不確実性と微分可能性を考慮した機械学習

徐　宏坤

2021.02.22

予測と意思決定

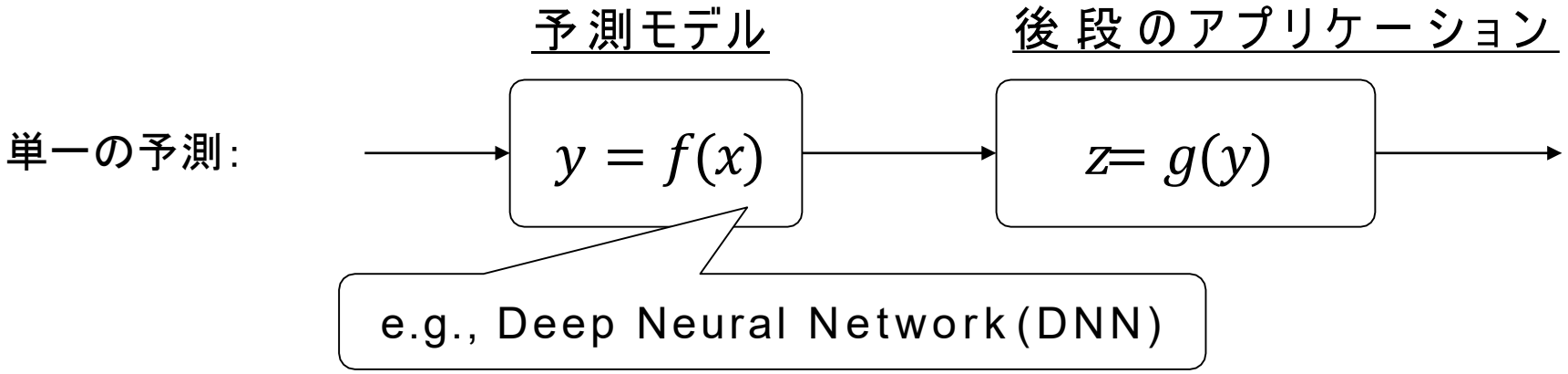
・多くの意思決定は予測に基づいて行われる

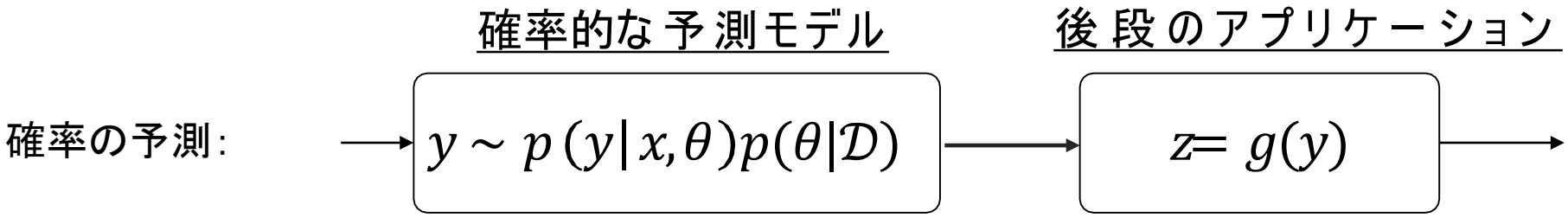
・予測は外れるときもある

・意思決定は予測の不確実さも考慮して実施されるべき

予測と意思決定の実装

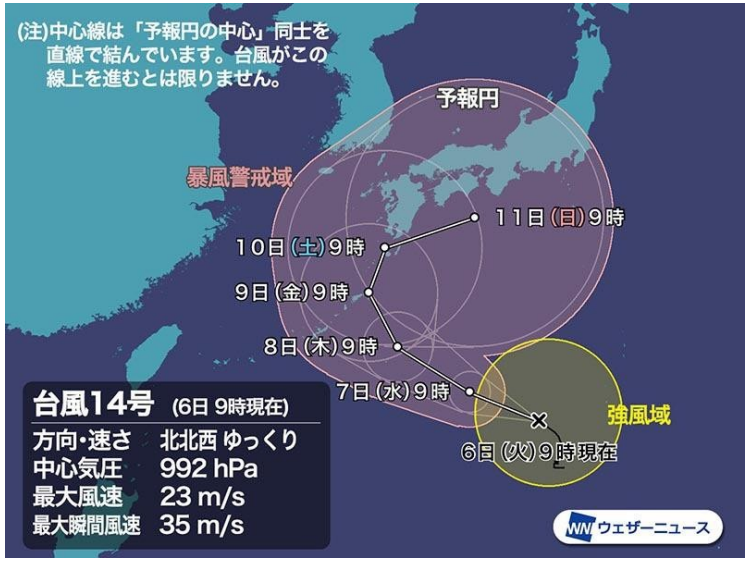
- 最先端の技術でも精度100%は保証されない





予測の不確実さ

偶発的不確実性(Aleatoric Uncertainty):
確率的振る舞いに起因



認識論的不確実性(Epistemic Uncertainty):
知識(データ・モデル)の違いに起因

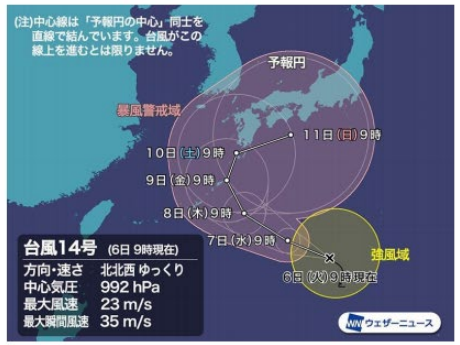
- 不確実性による予測の幅を知ることが柔軟・頑健な意思決定につながる

不確実性を考慮した機械学習

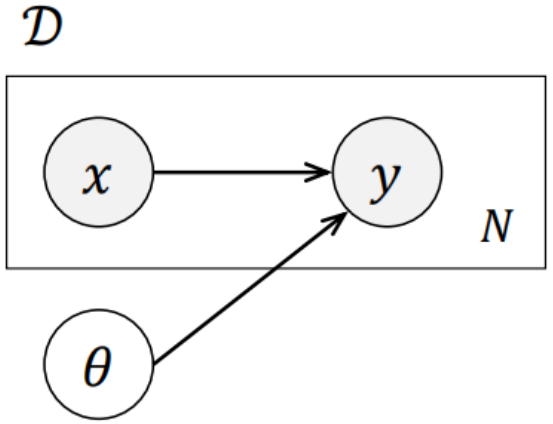
- 確率的な予測モデル

$$y \sim p(y|x,\theta)p(\theta|\mathcal{D})$$

Aleatoric Uncertainty



Epistemic Uncertainty



モデルパラメータ
(e.g., DNN の重み)

偶発的不確実性

- 一般的な回帰：
  - $p(y|x,\theta) \coloneqq \mathcal{N}(\mu_\theta(x), \Sigma)$, Σ:分散（定数）
  - 損失関数：$-\log p(y|x,\theta) \propto \left(y-\mu_\theta(x)\right)^2$

二乗誤差の最小化

- 偶発的不確実性を考慮した回帰：
  - $p(y|x,\theta) \coloneqq \mathcal{N}(\mu_\theta(x), \Sigma_\theta(x))$
  - 損失関数：

$$-\log p(y|x,\theta) = \frac{1}{2\Sigma_\theta^2(x)}\left(y-\mu_\theta(x)\right)^2 + \frac{1}{2}\log \Sigma_\theta^2(x)$$

認識論的不確実性

- $p(\theta|\mathcal{D}) = p(\mathcal{D}|\theta)p(\theta)$
  - ➢データ $\mathcal{D}$ で条件付けされたパラメタ θ の事後分布

