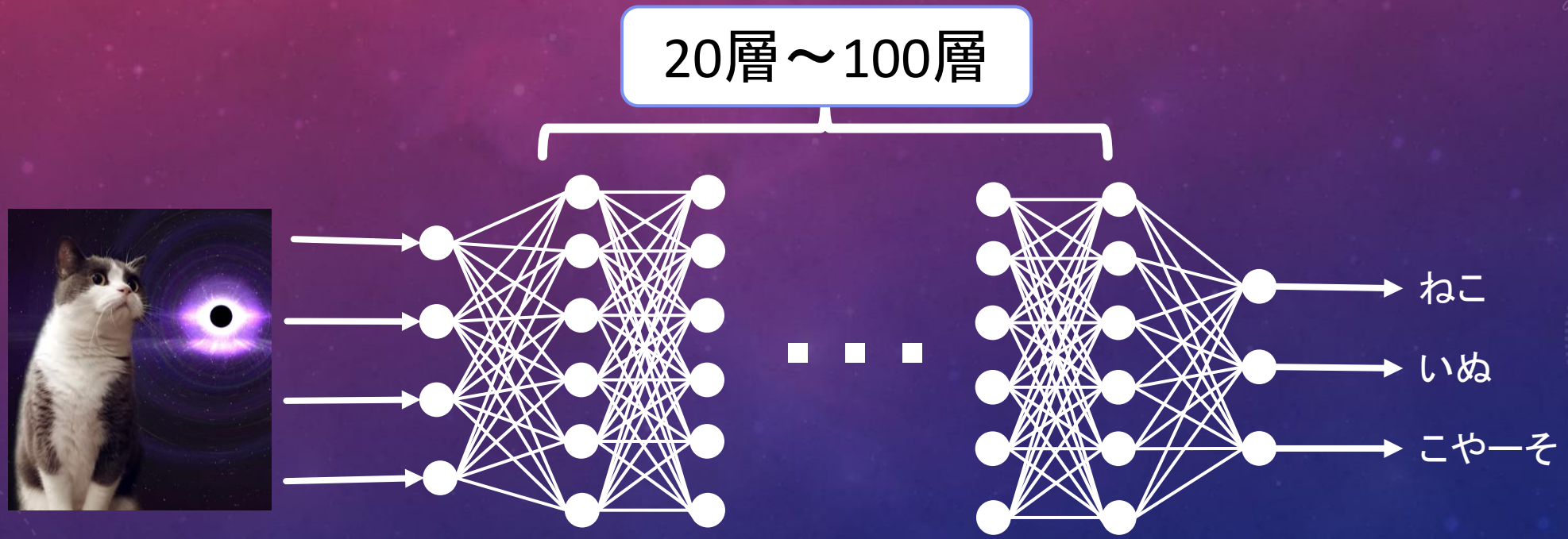
第7回
y=ax+bから始める
初心者向けML講座

これまでのあらすじ

単純なCNNやDNNの限界 － 深いNN

- 逆に層が深いCNNやDNNだと複雑なタスクに対応できるようになり、精度は高くなる可能性も高まるが、膨大な学習データや学習時間が必要となる。

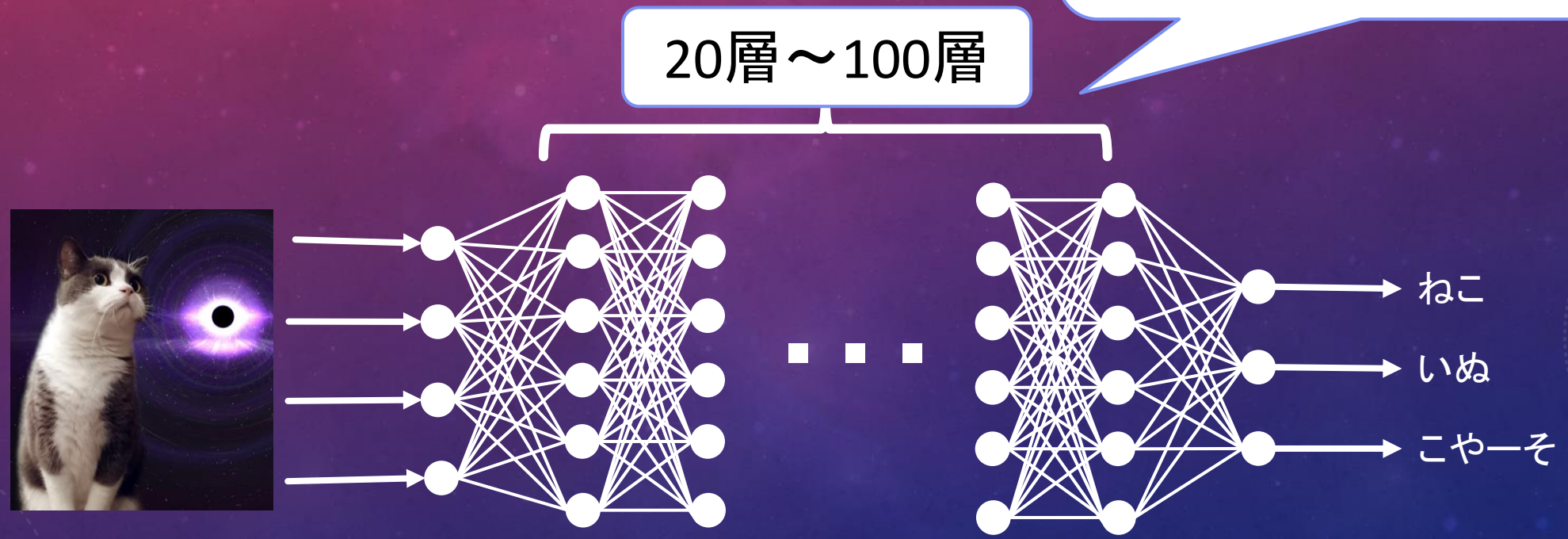
  →そもそも誤差が小さくならず学習が進まなくなってしまうケースも。



単純なCNNやDNNの限界 － 深いNN

- 逆に層が深いCNNやDNNだと複雑なタスクに対応できるようになり、精度は高くなる可能性も高まるが、膨大な学習データ
  →そもそも誤差が小さくならず学習が進まな

20層以上となると
おそらく高確率で
学習は進まなくなる

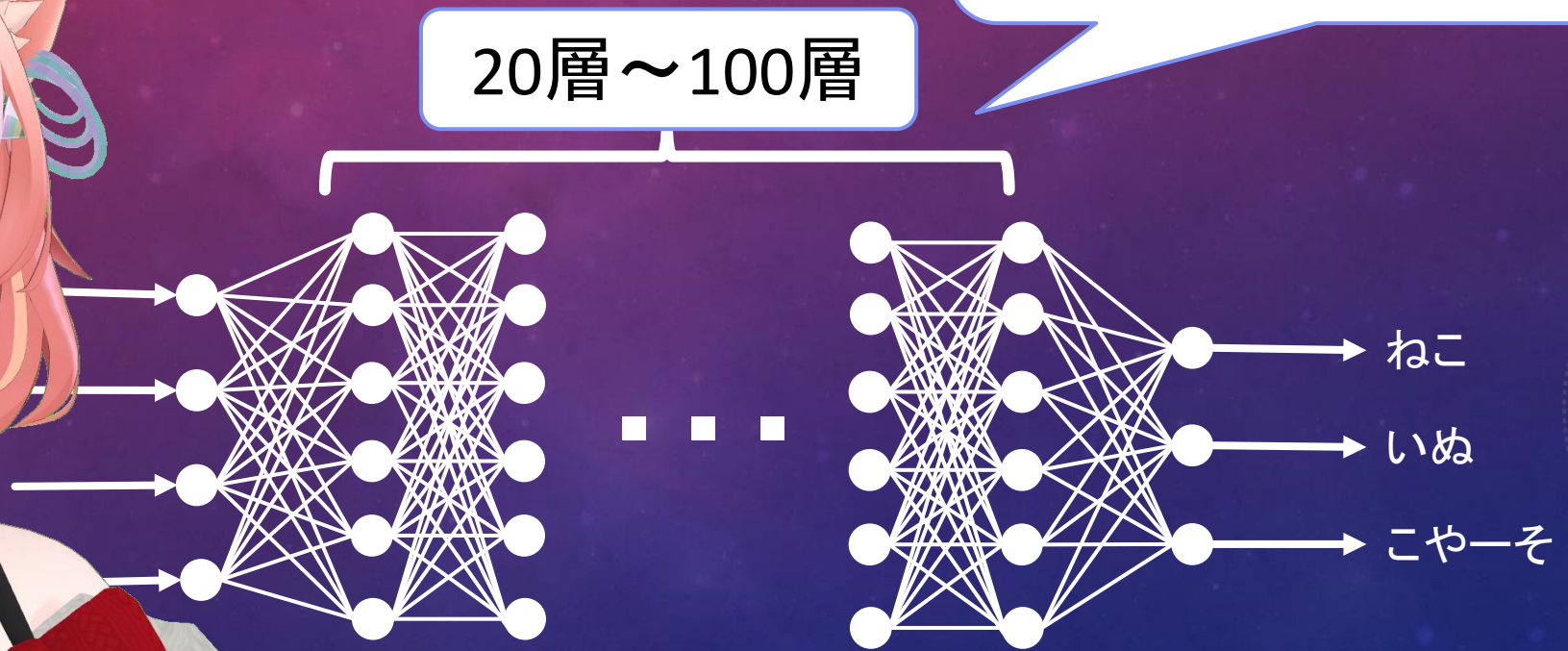


単純なCNNやDNNの限界 － 深いNN

- 逆に[illegible]だと複雑なタスクに対応できるようになり、精度は高くなる[illegible]膨大な学習データ[illegible]
  →そもそも誤差が小さくならず学習が進まな[illegible]

