

連合学習  
*(Federated Learning)*

鹿島久嗣  
京都大学

# 目次： 連合学習

- 連合学習とは何か：
  - 連合学習の問題設定
  - 連合学習で何ができるか
  - 連合学習の歴史
- 教師付き学習の復習：最適化フレームワークと勾配法による学習
- 連合学習の方法：
  - 連合学習の問題設定
  - 連合学習の解法

# 連合学習とは何か

# 連合学習とはなにか：

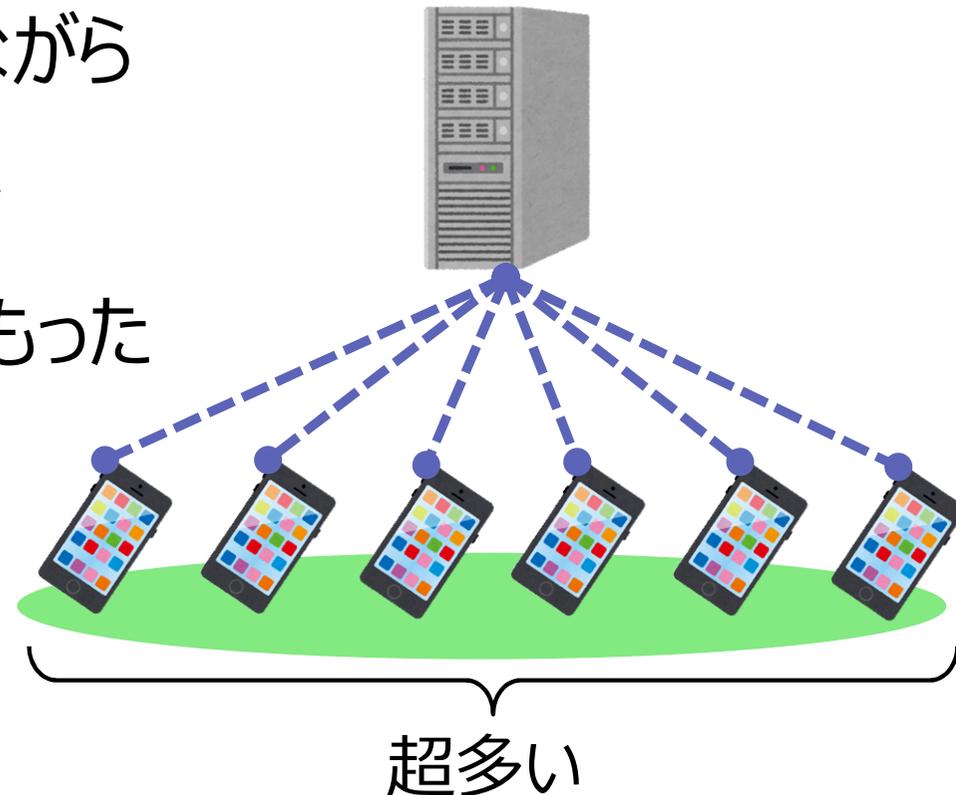
## 連合学習は分散機械学習のアプローチ

- 連合学習は：

- 大きな計算能力・記憶能力をもつ1台のサーバと
- データ収集能力とそこそこの計算資源をもつ多数のクライアントが
- 狭い帯域の回線を使って通信しながら

協力して効率的に学習をする枠組み

- クライアントは、センサと計算能力をもったモバイルデバイスをイメージ



## 連合学習には何ができるか：

できるモデルは通常の（分散しない）機械学習と同じ

- 最終的な成果物は通常の機械学習と同じだと思ってよい
- 全てのデータを一箇所（サーバ）に集めて機械学習モデルを学習するのと、同じことができる
  - ー ただし、さらにこれを各クライアントに個別適応したモデルを各クライアントにおいて作ることも可能
- 各クライアントで収集されたデータは、クライアントを出ない
  - ー 低通信負荷で学習を実現できる