- 一般的な学習：点推定　⇒　予測モデルは一つに限定
  - ➢最尤推定：$\theta^* = \mathrm{argmax}_\theta \log p(\mathcal{D}|\theta)$
  - ➢MAP (Maximum a Posterior) 推定：$\theta^* = \mathrm{argmax}_\theta \log p(\mathcal{D}|\theta)p(\theta)$

- 認識論的不確実性を考慮した学習：
  - ➢$p(\theta|\mathcal{D})$の分布そのものを求めたい（分布推定）
  - ➢ただし解析的な導出な不可能

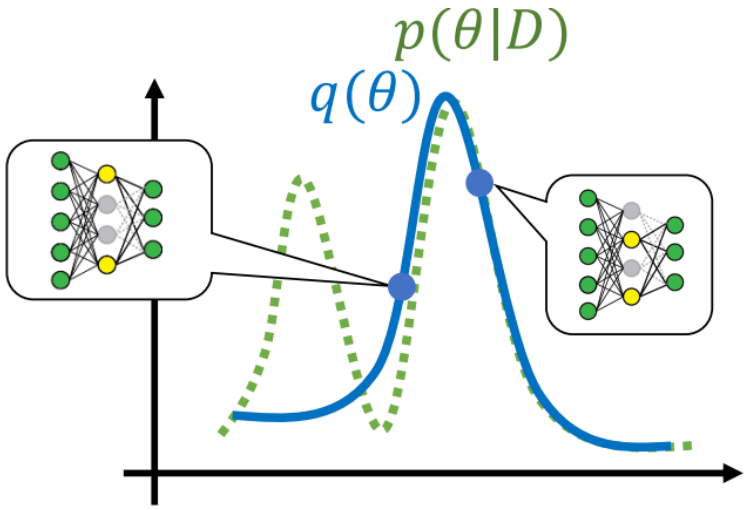
認識論的不確実性

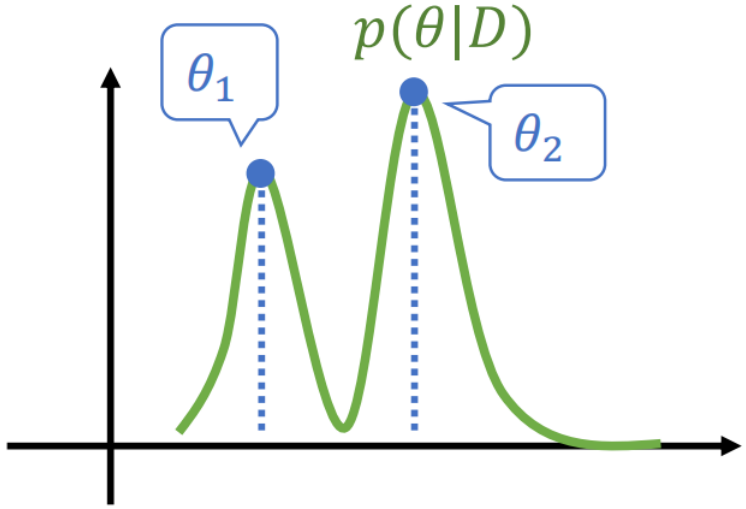
- 変分推論

➢扱いやすい変分分布 $q(\theta)$ を定義

➢真の分布を最も近似する $q^*(\theta)$ を導出

$$q^*(\theta) = \mathrm{argmin}_q D_{KL}[q(\theta)||p(\theta|\mathcal{D})]$$

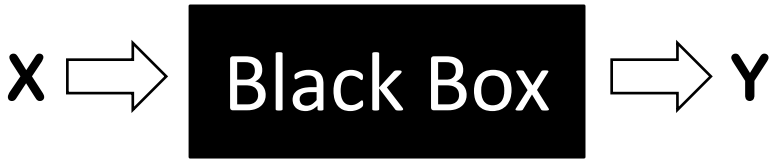




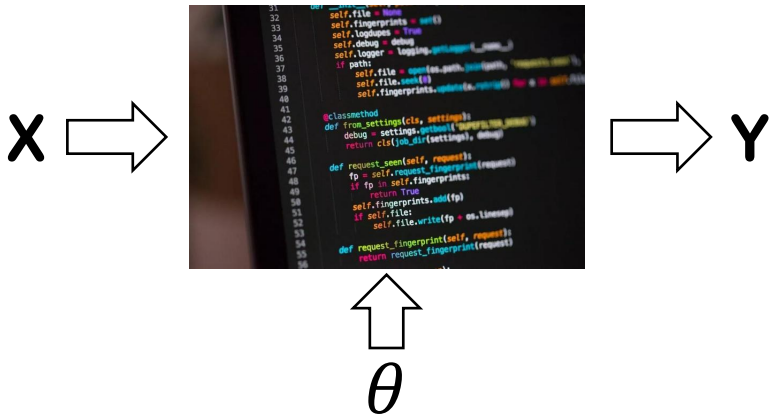
微分可能性

- 機械学習モデルは一般的にブラックボックスになりがち

➢産業応用には動作の説明可能性や動作保証が求められることが多いが、多くの機械学習モデルはそれを満足しない



- 扱う問題を解けるアルゴリズムが存在するのであれば、そちらを使うほうが望ましい



微分可能性

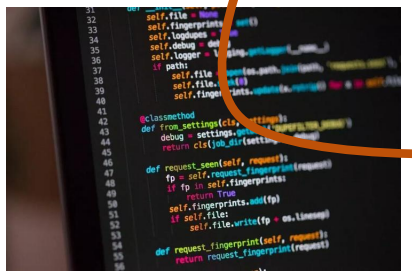