マジで！？

20層以上となると
おそらく高確率で
学習は進まなくなる

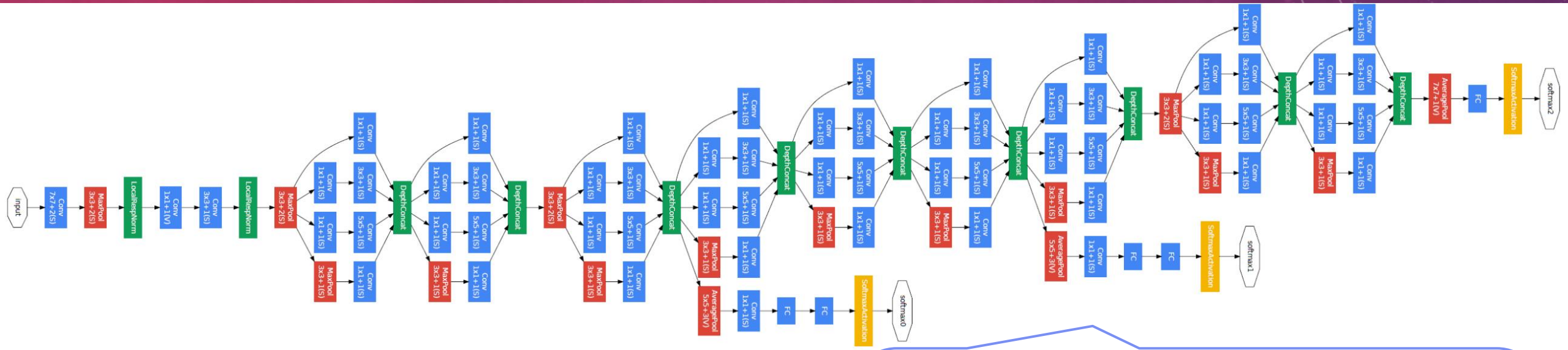


CNNやDNNを深いNNにするには？

- モデルそのものを大規模深層モデルにする

限界を超えられた最初のモデル(22層)

- GoogLeNet(2014年画像コンペ優勝モデル)



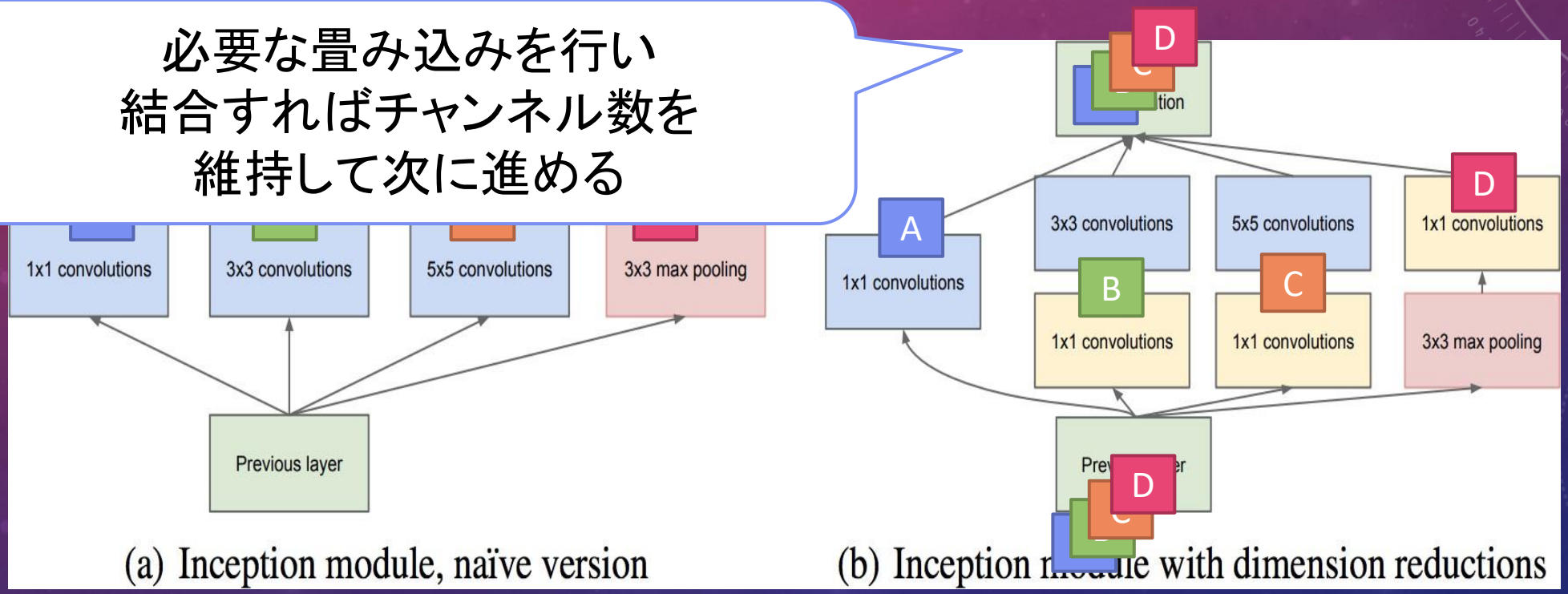
青：Convolution(畳み込み)
赤：Pooling
黄色：Softmax関数
緑:その他

出典:[Szegedy,C.etal.,2014]

Inceptionモジュール

- 1x1と3x3と5x5のフィルターを使った畳み込み、3x3のフィルターを使ったマックスプーリングを別々に行いチャンネル数を積み上げるように結合する

必要な畳み込みを行い
結合すればチャンネル数を
維持して次に進める



(a) Inception module, naïve version

(b) Inception module with dimension reductions

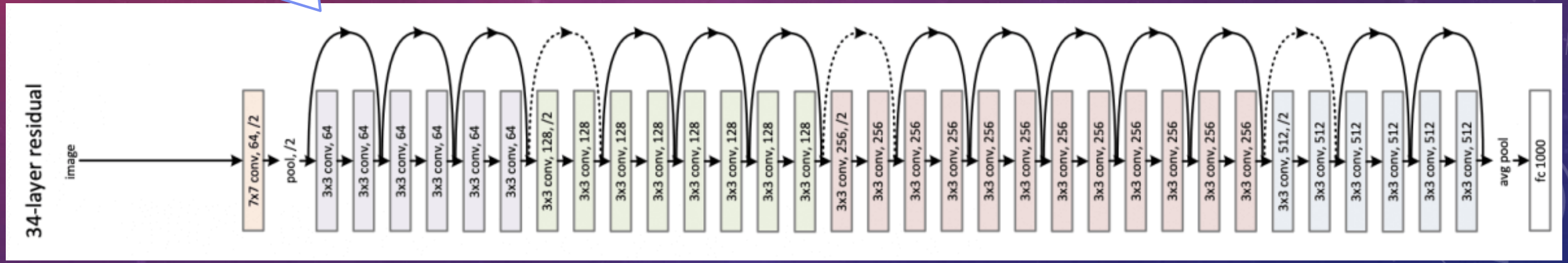
[Szegedy, C. et al., 2014]

理論上無限？に繋げられるResNet(2015年登場)