# 連合学習の歴史：

## 本質的には10年前の技術の深層学習版

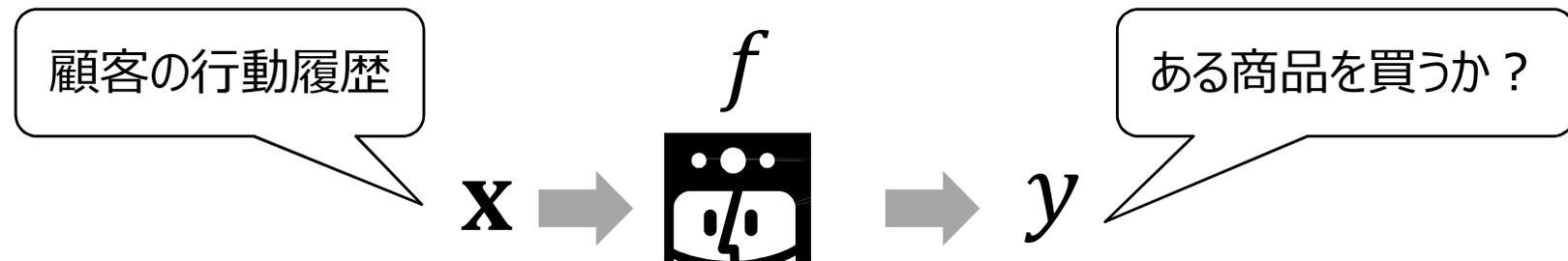
- 分散環境での機械学習（2000年代前半～）：
  - Google, Microsoft, Yahoo!などの企業を中心に研究がすすむ
  - 大規模なユーザデータを対象として、現実的な時間内で、大規模学習と個別化を行うという実用的な要請から発展
  - 応用：Gmailの優先トレイへの振り分けなど
- 連合学習（2015～）：
  - 上記の深層学習版として再登場
  - 応用：Gboard（キーボード）の予測入力
  - ほか、ヘルスケア応用など

# 教師付き学習の復習

## 教師付き学習：

入出力関係を再現する予測モデルをデータから推定

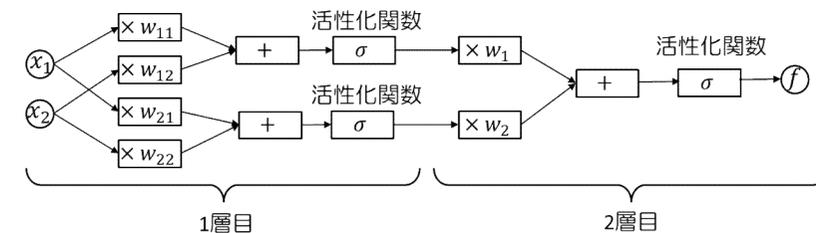
- 教師付き学習の予測モデル：入出力の関係  $f$ 
  - 入力  $\mathbf{x} \in \mathbb{R}^D$  は  $D$ 次元実数値ベクトル、出力  $y$  は1次元
    - 例：2値分類問題では  $y \in \{+1, -1\}$
- 教師付き学習：入出力ペアが与えられて、これから  $f$  を推定
  - $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$  :  $N$ 個の入出力ペア



# 教師付き学習のモデル：

## モデルパラメータによって決定される入出力関係

- 予測モデル（入出力の関係）  $f$  の具体的な形は用途に合わせていろいろ
  - 線形モデル： $y = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + w_0)$ 
    - パラメータ $\mathbf{w}$ と入力 $\mathbf{x}$ の線形関数で出力が決まる
  - ニューラルネットワーク：
    - 線形関数 + 非線形変換を積み上げて表現力を上げたモデル
    - パラメータ $\mathbf{w}$ と入力 $\mathbf{x}$ の関係が非線形



- モデルの詳細はともかく、パラメータ $\mathbf{w}$ が決まればモデル $f$ が決まる

# 教師付き学習の定式化の基本: 損失関数（と正則化項）の最小化

- 学習問題を最適化問題として定式化：
- モデルとデータの乖離度を損失関数として測る
  - 損失関数  $\ell_i(\mathbf{w})$  は  $i$  番目のデータ  $(\mathbf{x}_i, y_i)$  に対する乖離度
    - 例えば、損失関数を2乗誤差にすると：

$$\ell_i(\mathbf{w}) = (y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle + w_0))^2$$

- 訓練データ全体に対する損失関数の平均  $L(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N \ell_i(\mathbf{w})$  を目的関数として最小化するパラメータ  $\mathbf{w}$  を求める

※しばしば、訓練データへの過剰な適合を防ぐための正則化項  $R(\mathbf{w})$  を入れるが、今回は省略する

# 教師付き学習の解法: 勾配法による最適化

- パラメータの最適化は勾配法によって行われる :

