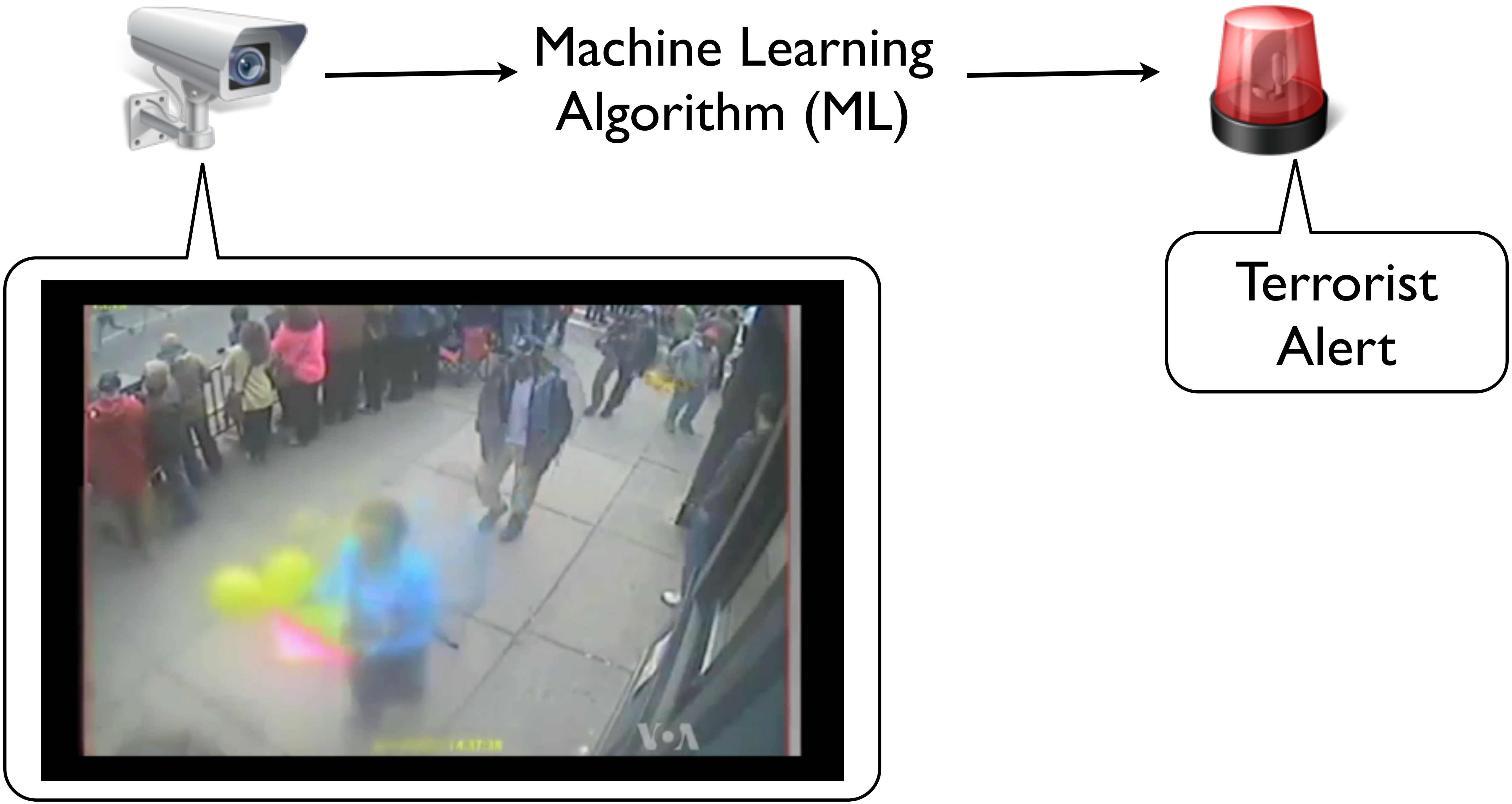
Scalable and Fast Machine Learning

By Niketan Pansare (np6@rice.edu)

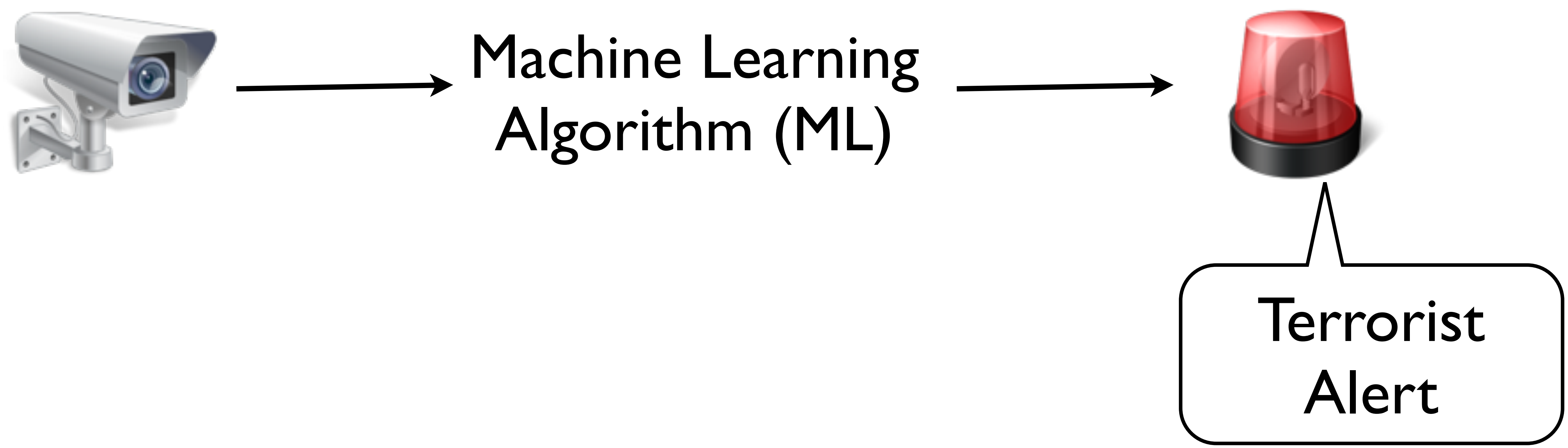
Why is it important ?

Terrorist Alert

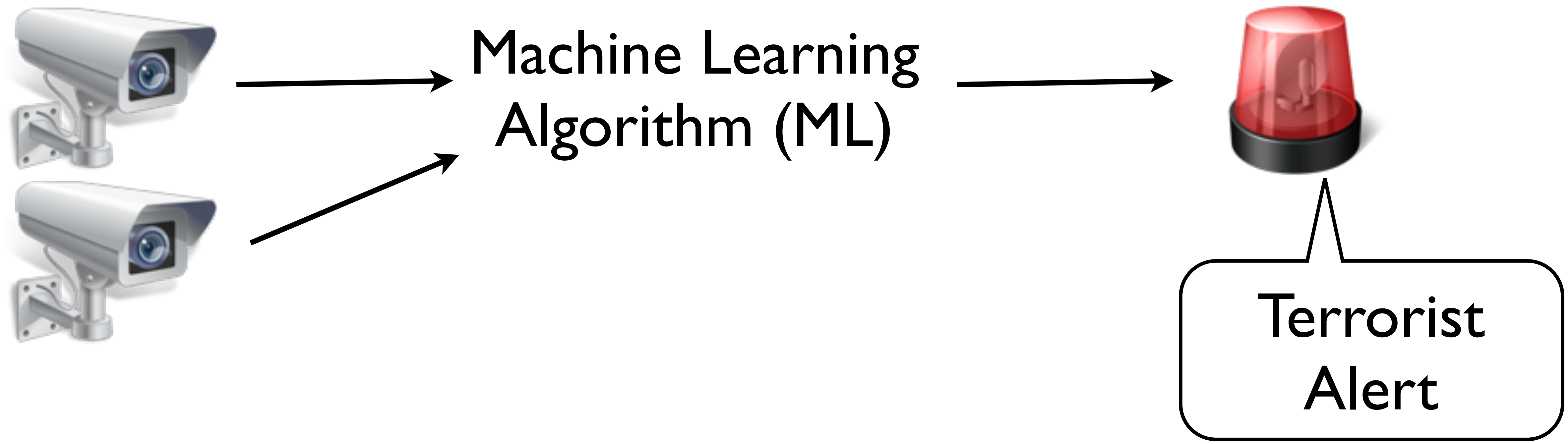
Why is it important ?