- 論文上では最大152層繋げて学習している。

Transformer等に
つながる革新的モデル

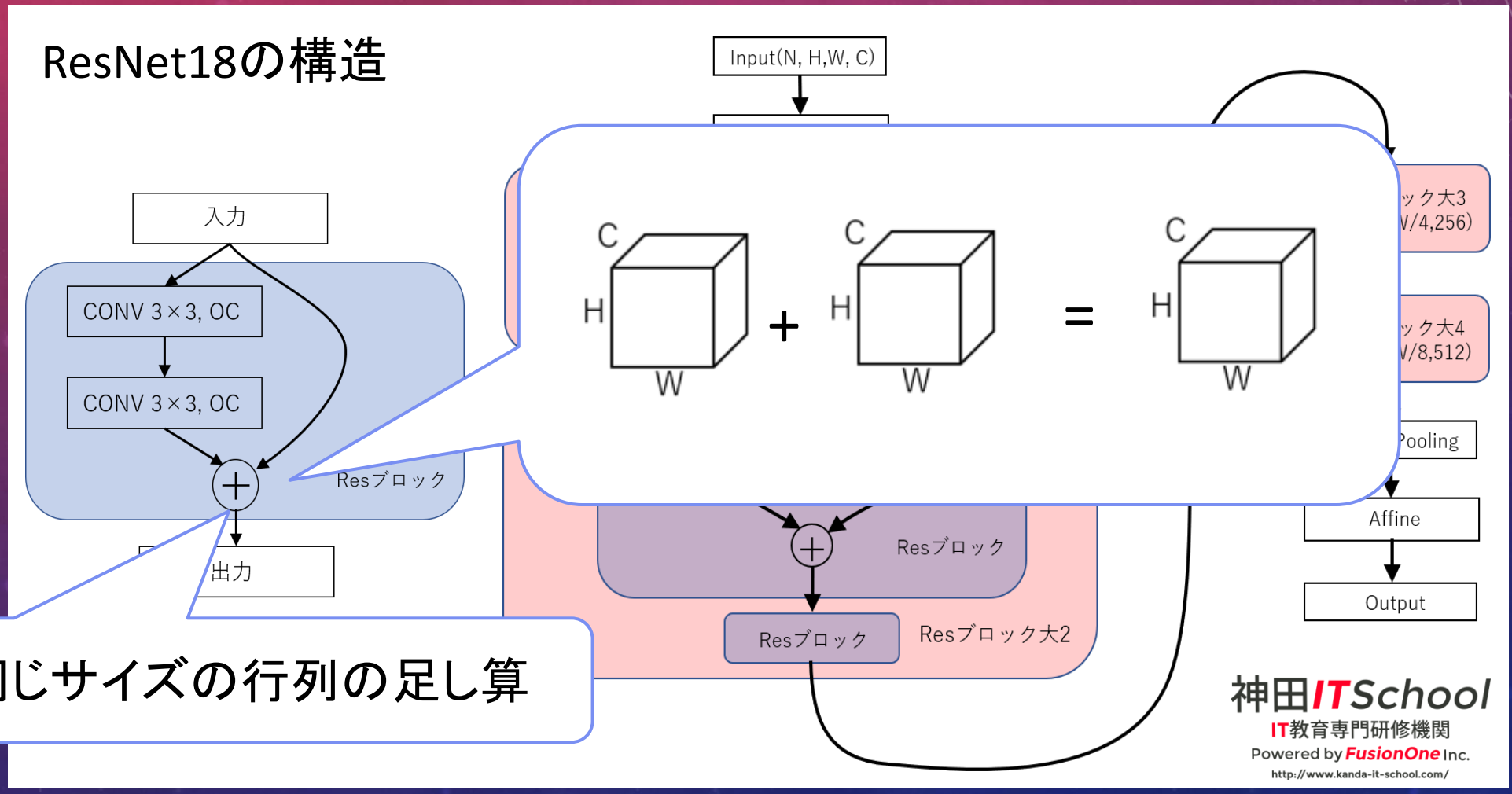
ChatGPT
Stable Diffusion



出典：**Deep Residual Learning for Image Recognition**
https://arxiv.org/abs/1512.03385