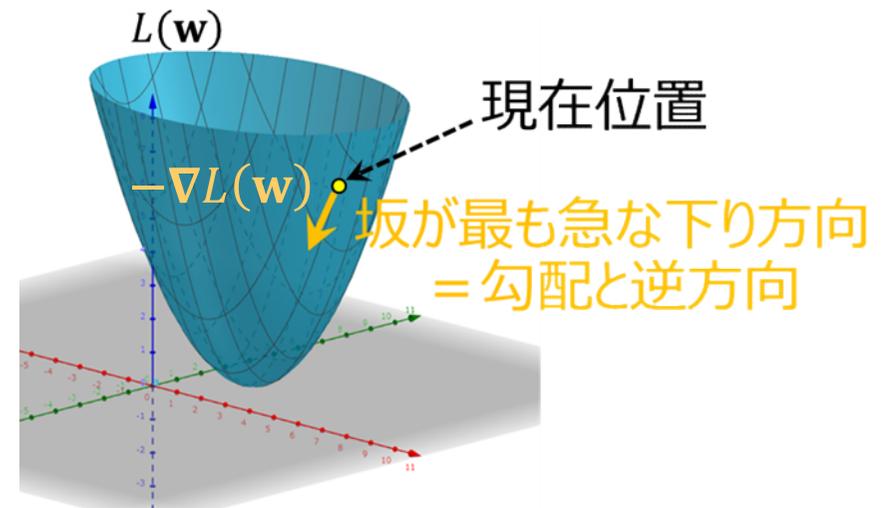
$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla L(\mathbf{w}) = \mathbf{w} - \eta \frac{1}{N} \sum_{i=1}^N \nabla \ell_i(\mathbf{w})$$

– 各データ  $i$  に対するパラメータの勾配  $\nabla \ell_i(\mathbf{w})$  が計算できればパラメータ更新可能

- 確率的勾配法 :

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla L(\mathbf{w}) \approx \mathbf{w} - \eta \nabla \ell_i(\mathbf{w})$$

– データひとつで勾配を近似



# 連合学習の方法

# 連合学習の問題設定：

データが分散している状況で、データを移さずに学習したい

- データが（簡単のため）2つに分割されている状況で学習する
- 前半の $N$ 個をクライアント1が、後半の $M$ 個をクライアント2が担当

$$L(\mathbf{w}) = \frac{1}{N+M} \sum_{i=1}^{N+M} \ell_i(\mathbf{w})$$
$$= \frac{1}{N+M} \underbrace{\sum_{i=1}^N \ell_i(\mathbf{w})}_{\text{クライアント1}} + \frac{1}{N+M} \underbrace{\sum_{i=N+1}^{N+M} \ell_i(\mathbf{w})}_{\text{クライアント2}}$$

- ルール：データはクライアントから出せない

# 勾配法の適用： 各クライアントで分散計算可能

- 全体の勾配は、それぞれのクライアントの持つデータの勾配を足して計算できる

$$\begin{aligned}\nabla L(\mathbf{w}) &= \frac{1}{N+M} \sum_{i=1}^{N+M} \nabla \ell_i(\mathbf{w}) \\ &= \frac{1}{N+M} \underbrace{\sum_{i=1}^N \nabla \ell_i(\mathbf{w})}_{\text{クライアント1の勾配}} + \frac{1}{N+M} \underbrace{\sum_{i=N+1}^{N+M} \nabla \ell_i(\mathbf{w})}_{\text{クライアント2の勾配}}\end{aligned}$$



クライアント1の勾配



クライアント2の勾配

- ルール：データをクライアントから出さなくても勾配計算できる

## 勾配法の適用における問題点： 更新のたびに集めるのはコストが高い

- 全体モデルのパラメータ更新式：

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \left( \frac{1}{N+M} \sum_{i=1}^N \nabla \ell_i(\mathbf{w}) + \frac{1}{N+M} \sum_{i=N+1}^{N+M} \nabla \ell_i(\mathbf{w}) \right)$$