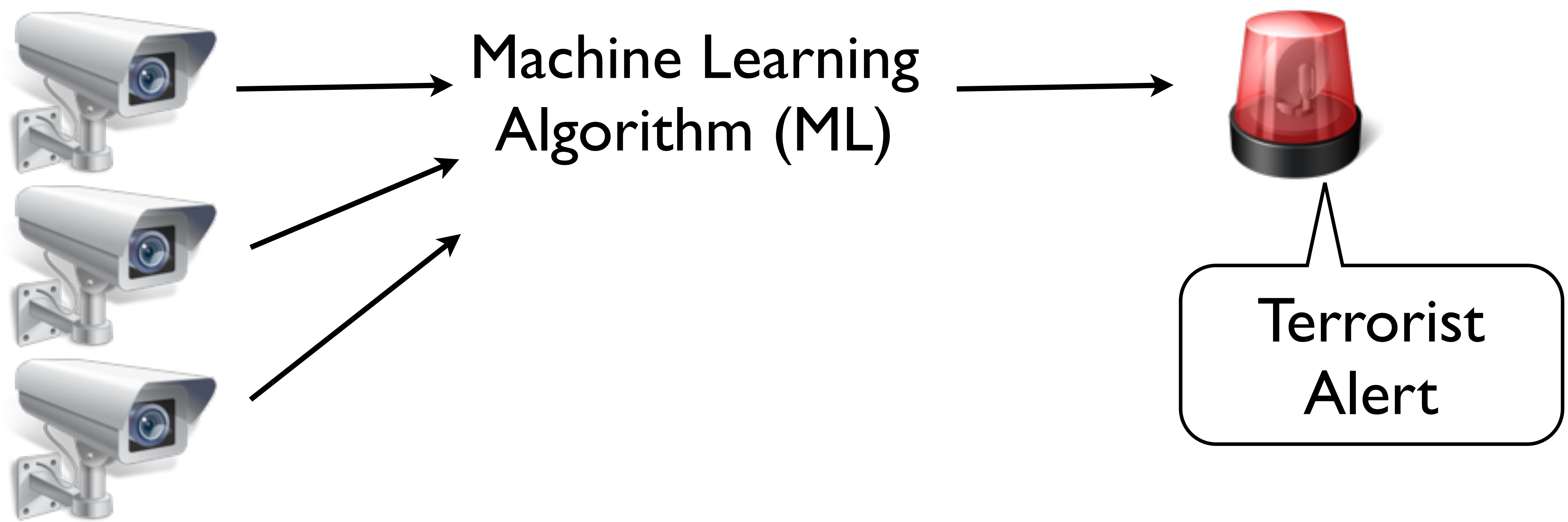
Challenges



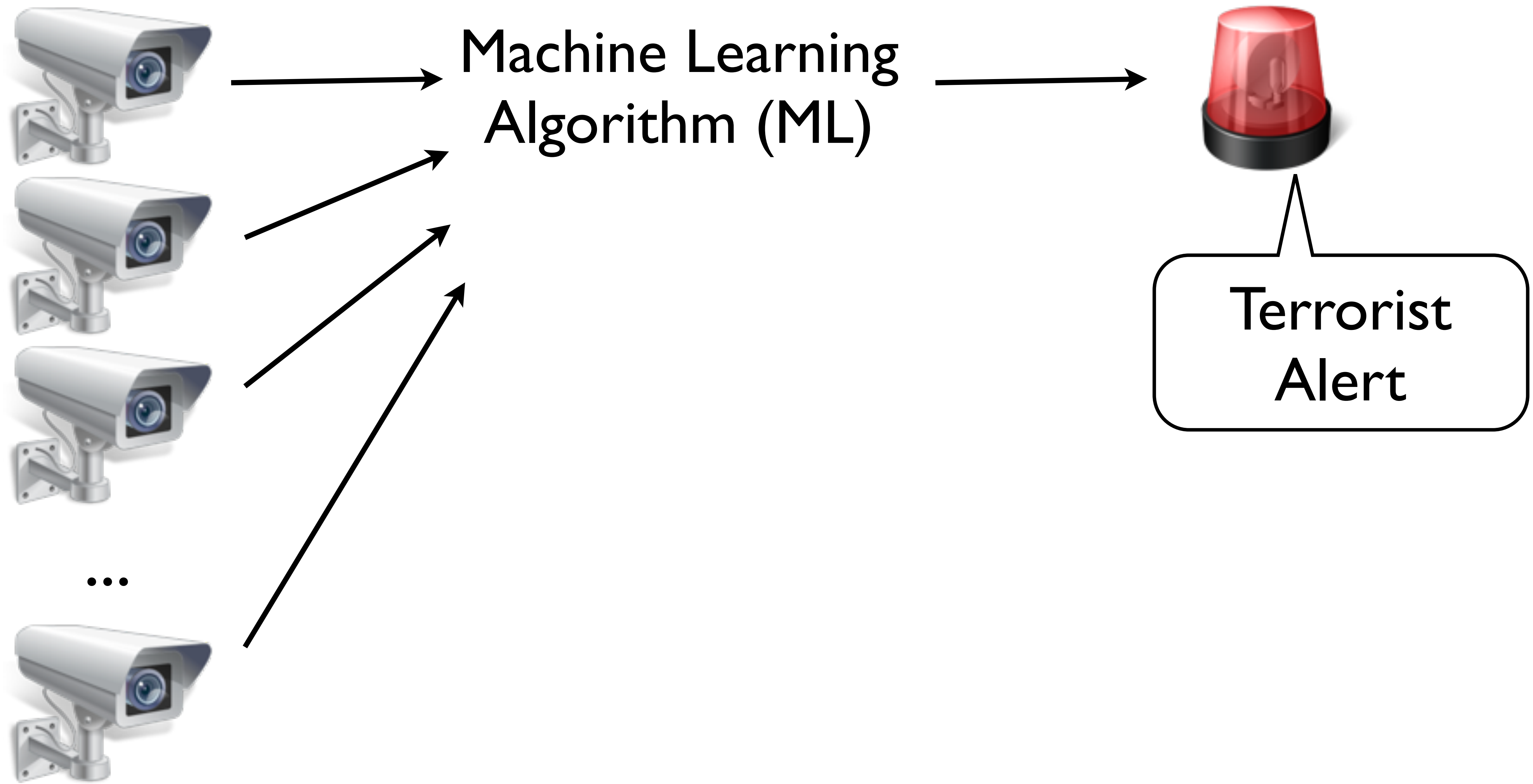
Challenges



Challenges

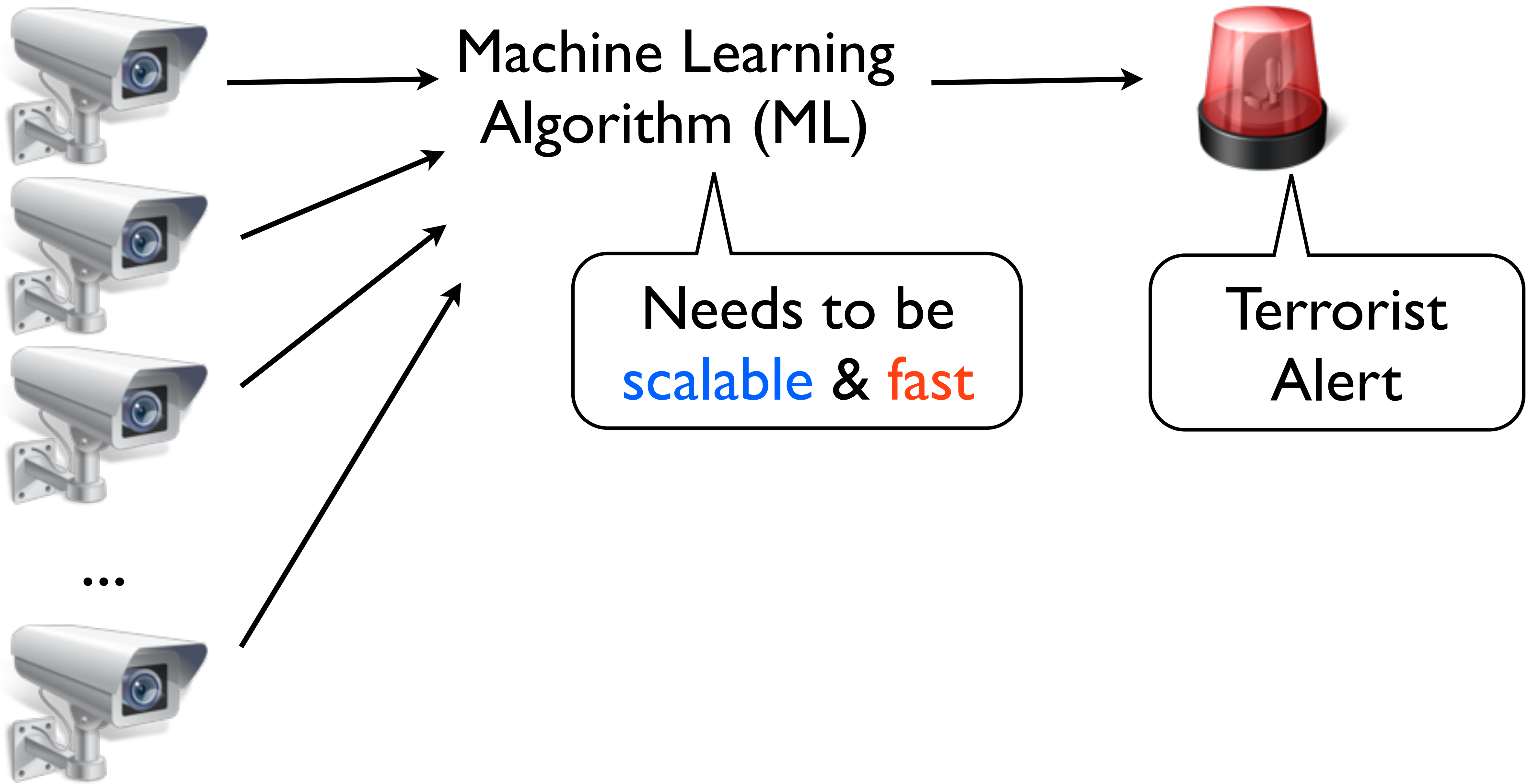


Challenges



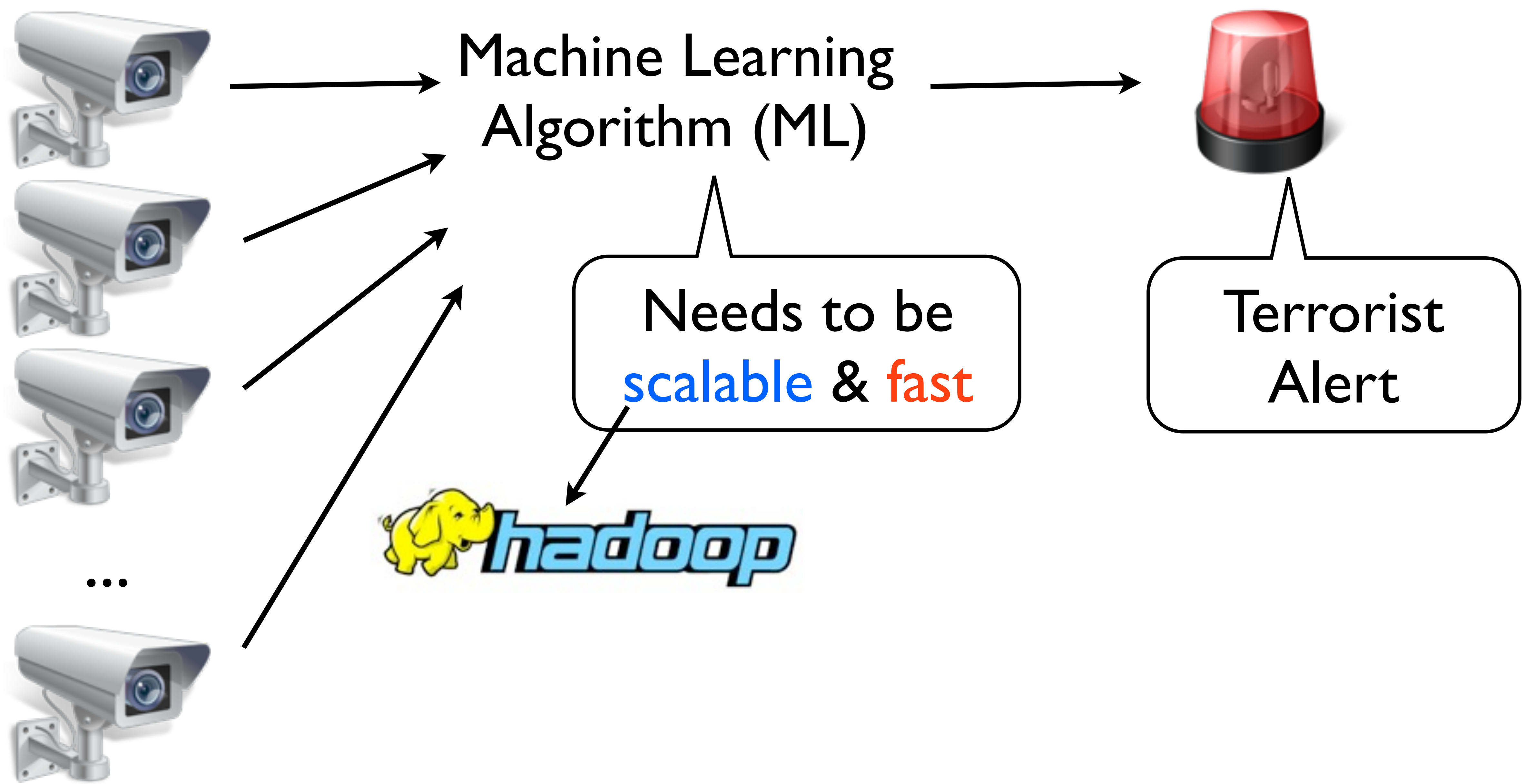
100 hours of videos uploaded on
Youtube per minute !!!