スキップ接続が最大の特徴

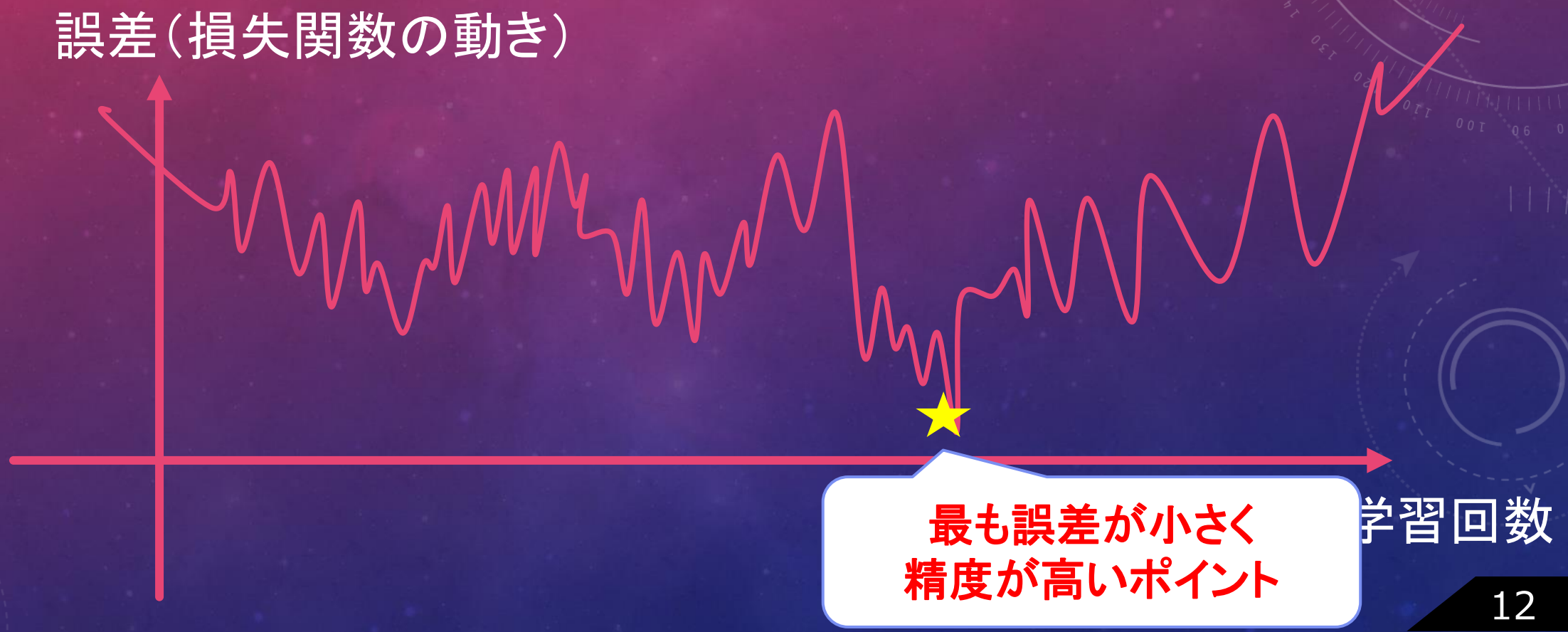
- 畳み込みをしたものしないものを足し合わせている（スキップ接続）



ニューラルネットワーク(NN)深層化対策その２

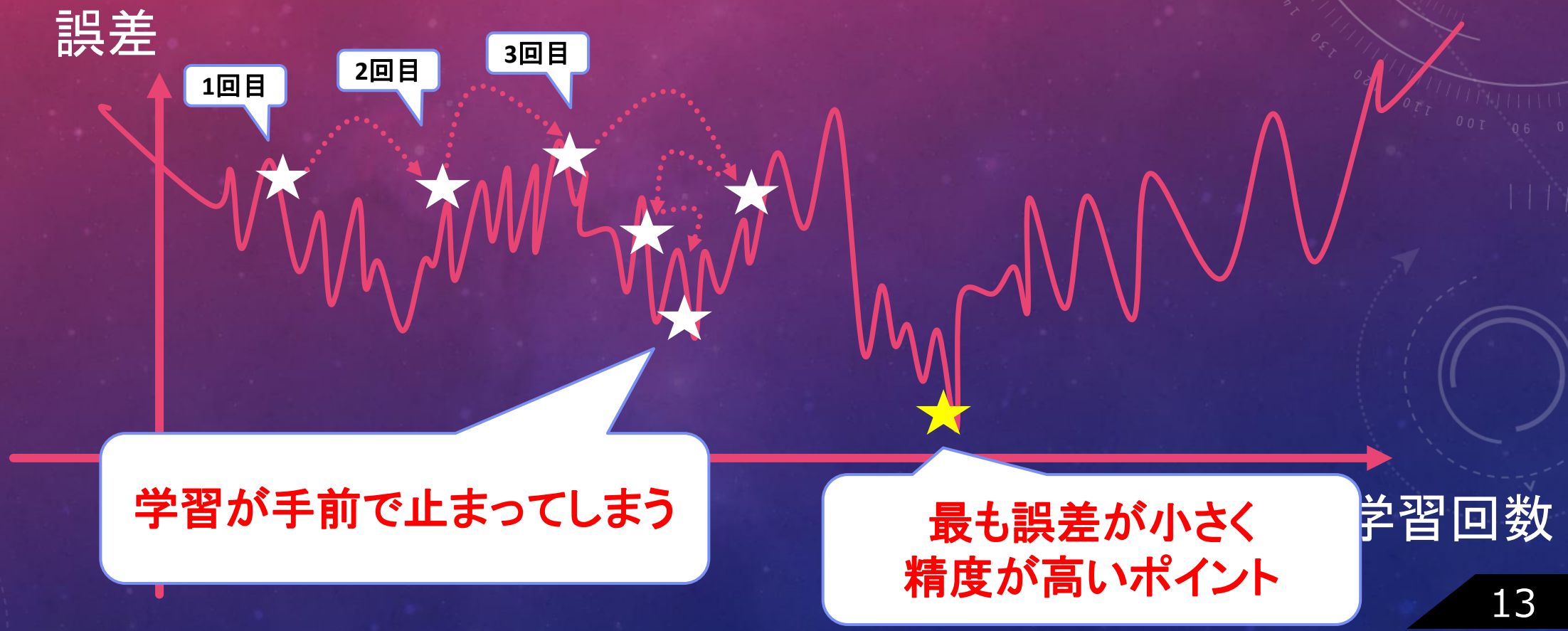
何故、層が深くなると学習が進まないのか？

- NNはあくまでも正解と予測値の誤差が一番小さい場所を探すアルゴリズム
- 誤差の曲線が複雑になり局所解に陥りやすくなる



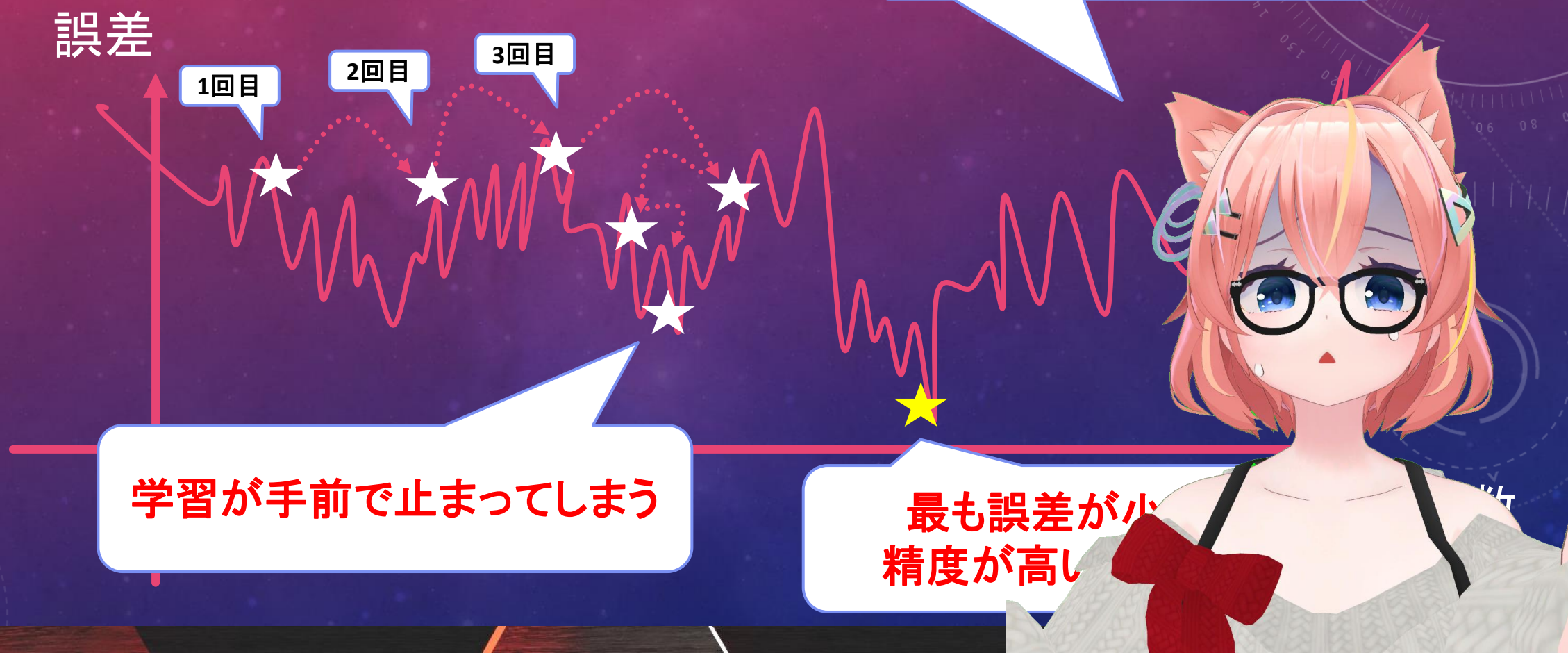
何故、層が深くなると学習が進まないのか？

- NNはあくまでも正解と予測値の誤差が一番小さい場所を探すアルゴリズム
- 誤差の曲線が複雑になり局所解に陥りやすくなる



何故、層が深くなると学習が進まないのか？

- NNはあくまでも正解と予測値の誤差が一番小
- 誤差の曲線が複雑になり局所解に陥りやすくな



CNNやDNNを深いNNにするには？

CNNやDNNを深いNNにするには？

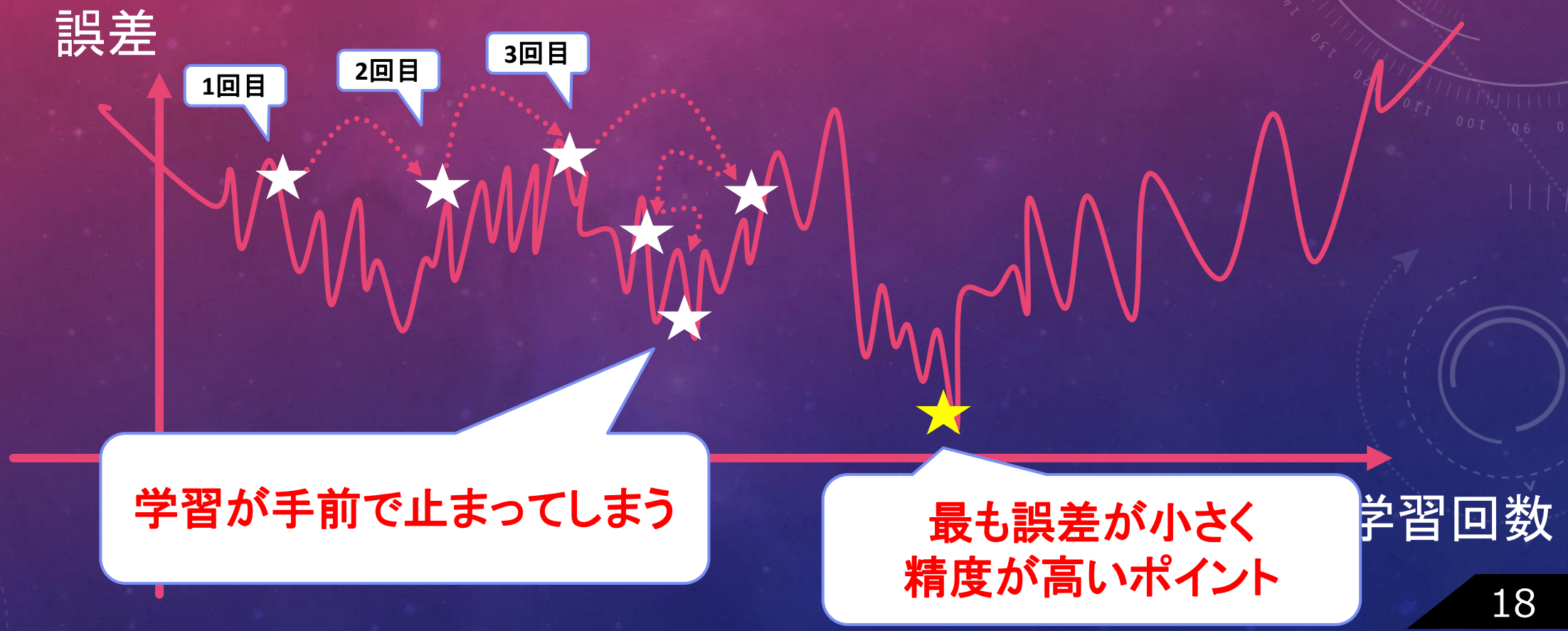
- モデルそのものを大規模深層モデルにする

CNNやDNNを深いNNにするには？

- モデルそのものを大規模深層モデルにする
- 確率的勾配法(SGD)を効率の良い計算式に変える **New!!**

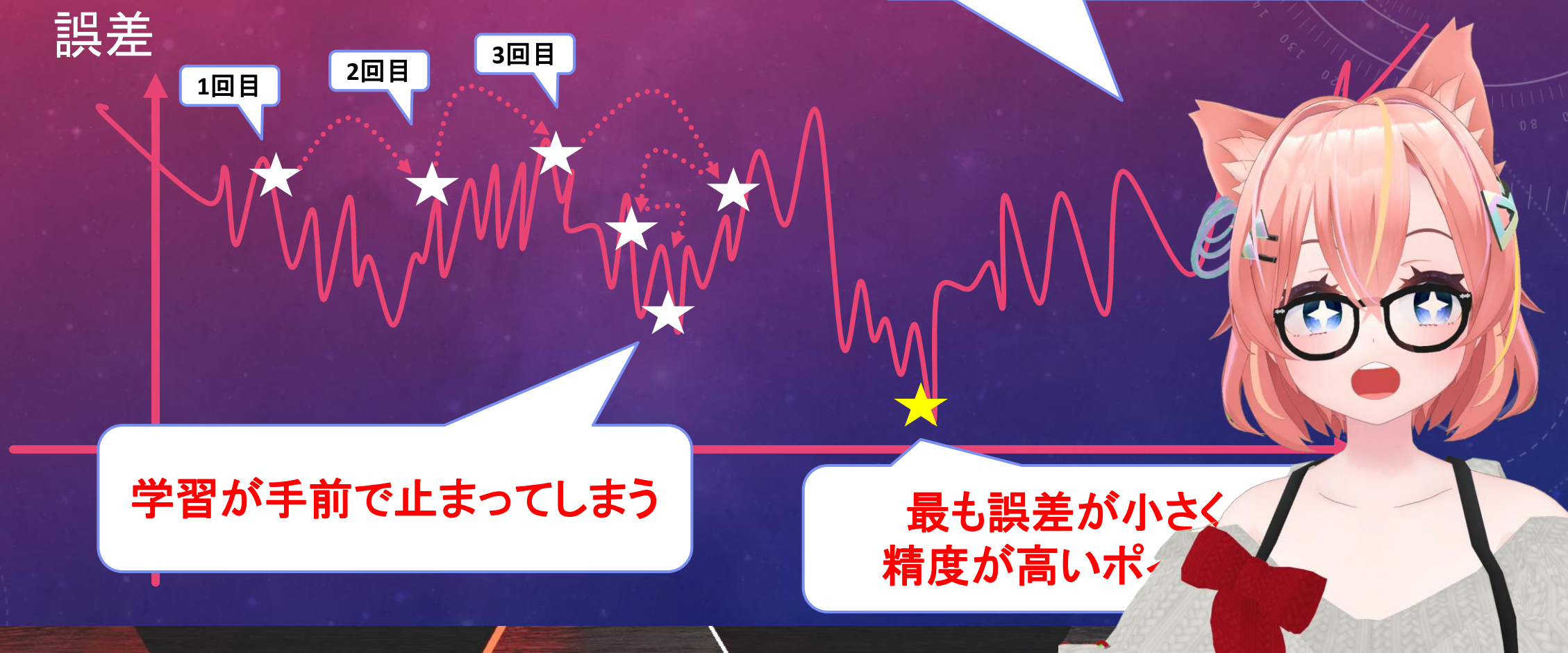
何故、層が深くなると学習が進まないのか？

- NNはあくまでも正解と予測値の誤差が一番小さい場所を探すアルゴリズム
- 誤差の曲線が複雑になり局所解に陥りやすくなる



何故、層が深くなると学習が進まないのか？

- NNはあくまでも正解と予測値の誤差が一番小
- 誤差の曲線が複雑になり局所解に陥りやすくな



機械学習の学習手順

- ①入力データから機械学習アルゴリズムを使用して予測値の計算
- ②予測値と正解の間の誤差を求めその誤差から学習パラメータに関する各微分値を計算
- ③微分値から勾配法を用いてパラメータの更新