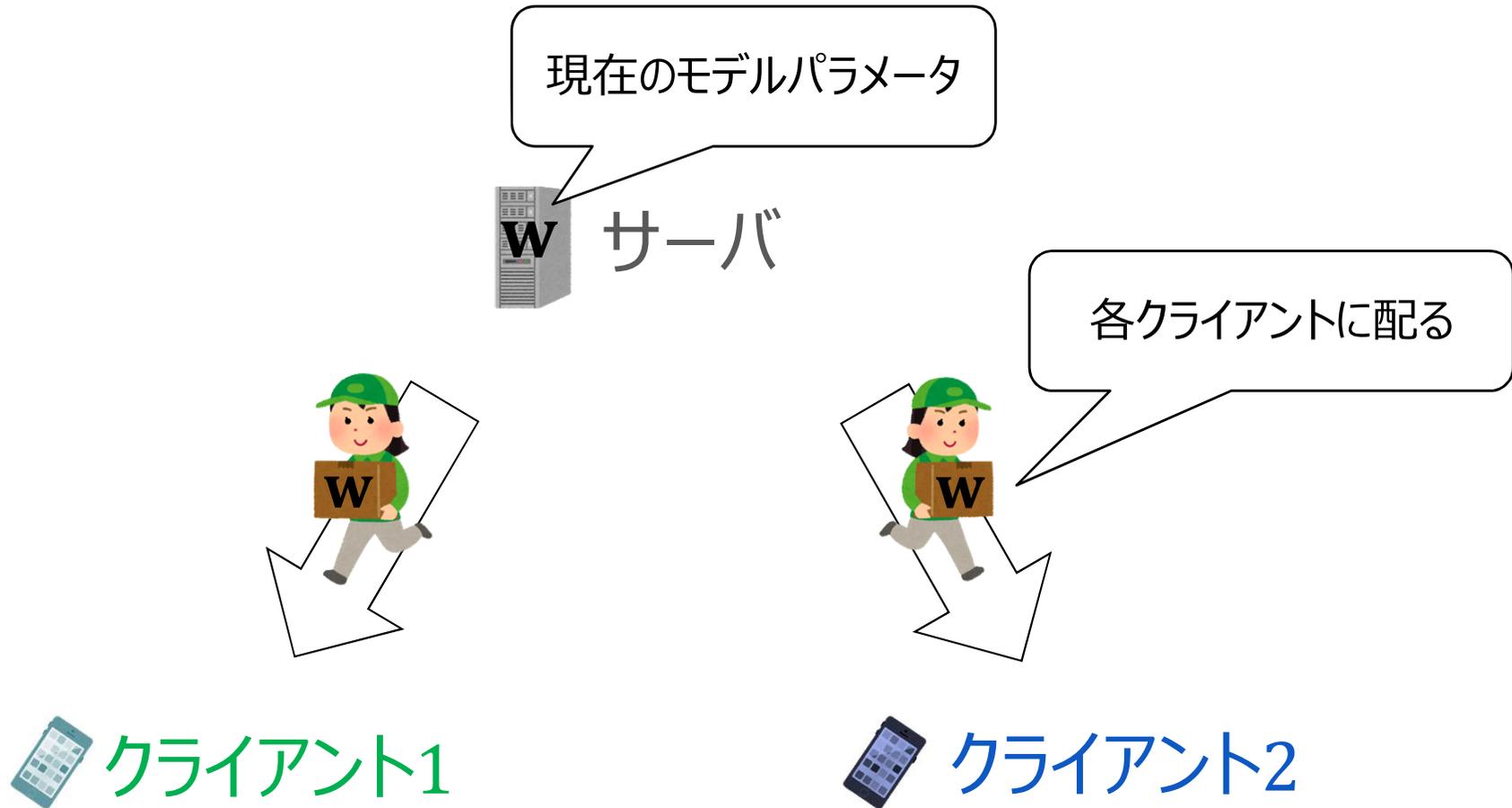
- パラメータ更新のたびにクライアントから勾配を集めるのはコストが高い
- 各クライアントで学習し終わってから集めて平均をとればよい？  
→ これは、全体モデルの最適解ではない...