Challenges



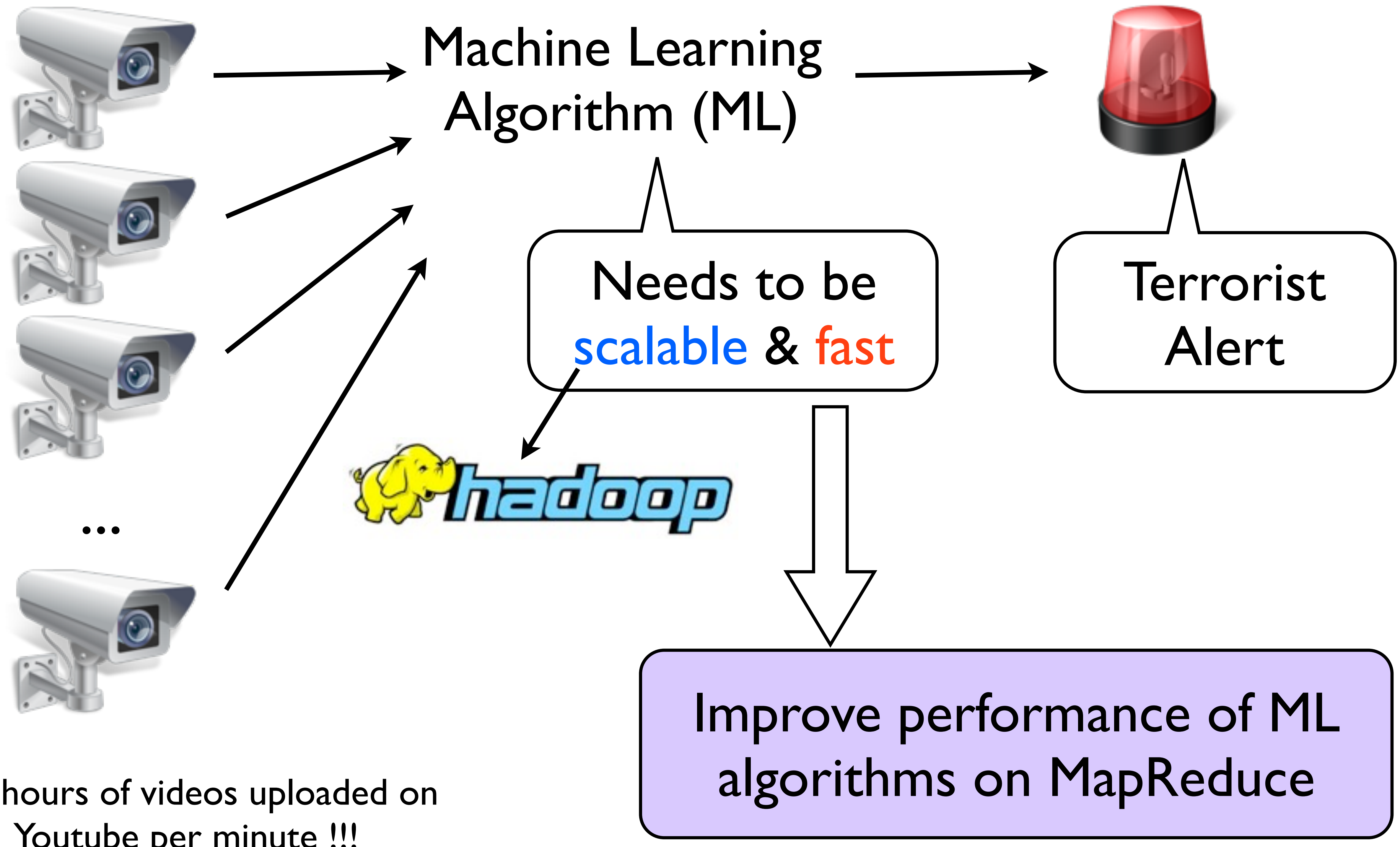
100 hours of videos uploaded on Youtube per minute !!!

Challenges



100 hours of videos uploaded on
Youtube per minute !!!

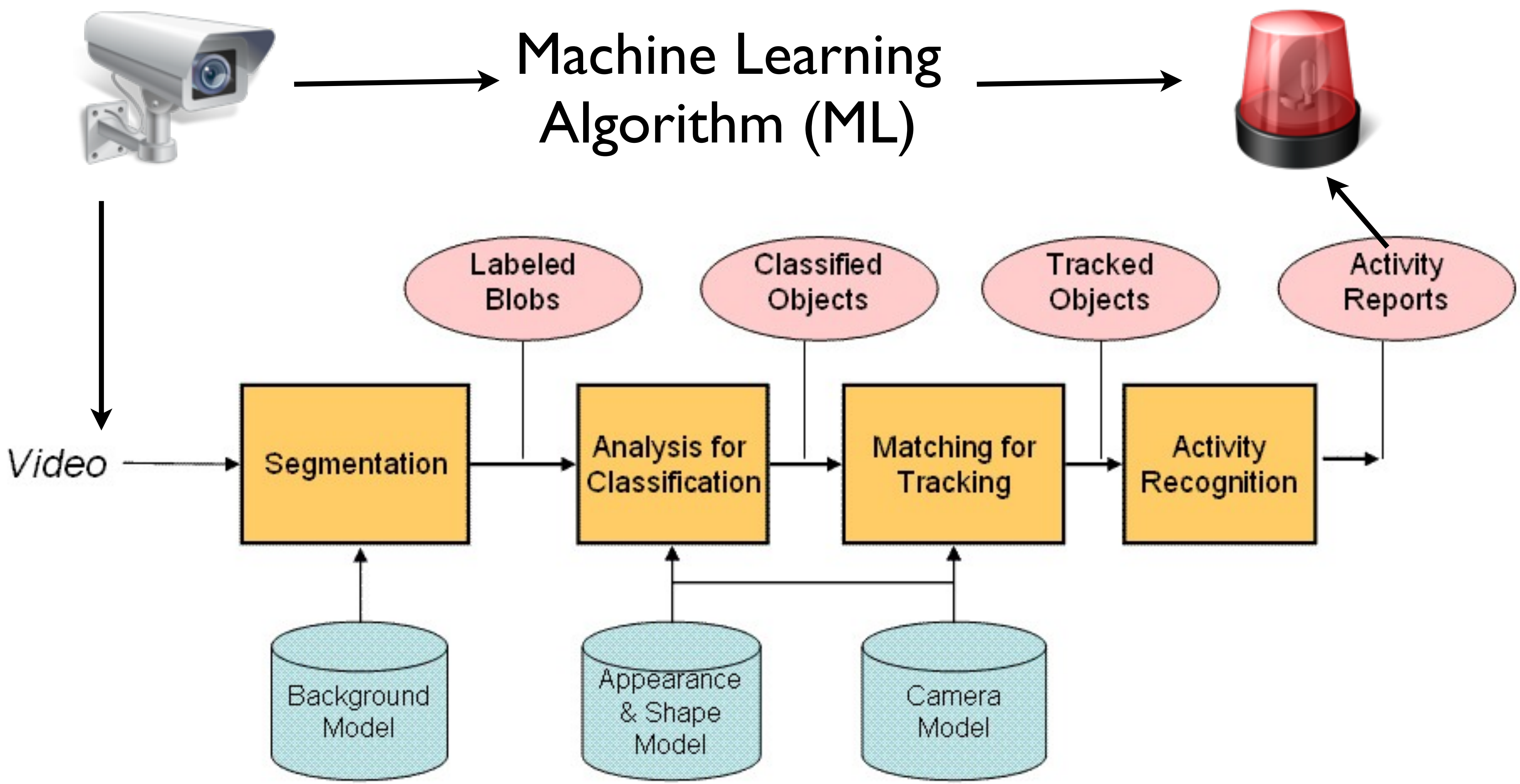
Challenges


Machine Learning Algorithm (ML)
Needs to be scalable & fast
Terrorist Alert
...
hadoop
Improve performance of ML algorithms on MapReduce
100 hours of videos uploaded on Youtube per minute !!!

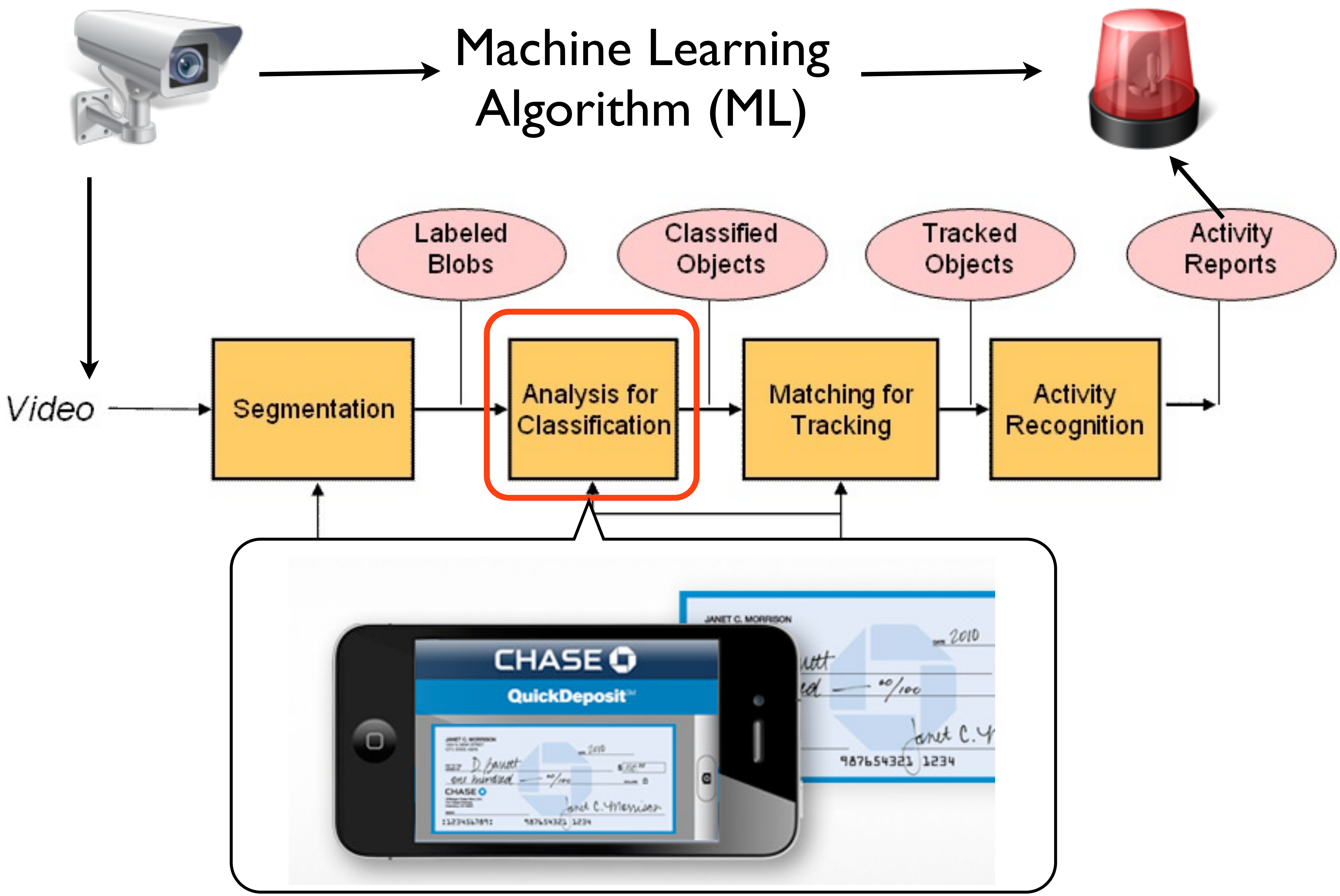
Machine Learning in little more detail

Machine Learning
Algorithm (ML)