- これを繰り返すのが機械学習の学習手順

機械学習の学習手順

- ①入力データから機械学習アルゴリズムを使用して予測値の計算
- ②予測値と正解の間の誤差を求めその誤差から学習パラメータに関する各微分値を計算
- ③微分値から勾配法を用いてパラメータの更新

- これを繰り返すのが機械学習の学習手順

機械学習の学習手順

- ①入力データから機械学習アルゴリズムを使用して予測値の計算
- ②予測値と正解の間の誤差を求めその誤差から学習パラメータに関する各微分値を計算
- ③微分値から勾配法を用いてパラメータの更新

- ③の計算式はy=ax+bのaを学習する際は

a':次のaのパラメーター、λ：学習係数、$\frac{\partial E}{\partial a}$：誤差を$a$について偏微分した値(誤差の傾き)

$$a' = a - \lambda \frac{\partial E}{\partial a}$$

単純なモデルでの機械学習

y=ax+b

- ①適当(ランダム)で決めたaとbの値で計算した予測値y'を求める。
- ②予測値y'と正解数値yから誤差を求める2乗和誤差なら$(y'-y)^2$、その微分値は$\frac{1}{2}(y'-y)$で計算する。
- ③以下の計算式に微分値、学習係数λ、今回のaの値から次のa'の値を求める。

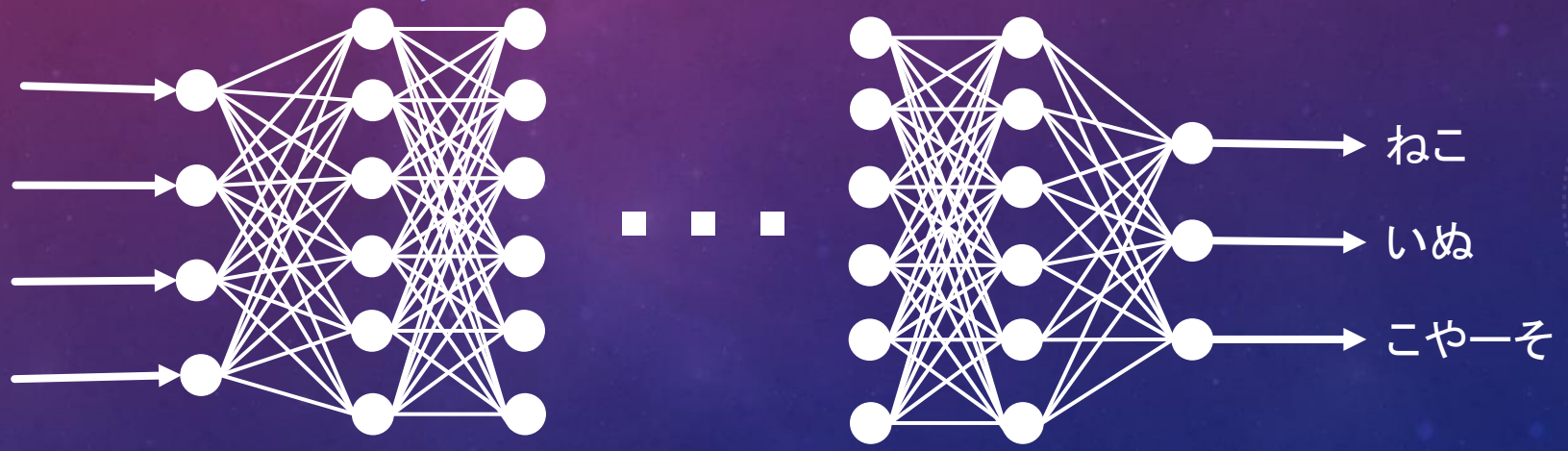
$$a' = a - \lambda \frac{\partial E}{\partial a}$$

単純なCNNやDNNの限界 － 深いNN

- 複雑なモデルであっても、各線1本1本にある重みの更新値を求めて学習している。

線1本1本の所にある
重みwは全て偏微分値を
求めて同じ計算をしている。

$$w' = w - \lambda \frac{\partial E}{\partial w}$$



機械学習の学習手順

- ①入力データから機械学習アルゴリズムを使用して予測値の計算
- ②予測値と正解の間の誤差を求めその誤差から学習パラメータに関する各微分値を計算
- ③微分値から勾配法を用いてパラメータの更新