## 連合学習のアプローチ：

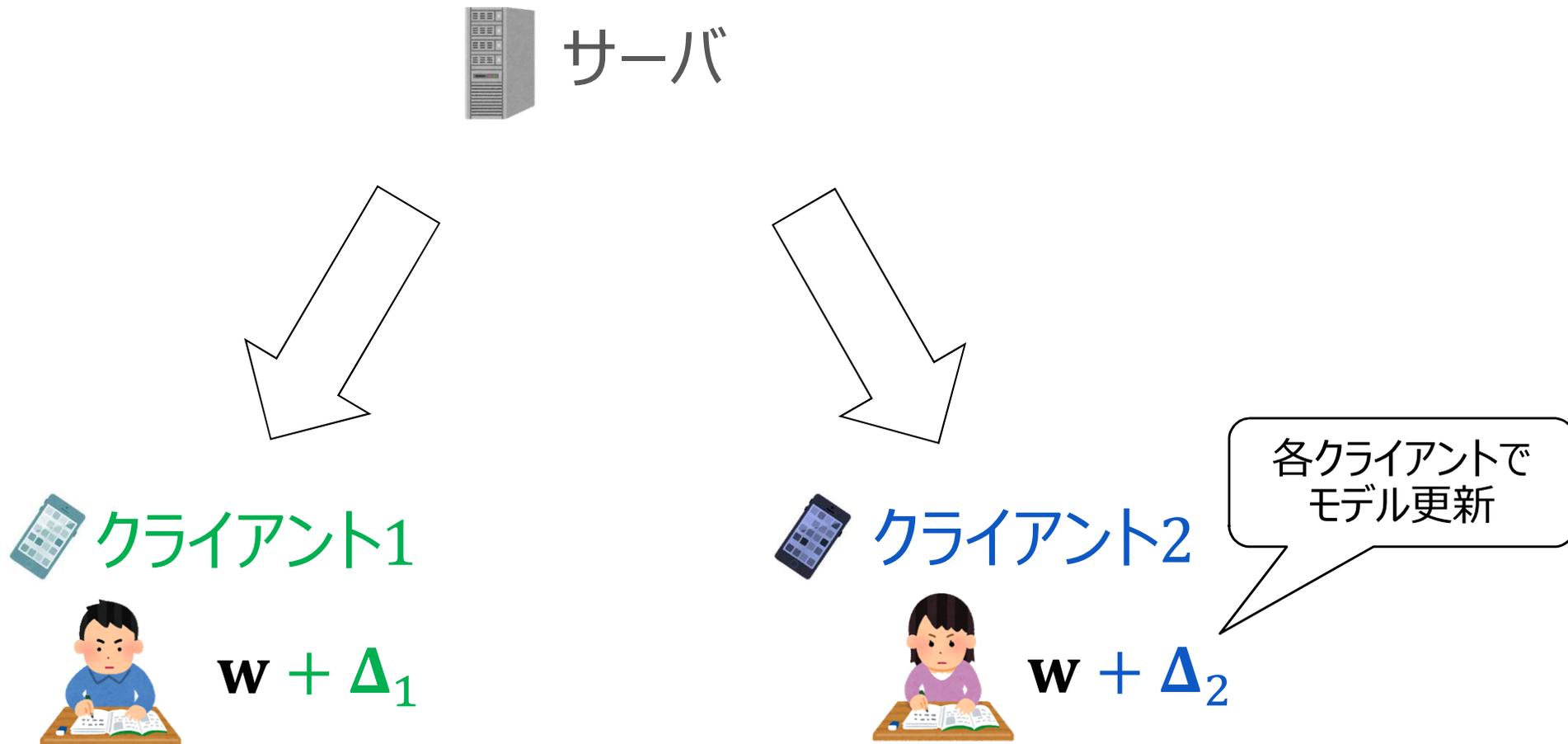
サーバとクライアントがモデルをやり取りしながら学習を進める

- 基本戦略：各クライアントで別々に学習を進め、時々持ち寄って同期をとる
- 学習は以下のステップを**繰り返す**ことで行われる：
  1. サーバは、現在のモデルパラメータをクライアントに配る
  2. 各クライアントは、自身のデータでモデルを更新する
  3. 各クライアントは、モデルの更新差分をサーバに送信する
  4. サーバは、集まった差分を、データ量で重みつき平均をとり、全体モデルを更新する
  5. ステップ1にもどる