Machine Learning in little more detail



Machine Learning in little more detail



Machine Learning in little more detail

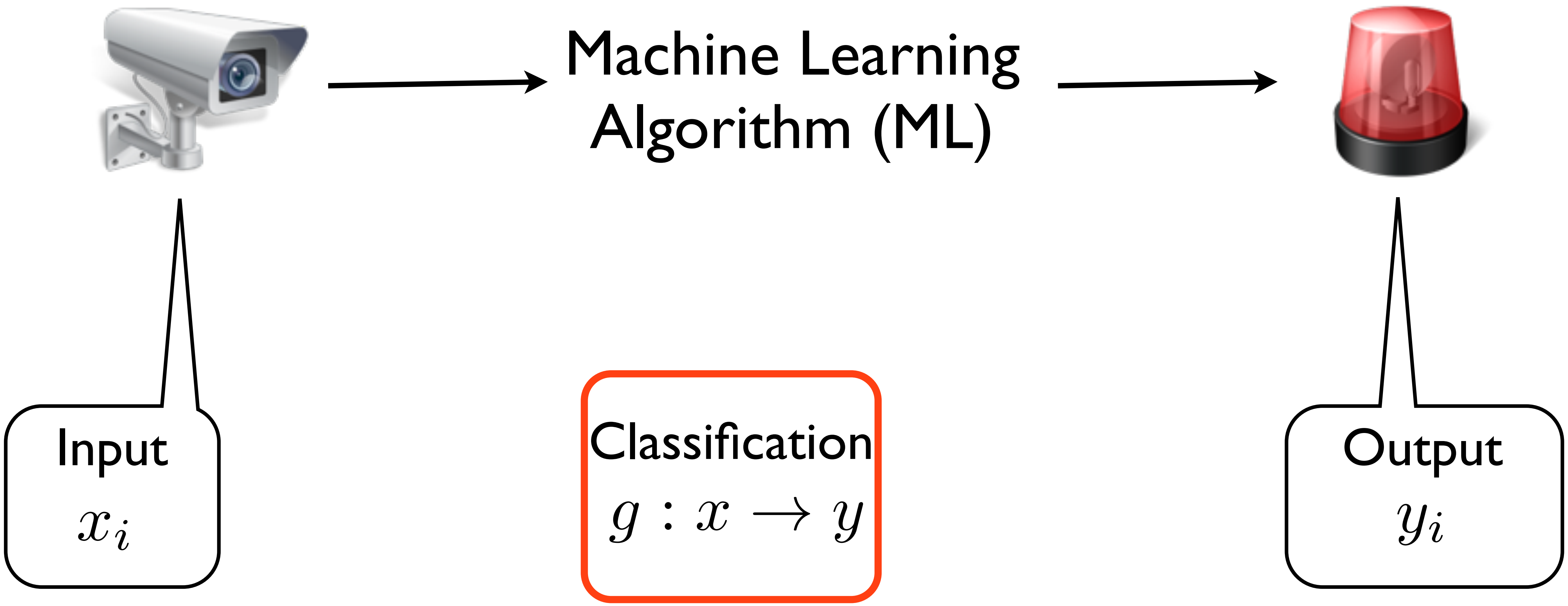
Machine Learning Algorithm (ML)

Input

Classification

Output

Machine Learning in little more detail



Key idea of ML:
Learn predictor g such that the loss $L(y, g(x))$ over the data is minimum.

Machine Learning in little more detail

Machine Learning Algorithm (ML)

Input
x_i

Classification
$g : x \rightarrow y$

Output
y_i

Key idea of ML:
Learn predictor g such that the loss $L(y, g(x))$ over the data is minimum.

=> One way to do this is using gradient descent algorithm.

Machine Learning in little more detail, but with example

Machine Learning in little more detail, but with example

Consider "Linear regression", so learn w using gradient descent such that $\text{predictor } g = w^T x$

$$\text{loss } L = (y - w^T x)^2$$

$$\nabla L(w_i; x_j, y_j) = 2(y - w^T x)w$$

Machine Learning in little more detail, but with example

Consider "Linear regression", so learn w using gradient descent such that $\text{predictor } g = w^T x$

$$\text{loss } L = (y - w^T x)^2$$

$$\nabla L(w_i; x_j, y_j) = 2(y - w^T x)w$$

<u>Gradient descent setup:</u>

Initialize w_0
while not converged {

}

Machine Learning in little more detail, but with example

Consider "Linear regression", so learn w using gradient descent such that $\text{predictor } g = w^T x$

$$\text{loss } L = (y - w^T x)^2$$

$$\nabla L(w_i; x_j, y_j) = 2(y - w^T x)w$$

<u>Gradient descent setup:</u>

Initialize w_0
while not converged {
$$w_{i+1} = w_i - \alpha\, \mathbb{E}[\nabla L_{j \in B}]$$
}

learning rate

Estimate of gradient over the loss using datapoints in set B

Machine Learning in little more detail, but with example

Consider "Linear regression", so learn w using gradient descent such that predictor $g = w^T x$

$$\text{loss } L = (y - w^T x)^2$$

$$\nabla L(w_i; x_j, y_j) = 2(y - w^T x)w$$

Gradient descent setup:

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

For regularized version, replace w_i by $w_i(1 - \alpha\lambda)$

Examples other than linear regression:

SVM:

$$\begin{aligned} w_{i+1} &= w_i - \alpha_i \lambda w_i && \text{if } y_j w^T \phi(x_j) > 1 \\ &= w_i - \alpha_i y_j \phi(x_j) && \text{otherwise} \end{aligned}$$

Guassian mixture model:

$$w = [\pi; \mu_0 ... \mu_K; \sigma_0 ... \sigma_K]$$

$$\nabla w = [\nabla\pi; \nabla\mu_0 ... \nabla\mu_K; \nabla\sigma_0 ... \nabla\sigma_K]$$

$$\nabla \mu_k[j] = \frac{-1}{N} \sum_{n=0}^{N-1} \frac{frac * (X_n[j] - \mu_k[j])}{(\sigma_k[j])^2}$$

$$\nabla \sigma_k[j] = \frac{-1}{N} \sum_{n=0}^{N-1} frac * \left\{ \frac{(X_n[j] - \mu_k[j])^2}{(\sigma_k[j])^3} - \frac{1}{\sigma_k[j]} \right\}$$

$$\nabla \pi_k = \frac{1}{N} \left\{ \sum_{n=0}^{N-1} |frac| * \left[\frac{1}{|\pi_k|} - \frac{1}{\sum_{k=1}^{K} |\pi_k|} \right] \right\} - \frac{2 * (\sum_{k=1}^{K} |\pi_k| - 1) * \pi_k}{|\pi_k|}$$

where $frac = getProbOfClusterK(k, X_n, w)$

Examples other than linear regression:

Perceptron: $$\begin{aligned} w_{i+1} &= w_i + \alpha_i y_j \phi(x_j) && \text{if } y_j w^T \phi(x_j) \leq 0 \\ &= w_i && \text{otherwise} \end{aligned}$$

Adaline: $$w_{i+1} = w_i + \alpha_i (y_j - w^T \phi(x_j)) \phi(x_j)$$

K-means: $$\begin{aligned} k* &= \arg\ \min_k (z_i - w_k)^2 \\ n_{k*} &= n_{k*} + 1 \\ w_{k*} &= w_{k*} + \frac{1}{n_{k*}} (z_i - w_{k*}) \end{aligned}$$

Lasso: $$L = \lambda |w|_1 + \frac{1}{2} (y - w^T \phi(x))^2$$ ... regularized least squares

Related work (variants of GD)

Gradient descent setup:

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha\ \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

- Constant learning rate

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0
while not converged {
$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$
}

learning rate

Estimate of gradient over the loss using datapoints in set B

- Constant learning rate

- Start with small learning rate

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0
while not converged {
$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$
}

learning rate

Estimate of gradient over the loss using datapoints in set B

- Constant learning rate

- Start with small learning rate
 => Increase if successive gradients in same direction, else decrease.

3 key areas:
1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0
while not converged {
$$w_{i+1} = w_i - \alpha\, \mathbb{E}[\nabla L_{j \in B}]$$
}

learning rate

Estimate of gradient over the loss using datapoints in set B

- Constant learning rate

- Start with small learning rate
  => Increase if successive gradients in same direction, else decrease.

- Monitor loss on validation set and increase/decrease rate accordingly

3 key areas:
1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

- Constant learning rate

- Start with small learning rate
  => Increase if successive gradients in same direction, else decrease.

- Monitor loss on validation set and increase/decrease rate accordingly

- Find learning rate through line search on sample dataset

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0
while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

- Constant learning rate

- Start with small learning rate
  => Increase if successive gradients in same direction, else decrease.

- Monitor loss on validation set and increase/decrease rate accordingly

- Find learning rate through line search on sample dataset

- Decrease rate according to heuristic formula:

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

<u>Gradient descent setup:</u>

Initialize w_0
while not converged {
$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$
}

learning rate

Estimate of gradient over the loss using datapoints in set B

- Constant learning rate

- Start with small learning rate
 => Increase if successive gradients in same direction, else decrease.

- Monitor loss on validation set and increase/decrease rate accordingly

- Find learning rate through line search on sample dataset

- Decrease rate according to heuristic formula: $\alpha_i = \dfrac{\alpha_0}{1 + \alpha_0 \lambda i}$

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0
while not converged {
$$w_{i+1} = w_i - \alpha\, \mathbb{E}[\nabla L_{j \in B}]$$
}

learning rate

Estimate of gradient over the loss using datapoints in set B

- HogWild [Niu 2011]

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0
while not converged {
$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$
}

learning rate

Estimate of gradient over the loss using datapoints in set B

- HogWild [Niu 2011]
 => If data is sparse, just keep updating w without any lock.

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0
while not converged {
$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$
}

learning rate

Estimate of gradient over the loss using datapoints in set B

- HogWild [Niu 2011]
 => If data is sparse, just keep updating w without any lock.