- ③の計算式はy=ax+bのaを学習する際は

a':次のaのパラメーター、λ：学習係数、$\frac{\partial E}{\partial a}$：誤差を$a$について偏微分した値(誤差の傾き)

$$a' = a - \lambda \frac{\partial E}{\partial a}$$

この計算式を変えたら
山を越えられるのでは？

モメンタム法を使った計算式に変える。

- モメンタム法の計算式

w′:次のwのパラメーター、**λ：学習係数**、$\frac{\partial E}{\partial w}$：誤差をwについて偏微分した値(誤差の傾き)

α：どの程度運動量を影響させるのかのハイパーパラメータ(0< α<1)

v：前回のwの更新値　　v'：今回のwの更新値

$$w' = w - \lambda \frac{\partial E}{\partial w}$$

$$v' = \alpha v - \lambda \frac{\partial E}{\partial w}$$

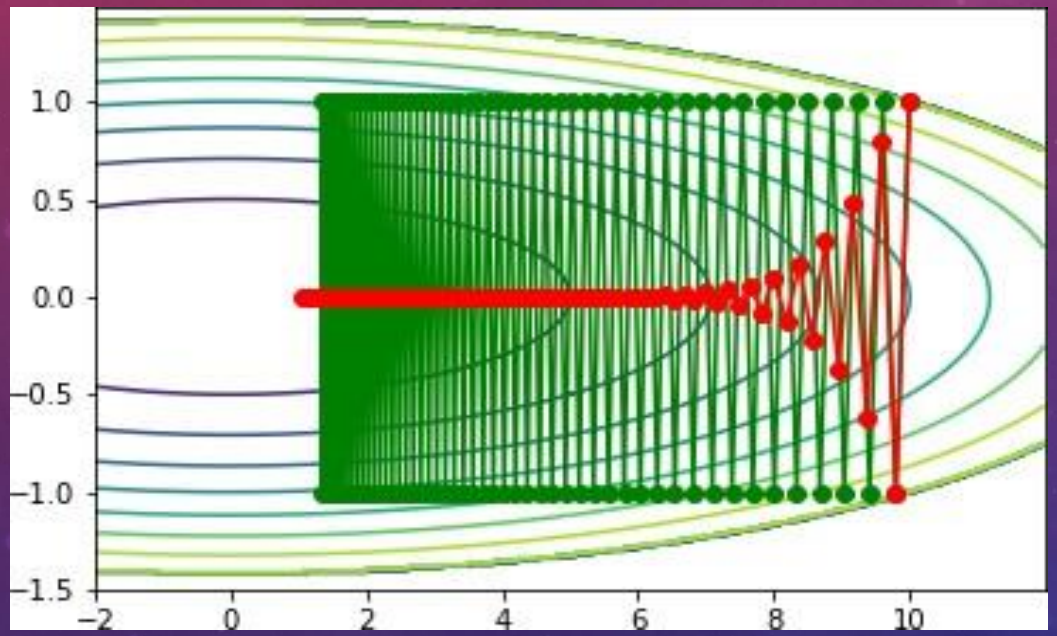
$$w' = w + v'$$

モメンタム法の効果を実験

- $z=x^2+100y^2$

という計算式で最も小さいｚを求めるために勾配法を用いるとすると

x=10、y=1から通常の勾配降下法とモメンタムを適応した場合の差



GDとモメンタムの時間発展の比較

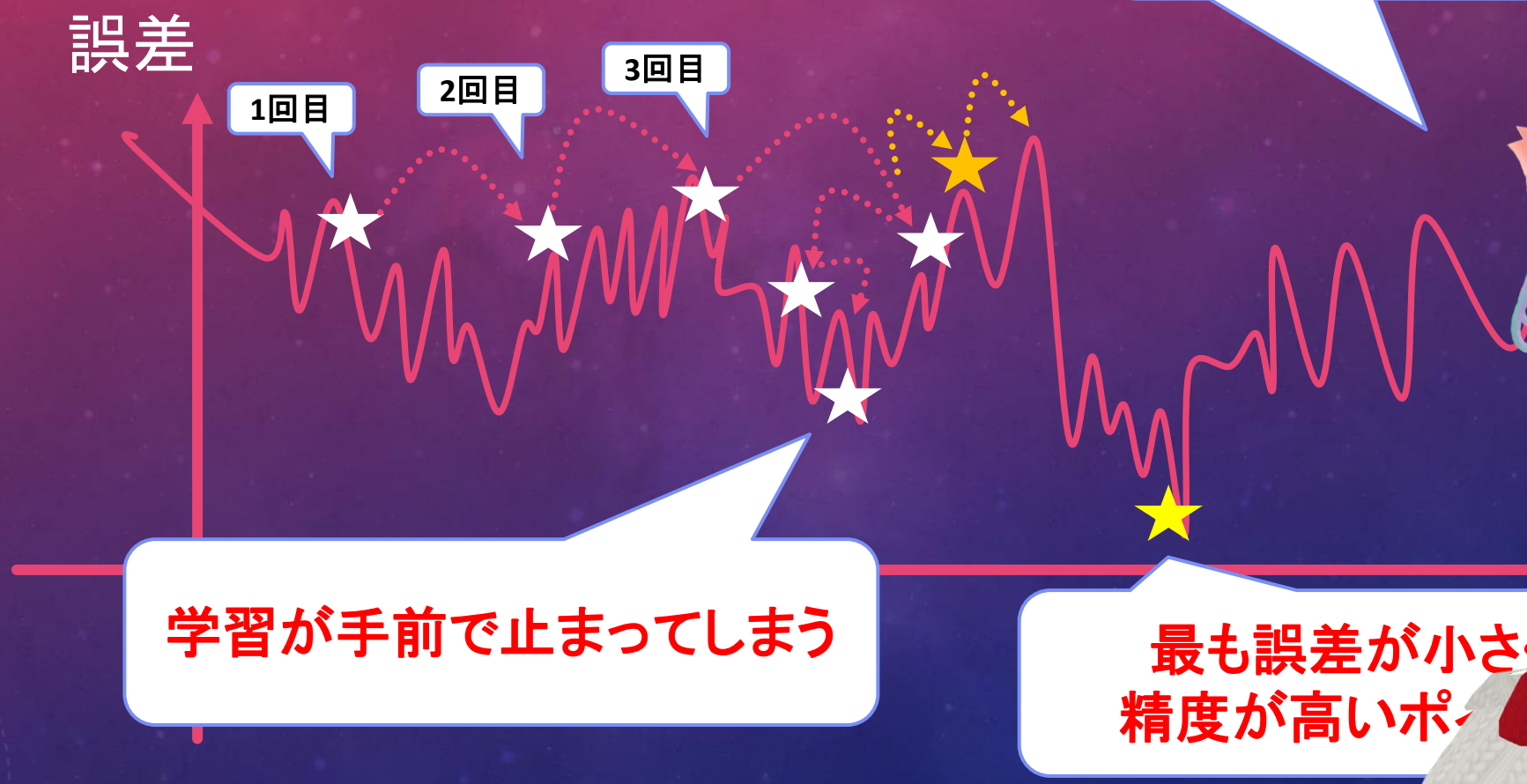
緑：GD，赤：モメンタム

通常の勾配法 $w' = w - \lambda \frac{\partial E}{\partial w}$

モメンタム法 $v' = \alpha v - \lambda \frac{\partial E}{\partial w}$

$w' = w + v'$

モーメンタムの場合



モーメンタムを使うことで
大きな誤差の山を
越えやすくなる！

その他のアルゴリズム：RMSrop

- r′、r：微分値(誤差の傾き)の絶対値
- **ρ(ロー)：(0<ρ<1)前回のrと今回の微分値の2乗のどちらを重視するか決めるハイパーパラメーター**
- **η(イータ)：(0<η<1) r′をどの程度影響させるか決めるハイパーパラメーター**
- ε：分母が0にならないよう発散防止のための微小量10^{-7}などの数値にする

ρが小さいほど直近の微分値(誤差の傾き)に影響されるよ

ρが大きいと今までのr(の積み重ね)を重視するよ

$$r' = \rho r + (1-\rho)\frac{\partial E}{\partial w} * \frac{\partial E}{\partial w}$$

$$w' = w - \frac{\eta}{\sqrt{r' + \varepsilon}}\frac{\partial E}{\partial w}$$

r′が分母にあるので微分値が大きければ小さく小さければ大きくなるよ

その他のアルゴリズム：RMSrop

- r′、r：微分値(誤差の傾き)の絶対値
- **ρ(ロー)：(0<ρ<1)前回のrと今回の微分値の2乗のどちらを重視するか決めるハイパーパラメーター**
- **η(イータ)：(0<η<1) r′をどの程度影響させるか決めるハイパーパラメーター**
- ε：分母が0にならないよう発散防止のための微小量10^{-7}などの数値にする