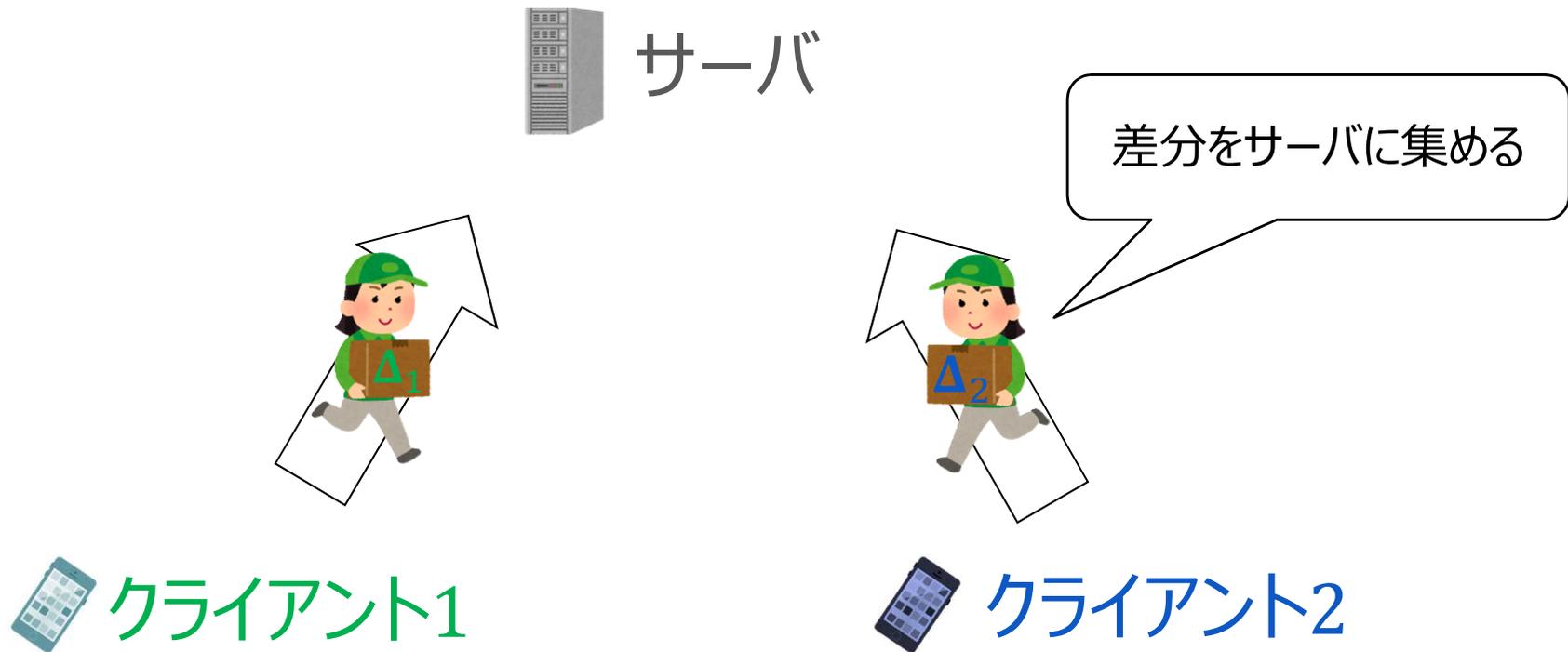
# 連合学習のアプローチ（ステップ1）： サーバは、現在のモデルパラメータをクライアントに配る