- Distributed SGD [Gemulla 2011]

3 key areas:
1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0
while not converged {
$$w_{i+1} = w_i - \alpha\, \mathbb{E}[\nabla L_{j \in B}]$$
}

learning rate

Estimate of gradient over the loss using datapoints in set B

3 key areas:
1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

- B = entire dataset => Batch GD

$$w_{i+1} = w_i - \alpha \left\{ \frac{1}{n} \sum_{j=1}^{n} \nabla L(w_i; x_j, y_j) \right\}$$

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

- B = entire dataset => Batch GD

$$w_{i+1} = w_i - \alpha \left\{ \frac{1}{n} \sum_{j=1}^{n} \nabla L(w_i; x_j, y_j) \right\}$$

- B = 1 random block

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0
while not converged {
$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$
}

learning rate

Estimate of gradient over the loss using datapoints in set B

- B = entire dataset => Batch GD

$$w_{i+1} = w_i - \alpha \left\{ \frac{1}{n} \sum_{j=1}^{n} \nabla L(w_i; x_j, y_j) \right\}$$

- B = 1 random block
  => sum over b data items instead of n

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

- B = entire dataset => Batch GD

$$w_{i+1} = w_i - \alpha \left\{ \frac{1}{n} \sum_{j=1}^{n} \nabla L(w_i; x_j, y_j) \right\}$$

- B = 1 random block
  => sum over b data items instead of n
  => Mini-batch GD

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha\, \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

- B = entire dataset => Batch GD

$$w_{i+1} = w_i - \alpha \left\{ \frac{1}{n} \sum_{j=1}^{n} \nabla L(w_i; x_j, y_j) \right\}$$

- B = 1 random block
  => sum over b data items instead of n
  => Mini-batch GD

- B = 1 random datapoint

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

<u>Gradient descent setup:</u>

Initialize w_0
while not converged {
$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$
}

learning rate

Estimate of gradient over the loss using datapoints in set B

- B = entire dataset => Batch GD

$$w_{i+1} = w_i - \alpha \left\{ \frac{1}{n} \sum_{j=1}^{n} \nabla L(w_i; x_j, y_j) \right\}$$

- B = 1 random block
  => sum over b data items instead of n
  => Mini-batch GD

- B = 1 random datapoint
  => Stochastic GD

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

Gradient descent setup:

Initialize w_0
while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

- B = entire dataset => Batch GD

$$w_{i+1} = w_i - \alpha \left\{ \frac{1}{n} \sum_{j=1}^{n} \nabla L(w_i; x_j, y_j) \right\}$$

- B = 1 random block
  => sum over b data items instead of n
  => Mini-batch GD

- B = 1 random datapoint
  => Stochastic GD
  => Not suitable for MapReduce

3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Related work (variants of GD)

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

learning rate

Estimate of gradient over the loss using datapoints in set B

- B = entire dataset => Batch GD

$$w_{i+1} = w_i - \alpha \left\{ \frac{1}{n} \sum_{j=1}^{n} \nabla L(w_i; x_j, y_j) \right\}$$

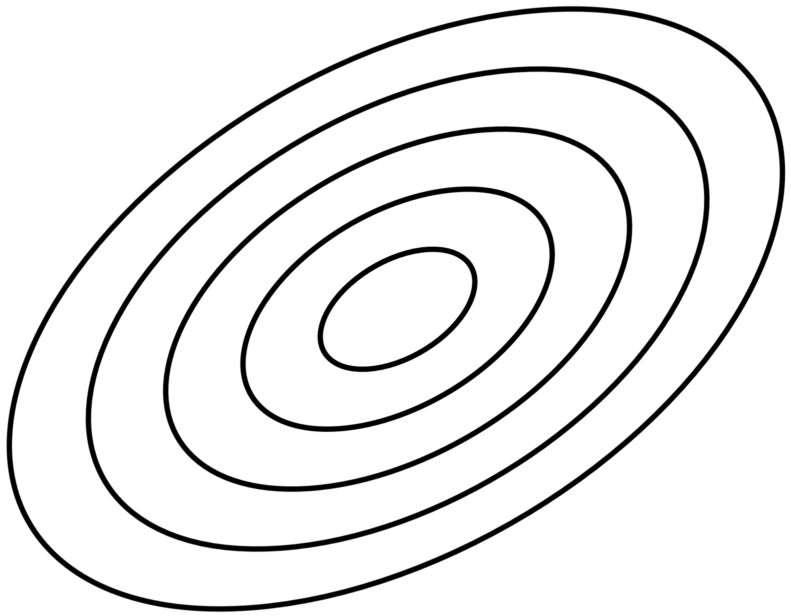
- B = 1 random block
  => sum over b data items instead of n
  => Mini-batch GD

- ~~B = 1 random datapoint~~
  ~~=> Stochastic GD~~
  ~~=> Not suitable for MapReduce~~

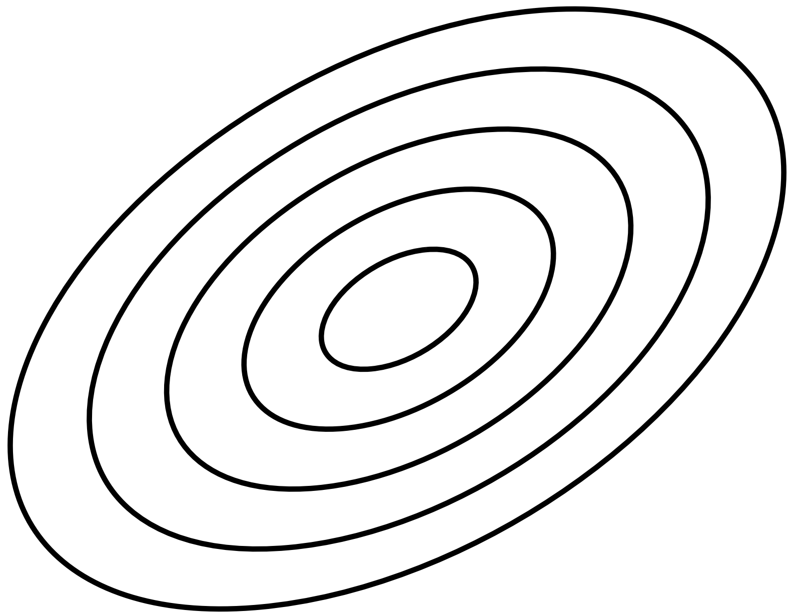
3 key areas:

1. Vary learning rate
2. Update different dimensions of w in parallel
3. Vary B

Effect of #blocks on performance



Batch GD



Mini-batch GD

<u>Gradient descent setup:</u>

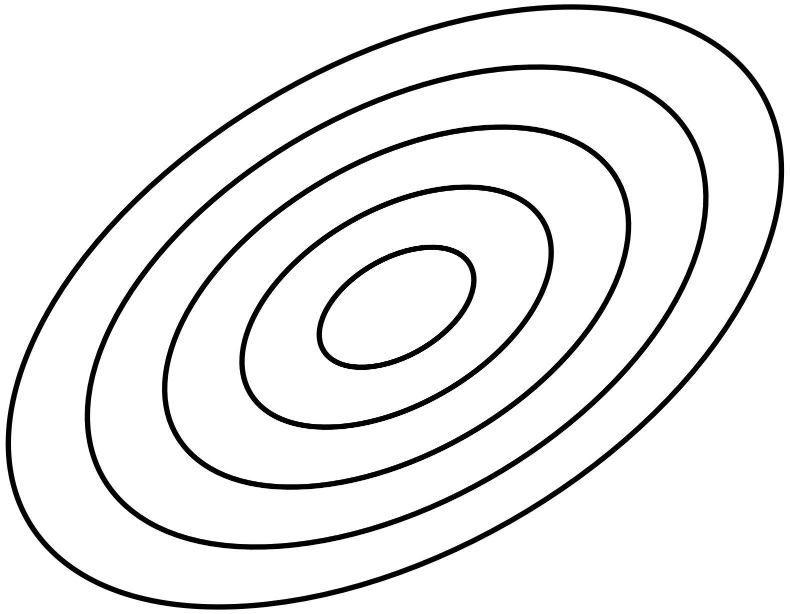
Initialize w_0

while not converged {

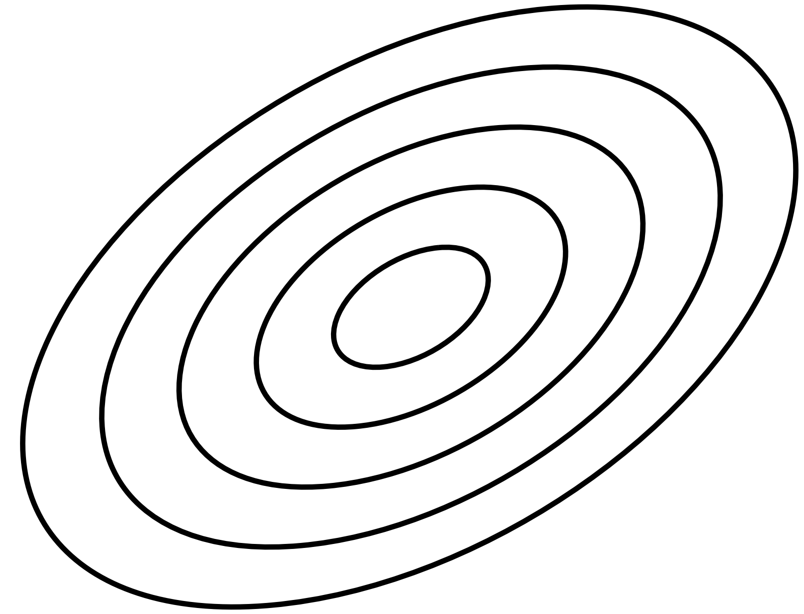
$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

Effect of #blocks on performance



Batch GD



Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

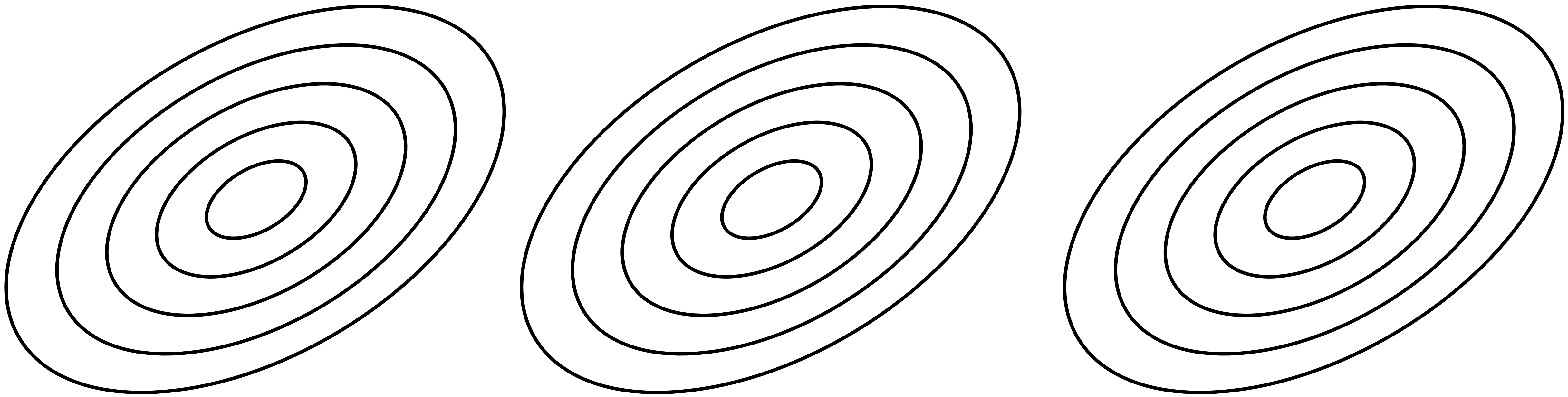
$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