ρが小さいほど直近の微分値
(誤差の傾き)に影響されるよ

ρが大きいと今までの
r(の積み重ね)を重視するよ

$$r' = \rho r + (1-\rho)\frac{\partial E}{\partial w} * \frac{\partial E}{\partial w}$$

$$w' = w - \frac{\eta}{\sqrt{r' + \varepsilon}}\frac{\partial E}{\partial w}$$

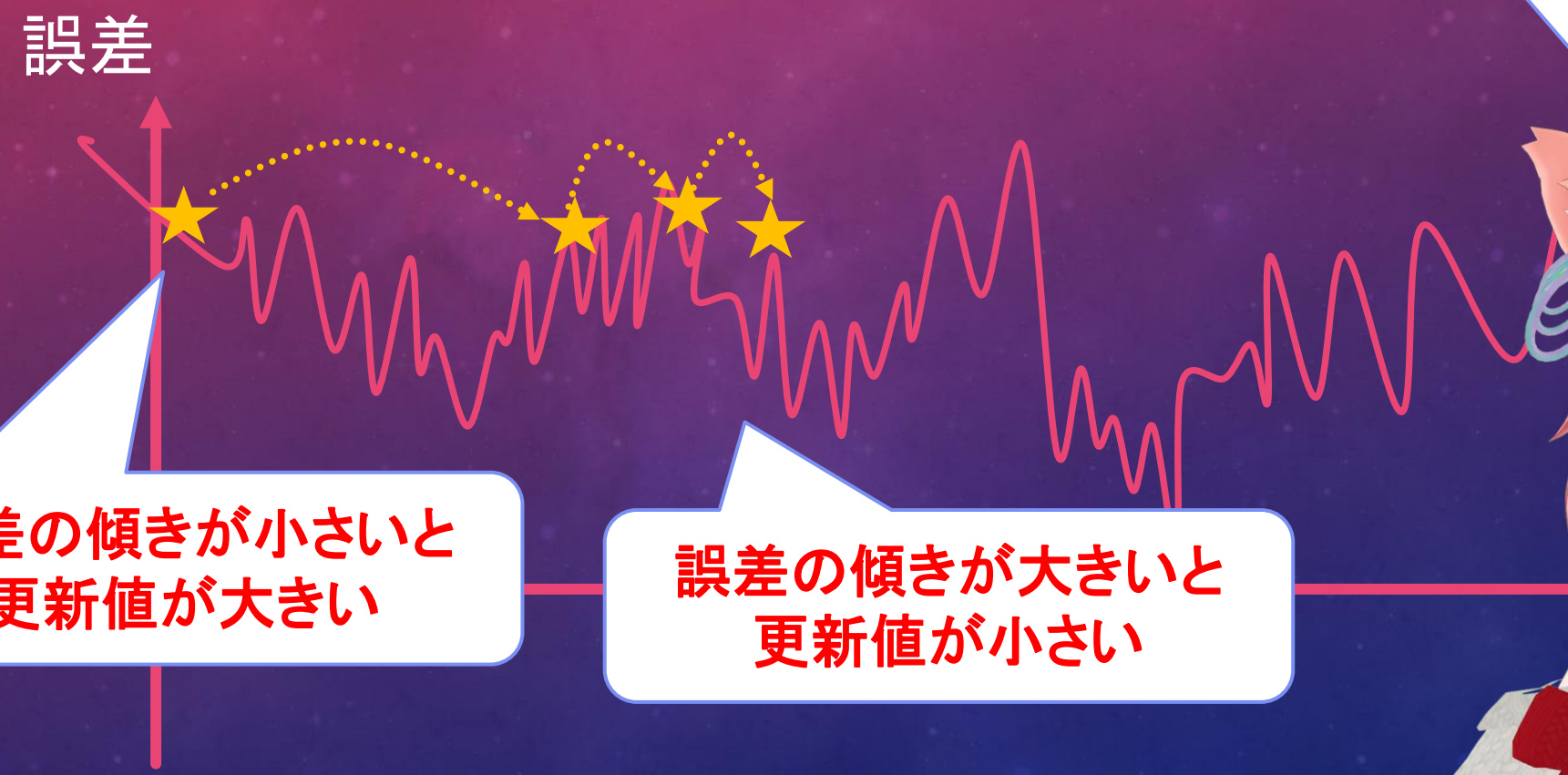
r′が分母にあるので
微分値が大きければ小さく
小さければ大きくなるよ

難しいけどどういった
ハイパーパラメータが
あるかは知っておこう

RMSropの場合

※グラフはイメージです。

誤差の傾きが大きいと小さく
傾きが小さいと大きく更新されていくよ



その他のアルゴリズム：Adam

- モーメンタムとRMSropの良いとこどりをしてるアルゴリズム
- s、s′：モーメンタム法　　**β_1：$(0 < \beta_1 < 1)$モーメンタム法の影響度、大きいほど影響力がある**
- r、r′:RMSrop　　**β_2：$(0 < \beta_2 < 1)$RMSropの影響度、大きいほど影響力がある**
- $\tilde{s}$、$\tilde{r}$：意味づけはしづらいが、更新回数が増えるほど小さくなりやすい
- **α：全体の学習係数**

$$s' = \beta_1 s + (1 - \beta_1)\frac{\partial E}{\partial w}$$

$$r' = \beta_2 r + (1 - \beta_2)\frac{\partial E}{\partial w} * \frac{\partial E}{\partial w}$$

$$\tilde{s} = \frac{s'}{1 - \beta_1^t} \quad (t\text{はそれまでの更新回数})$$

$$\tilde{r} = \frac{r'}{1 - \beta_2^t} \quad (t\text{はそれまでの更新回数})$$

$$w = w - \alpha\frac{\tilde{s}}{\sqrt{\tilde{r} + \varepsilon}}$$

回を重ねるごとに分母は
1以下の数値から1に近づくよ
つまり$\tilde{s}$、$\tilde{r}$は小さくなりやすいよ

その他のアルゴリズム：Adam

- モーメンタムとRMSropの良いとこどりをしてるアルゴリズム
- s、s′：モーメンタム法　　β_1：$(0 < \beta_1 < 1)$モーメンタム法の影響度、大きいほど影響力がある
- r、r′:RMSrop　　β_2：$(0 < \beta_2 < 1)$RMSropの影響度、大きいほど影響力がある
- $\tilde{s}$、$\tilde{r}$：意味づけはしづらいが、更新回数が増えるほど小さくなりやすい
- α：全体の学習係数

$$s' = \beta_1 s + (1 - \beta_1)\frac{\partial E}{\partial w}$$

$$r' = \beta_2 r + (1 - \beta_2)\frac{\partial E}{\partial w} * \frac{\partial E}{\partial w}$$

$$\tilde{s} = \frac{s'}{1 - \beta_1^t} \quad (t\text{はそれまでの更新回数})$$

$$\tilde{r} = \frac{r'}{1 - \beta_2^t} \quad (t\text{はそれまでの更新回数})$$

$$w = w - \alpha \frac{\tilde{s}}{\sqrt{\tilde{r} + \varepsilon}}$$

回を重ねるごとに分母は
1以下の数値から1に近づくよ
つまり$\tilde{s}$、$\tilde{r}$は小さくなりやすいよ

※実装しやすく変えた式

$$\mathbf{s} \leftarrow \beta_1 \mathbf{s} + (1 - \beta_1)\frac{\partial E}{\partial \theta}$$

$$\mathbf{r} \leftarrow \beta_2 \mathbf{r} + (1 - \beta_2)\frac{\partial E}{\partial \theta} * \frac{\partial E}{\partial \theta}$$

$$\lambda \leftarrow \lambda \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t} \quad (t \text{ はそれまでの更新回数})$$

$$\theta \leftarrow \theta - \lambda \frac{\mathbf{s}}{\sqrt{\mathbf{r} + \epsilon}}$$