- アルゴリズムが存在したとしても、それが単純に適用できるケースは少ない

例:アルゴリズムのパラメータが未知

θ

$\frac{\mathrm{d}J}{\mathrm{d}\theta}$

例:入力が画像などの高次元データで直接アルゴリズムに入力できない

X ⇨  ⇨ Y ⇨ J(Y)

- アルゴリズムに必要なパラメータを「学習」できるか？

⇒ アルゴリズムが微分可能(differentiable)であれば可能！

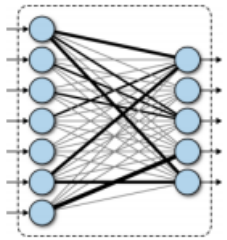
深層学習モデルの微分可能性

- 微分可能な関数の集合体（合成関数）
  - ➢ 微分可能な関数が直並列に並んだ（巨大な）ネットワーク
  - ➢ 連鎖律により全パラメータの微分を計算できる → 学習可能な関数

Fully-connected Layer

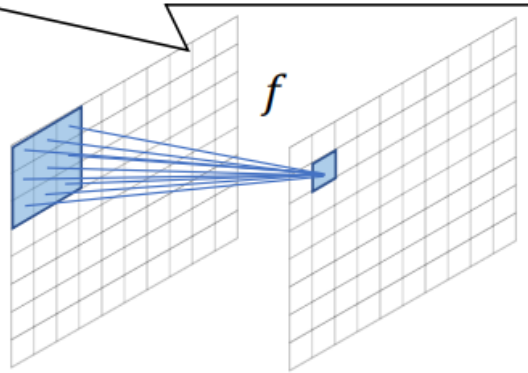
f は単なる積和演算

$$f(\boldsymbol{x}) = \boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b}$$



Convolutional Layer

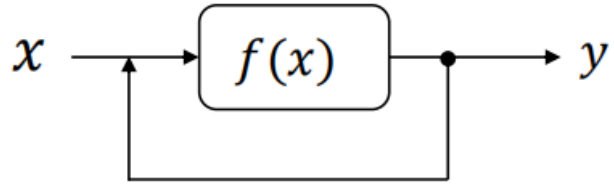
- 畳込みは単なる積和演算
- 同一演算を並列に複数回実行
  ＝ f を並列に接続



Recurrent Layer

同一関数を複数回実行
＝ f を直列に接続

$$y = f\left(f\left(f\left(\cdots\left(f(x)\right)\right)\right)\right)$$



ご清聴ありがとうございました