Use entire dataset

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

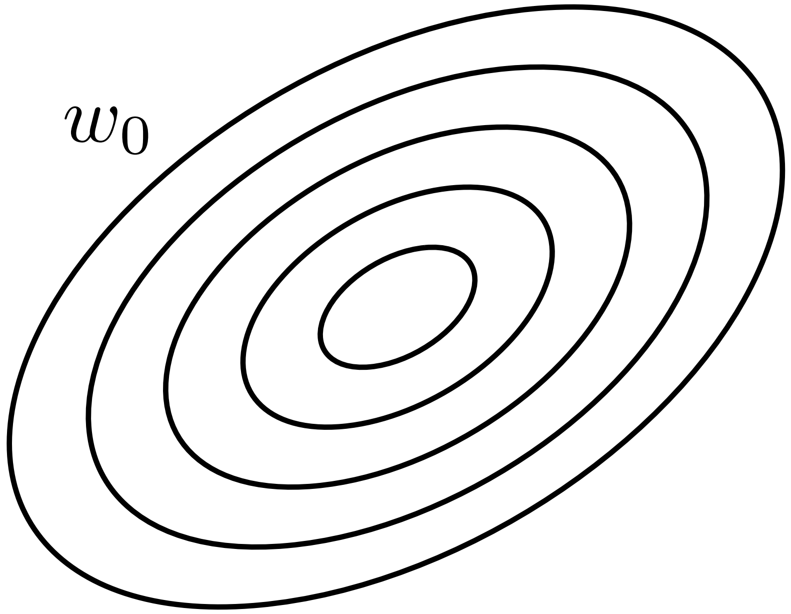
}

Use entire dataset

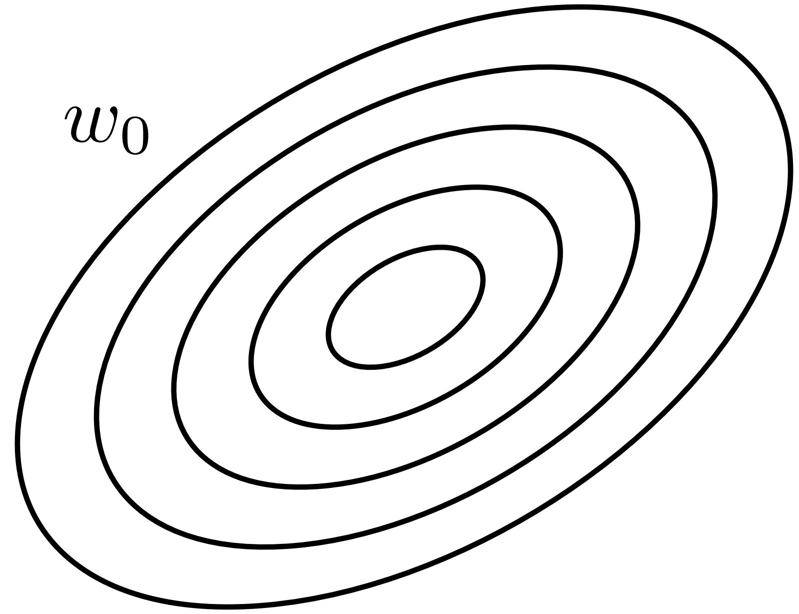
Use k_i blocks

Use one block

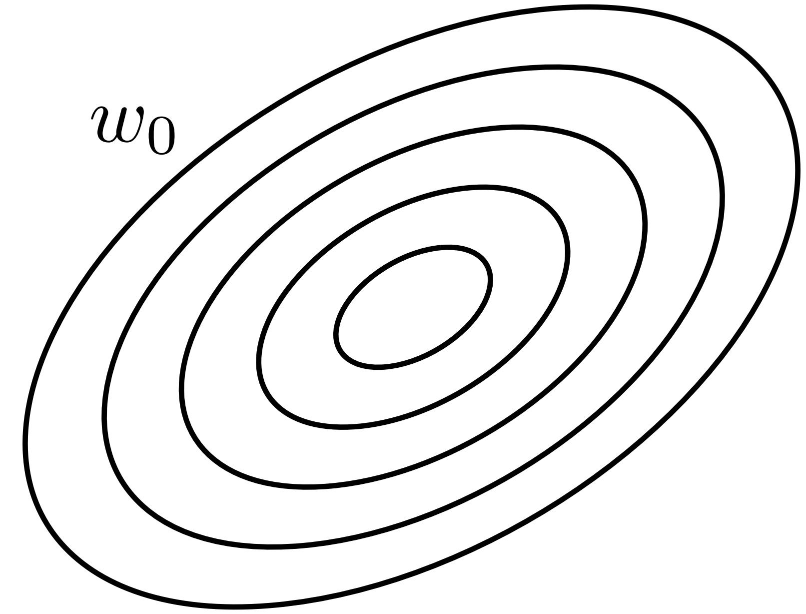
Effect of #blocks on performance



Batch GD



Hybrid approach



Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

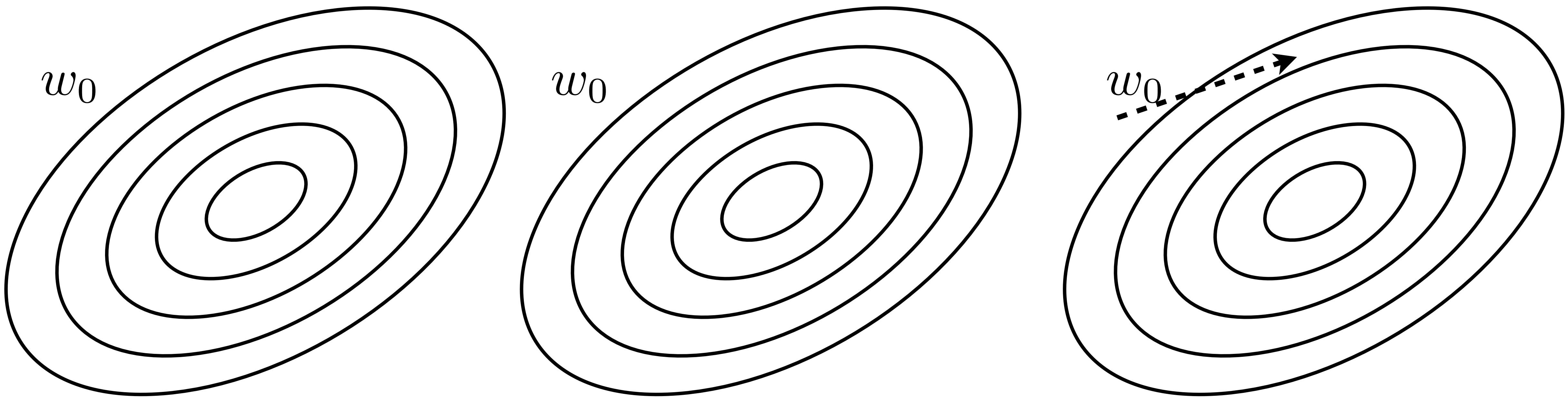
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD

Hybrid approach

Mini-batch GD

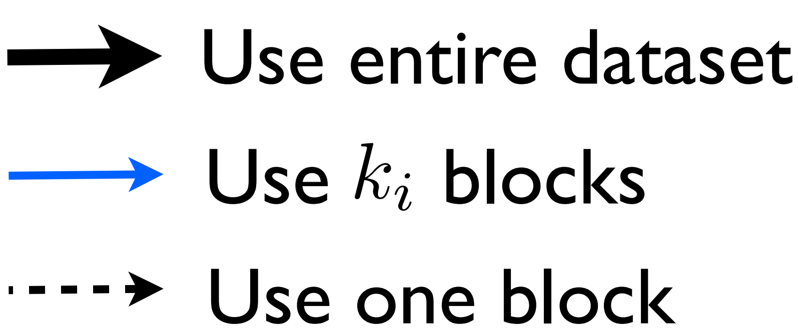
Gradient descent setup:

Initialize w_0

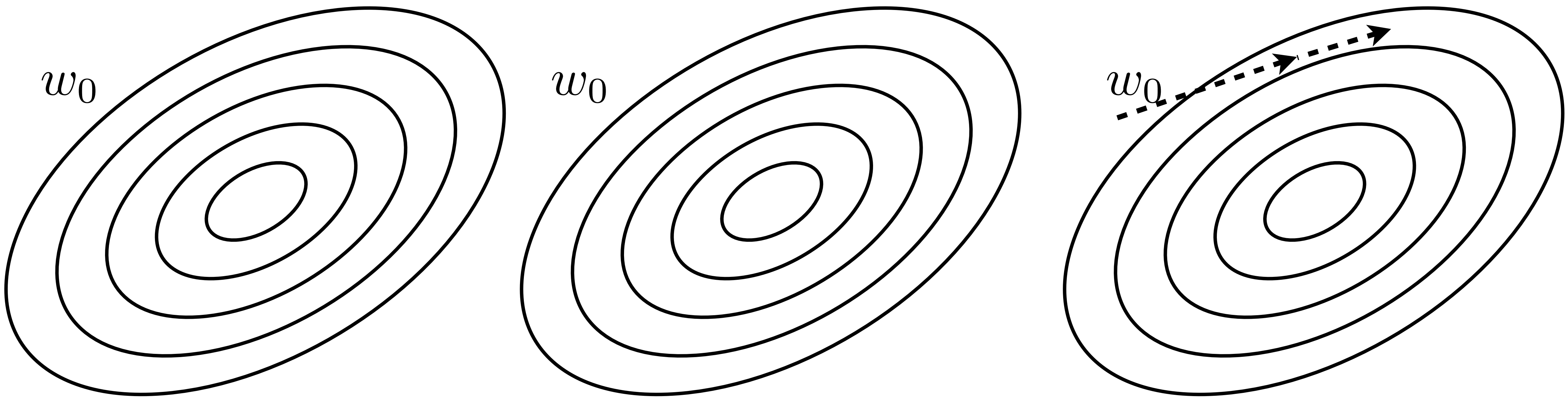
while not converged {

$$w_{i+1} = w_i - \alpha\, \mathbb{E}[\nabla L_{j \in B}]$$

}



Effect of #blocks on performance



Batch GD

Hybrid approach

Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha\, \mathbb{E}[\nabla L_{j \in B}]$$