その他のアルゴリズム：Adam

- モーメンタムとRMSropの良いとこどりをしてるアルゴリズム
- s、s′：モーメンタム法　　β_1：$(0 < \beta_1 < 1)$モーメンタム法の影響度、大きいほど影響力がある
- r、r′:RMSrop　　β_2：$(0 < \beta_2 < 1)$RMSropの影響度、大きいほど影響力がある
- $\tilde{s}$、$\tilde{r}$：意味づけはしづらいが、更新回数が増えるほど小さくなりやすい
- α：全体の学習係数

$$s' = \beta_1 s + (1 - \beta_1)\frac{\partial E}{\partial w}$$

$$r' = \beta_2 r + (1 - \beta_2)\frac{\partial E}{\partial w} * \frac{\partial E}{\partial w}$$

$$\tilde{s} = \frac{s'}{1 - \beta_1^t} \quad (t\text{はそれまでの更新回数})$$

$$\tilde{r} = \frac{r'}{1 - \beta_2^t} \quad (t\text{はそれまでの更新回数})$$

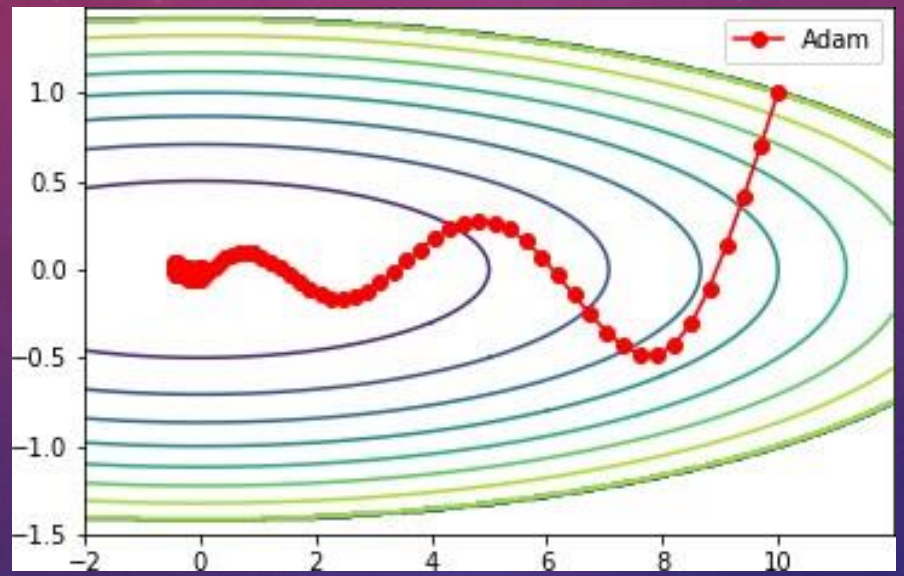
$$w = w - \alpha \frac{\tilde{s}}{\sqrt{\tilde{r} + \varepsilon}}$$

回を重ねるごとに分母は
1以下の数値から1に近づくよ
つまり$\tilde{s}$、$\tilde{r}$は小さくなりやすいよ

これを考えた人は
変態だと思うよ……

その他のアルゴリズム：Adam

- **運動量**と**誤差の微分値(傾き)**を考えつつ、徐々に小さく更新していく事が出来る



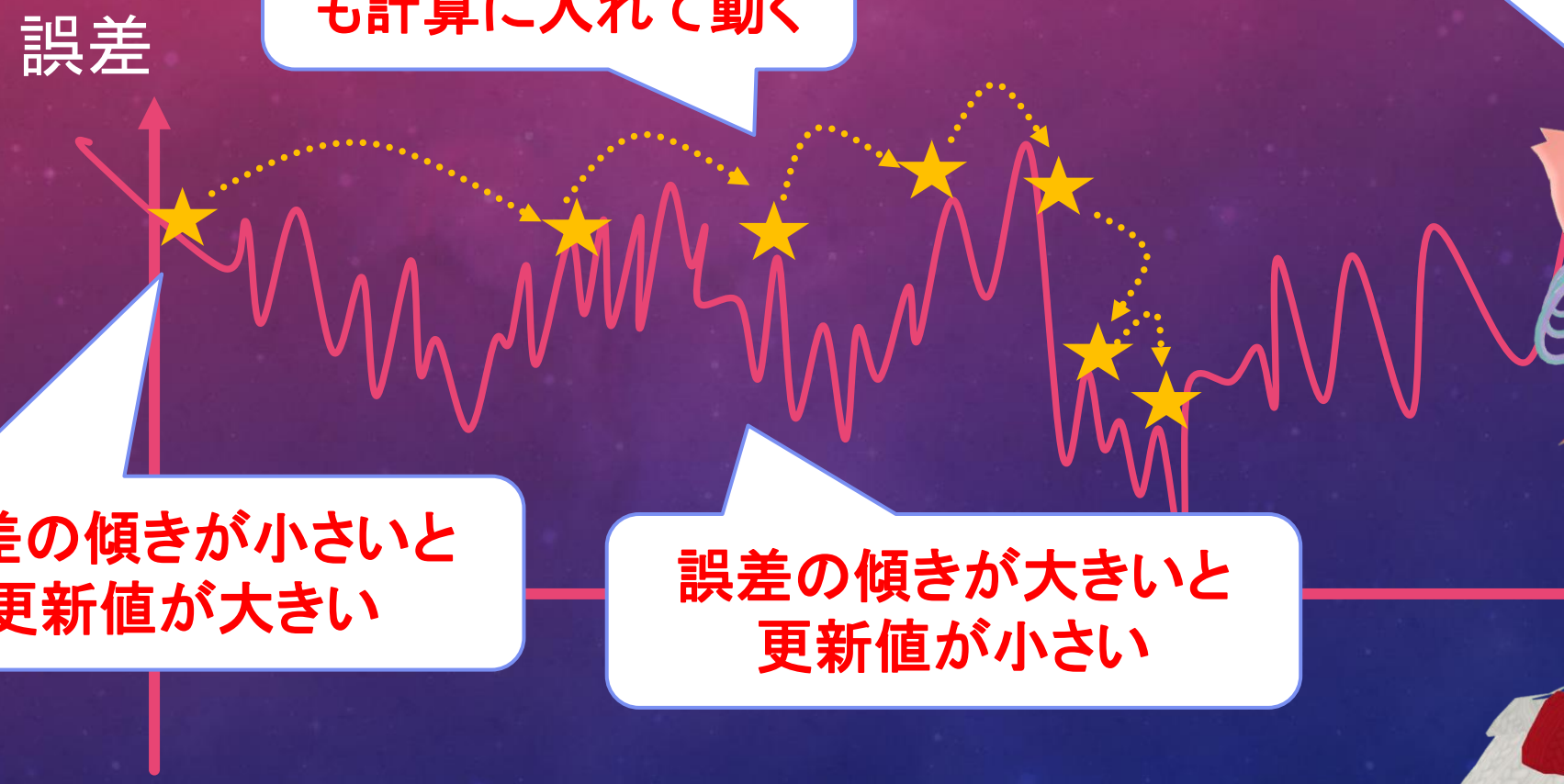
時間発展の様子

Adamの場合

※グラフはイメージです。

始めは大きく動きながら
徐々に小さく動くようになり、
かつ、モーメンタムとRMSropの
良いとこどりをするよ！

今までの運動量
も計算に入れて動く



誤差の傾きが小さいと
更新値が大きい

誤差の傾きが大きいと
更新値が小さい

Kerasの場合、結局どう使うの？

- optimizerが使うアルゴリズムの指定なのでそこを指定するだけでOK！

ココの指定を変えるだけでOKだよ！

```
'''optimizer定義（学習アルゴリズム）'''
model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])

'''学習'''
history=model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_data=(x_test, y_test))
```

Kerasの場合、結局どう使うの？

- optimizerが使うアルゴリズムの指定なのでそこを指定するだけでOK

ML Shukai

ココの指定を変えるだけでOKだよ！

```
'''optimizer定義（学習アルゴリズム）'''
model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])

'''学習'''
history=model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_data=(x_test, y_test))
```

なら計算式覚える意味なくない！？

Kerasの場合、結局どう使うの？

- optimizerが使うアルゴリズムの指定なのでそこを指定するだけでOK！

```
'''optimizer定義（学習アルゴリズム）'''
model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])

'''学習'''
history=model.fit(x_train, y_train, batch_size=batch_size, epoch                    lidation_data=(x_test, y_test))
```

引数でハイパーパラメータの指定が出来るのでハイパーパラメータの種類や意味が重要になる

Kerasの場合、結局どう使うの？

- optimizerが使うアルゴリズムの指定なのでそこを指定するだけでOK

```
1 '''optimizer定義（学習アルゴリズム）'''
2 model.compile(loss='categorical_crossentropy' optimizer=Adam(learning_rate=0.001) metrics=['accuracy'])
3
4 '''学習'''
5 history=model.fit(x_train, y_train, batch_size=batch_size, epoch                    lidation_data=(x_test, y_test))
```

重要なのは計算式ではなく
どのハイパーパラメータが
どういった影響があるかだよ！

引数でハイパーパラメータの指定が
出来るのでハイパーパラメータの
種類や意味が重要になる

おわり

ありがとうございました

次回はニューラルネットワーク深層化する時に使われる工夫その３

（予定）