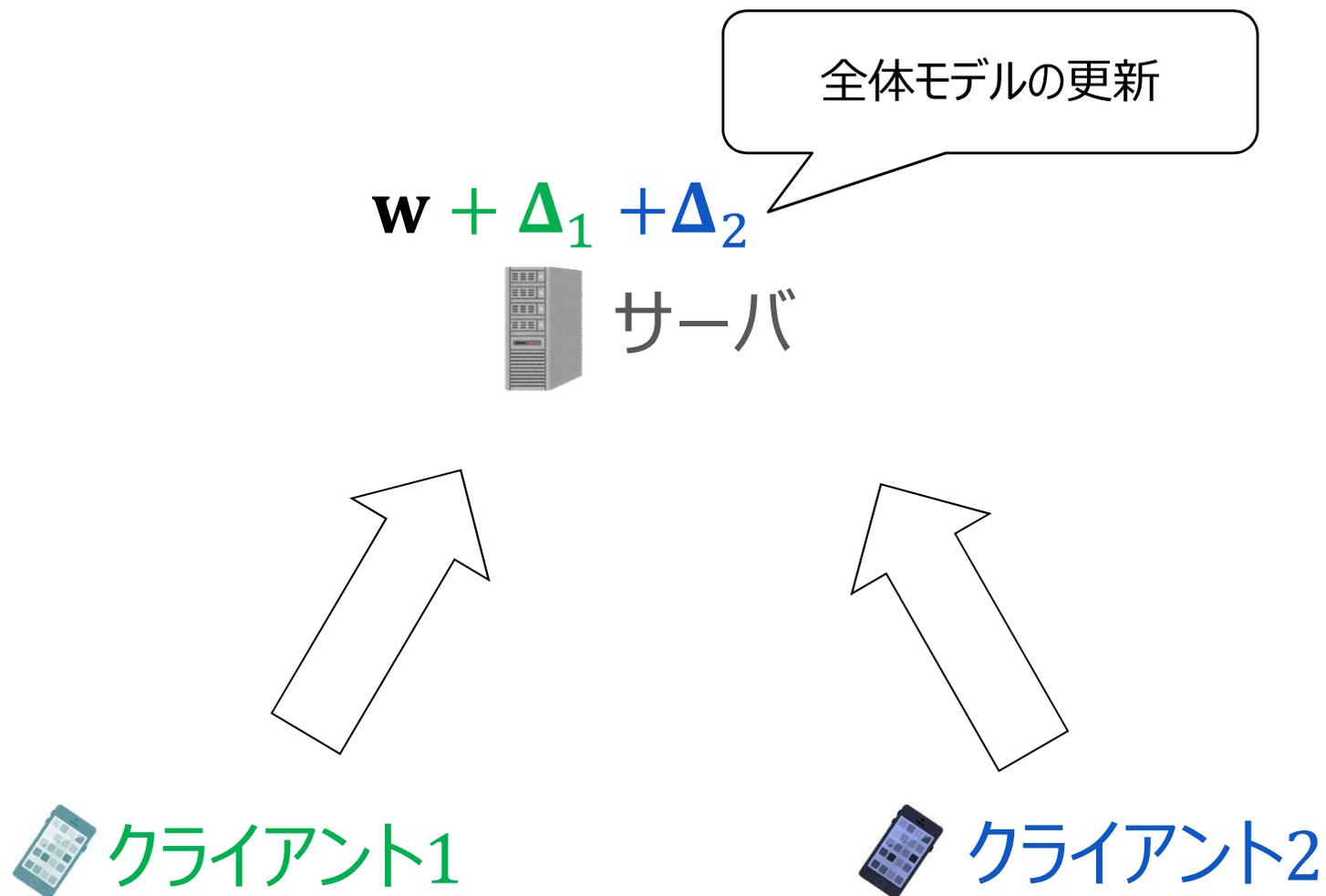
# 連合学習のアプローチ（ステップ2）： 各クライアントは、自身のデータでモデルを更新する



# 連合学習のアプローチ（ステップ3）： 各クライアントは、モデルの更新差分をサーバに送信する



# 連合学習のアプローチ（ステップ4）： サーバは、集まった差分で、全体モデルを更新する



⇒ ステップ1にもどる

## 連合学習のポイント：

サーバとクライアントはデータを送信しない

- 学習は以下のステップを繰り返すことで行われる：
  1. サーバは、現在の**モデルパラメータをクライアントに配る**
  2. 各クライアントは、自身のデータでモデルを更新する
  3. 各クライアントは、**モデルの更新差分をサーバに送信する**
  4. サーバは、集まった差分を、データ量で重みつき平均をとり、全体モデルを更新する
- いずれのステップにおいてもモデルパラメータが送信される
  - データサイズ  $\gg$  モデルパラメータ なので通信料がはるかに小さい
  - データを送信しないので、プライバシーが守られやすい

# まとめ

# 結局、連合学習には何が出来るか： 分散環境で集まるデータから、低通信コストで学習

- 連合学習が想定される状況：
  - データ収集が各モバイルデバイスレベルで分散して行われる状況
    - もともとのデータはエッジ側にある
  - 全部サーバにデータを集めて学習をすると通信料が大きすぎる
    - プライバシの問題は副次的か
  - スマホのようなエッジでの学習がある程度可能な状況
    - エッジの計算能力も活用したい
- できること：
  - 全データをサーバに集めることなく、集めた場合と同じモデルを、低通信コストで作ることができる
  - 各クライアントに個別適応したモデルも得られる

# まとめ： 連合学習

- 連合学習とは何か：
  - 連合学習の問題設定：分散データ収集環境での分散学習
  - 連合学習で何ができるか：低通信コストでの学習
  - 連合学習の歴史：かつての分散学習の深層学習バージョン
- 教師付き学習の復習：最適化フレームワークと勾配法による学習
- 連合学習の方法：
  - 連合学習の問題設定：勾配法の分散適用
  - 連合学習の解法：勾配の分散計算と同期