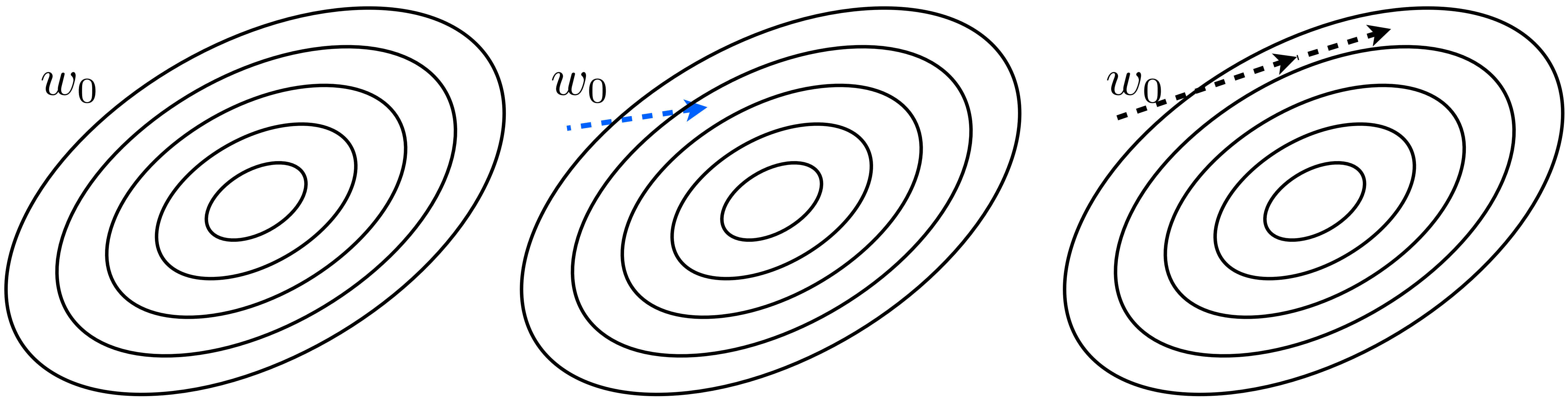
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

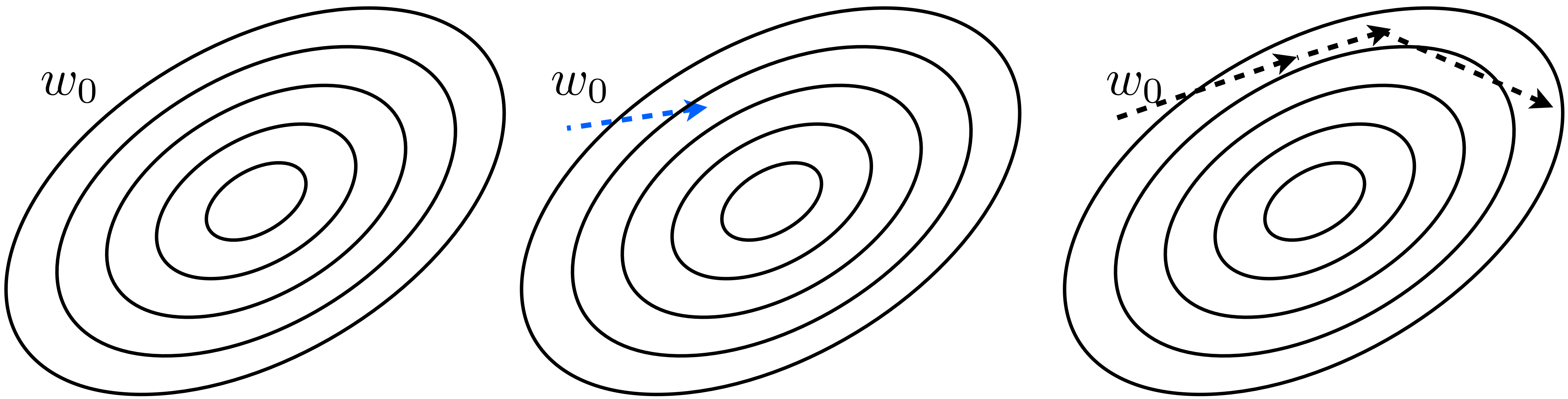
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

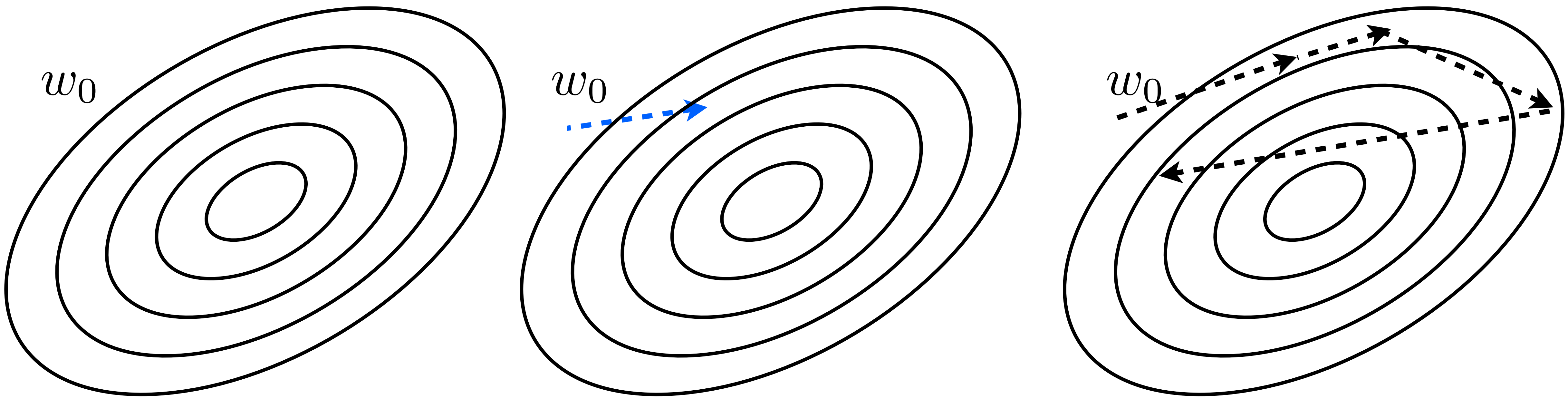
}

→ Use entire dataset

→ Use k_i blocks

⇢ Use one block

Effect of #blocks on performance



Batch GD

Hybrid approach

Mini-batch GD

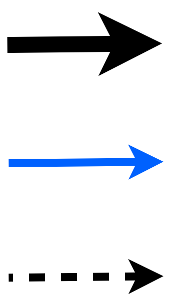
<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha\, \mathbb{E}[\nabla L_{j \in B}]$$

}

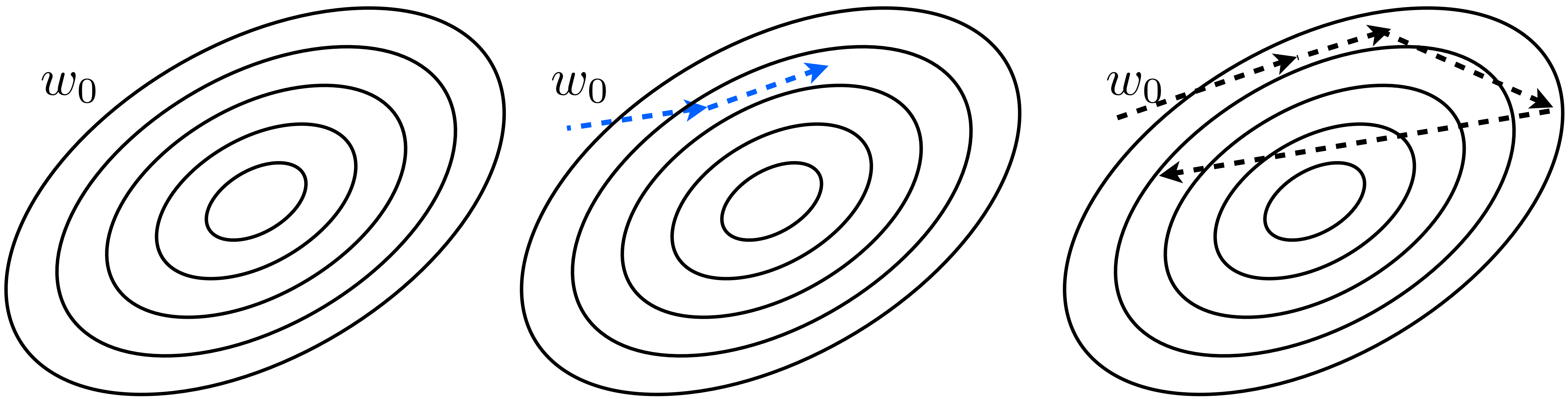


Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

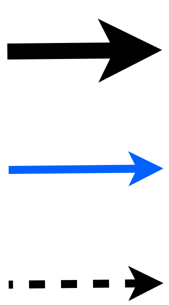
<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

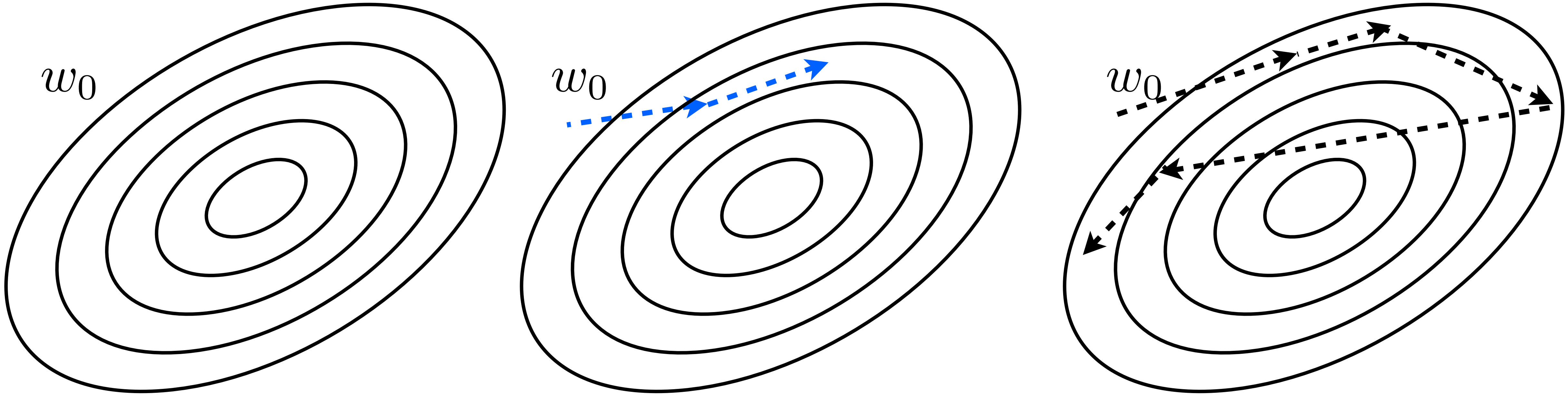


Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

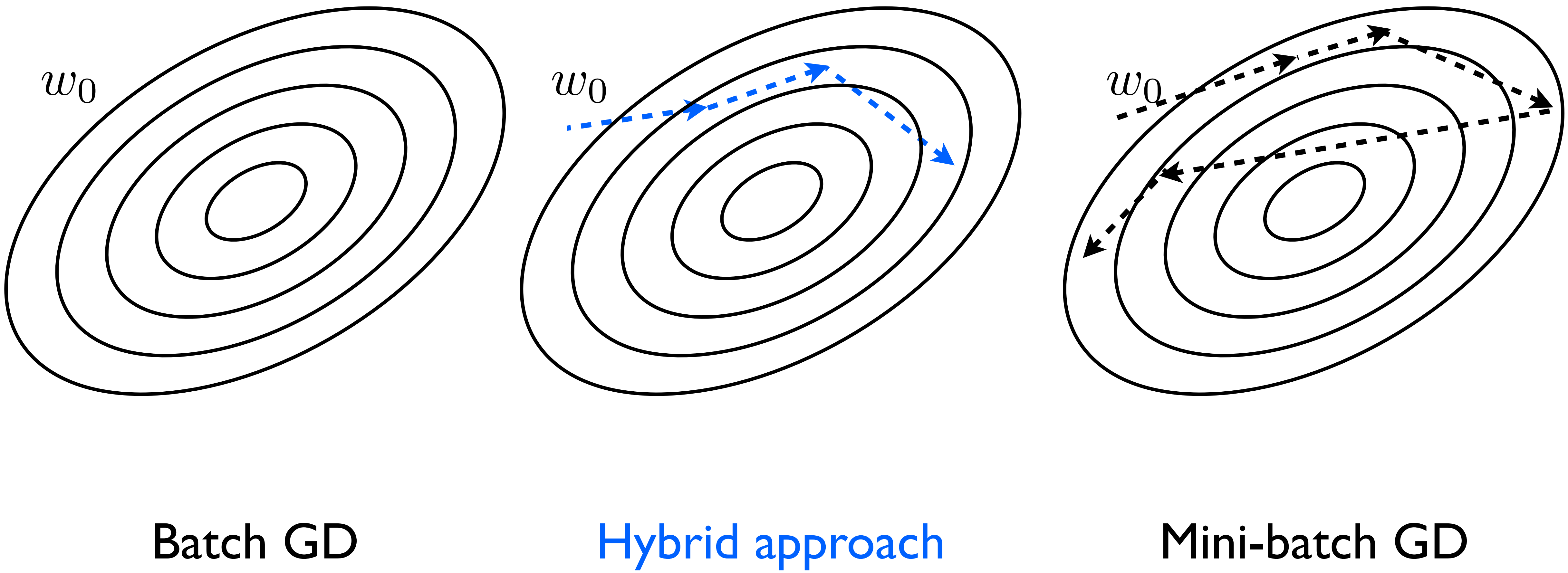
}

→ Use entire dataset

→ Use k_i blocks

⇢ Use one block

Effect of #blocks on performance



Gradient descent setup:

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha\, \mathbb{E}[\nabla L_{j \in B}]$$

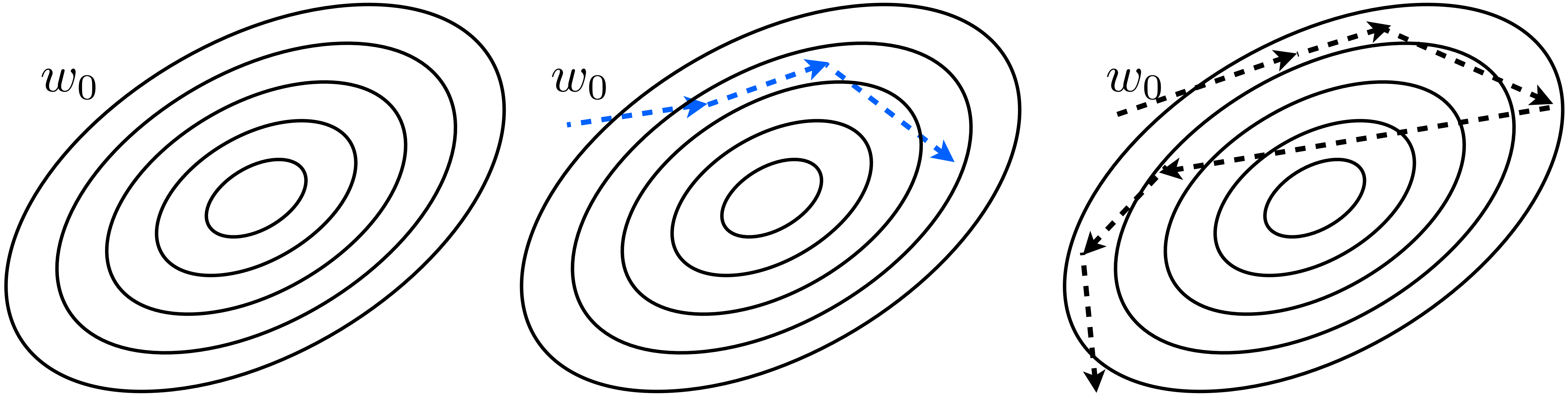
}

→ Use entire dataset

→ Use k_i blocks

⇢ Use one block

Effect of #blocks on performance



Batch GD

Hybrid approach

Mini-batch GD

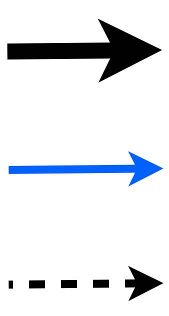
<ins>Gradient descent setup:</ins>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

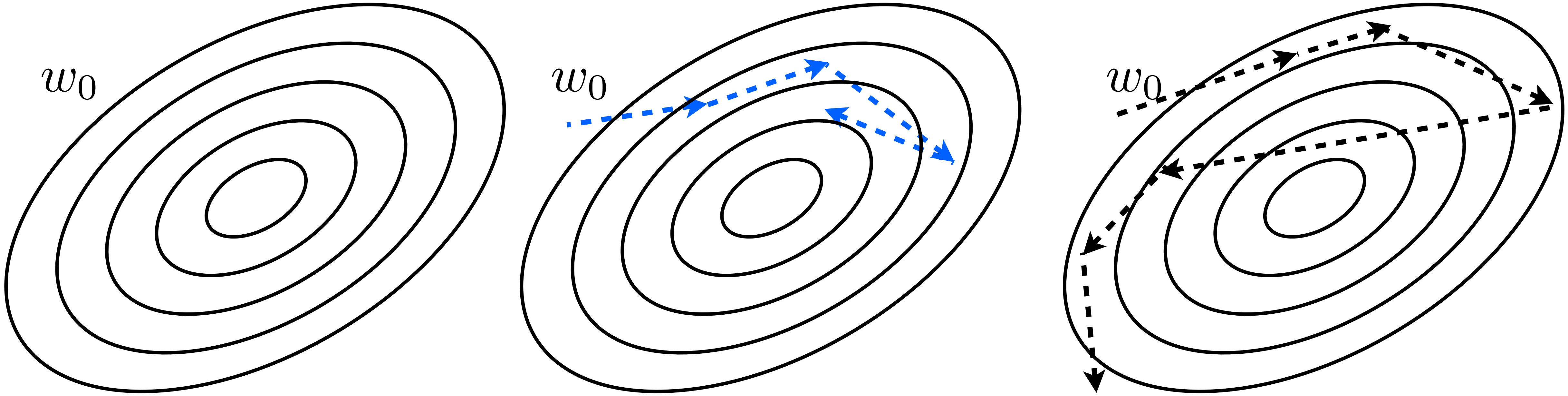


Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha\, \mathbb{E}[\nabla L_{j \in B}]$$

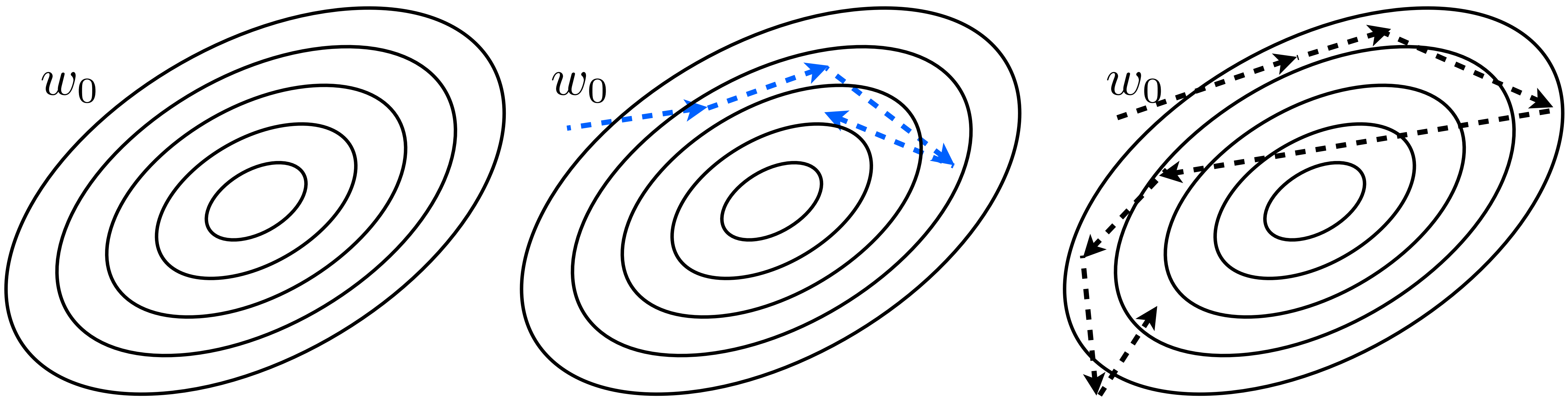
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

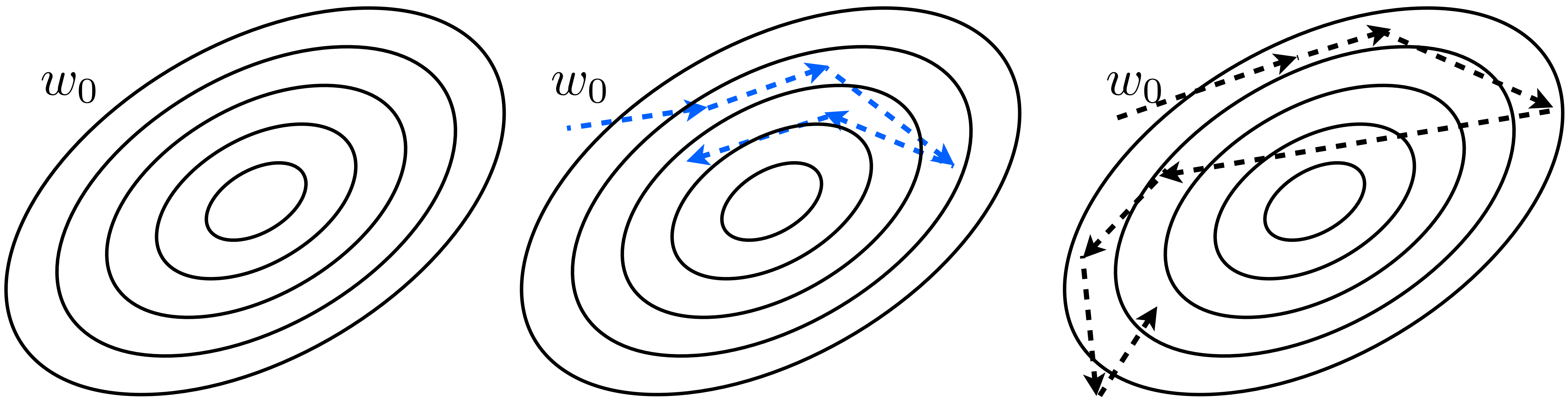
}

→ Use entire dataset

→ Use k_i blocks

⇢ Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

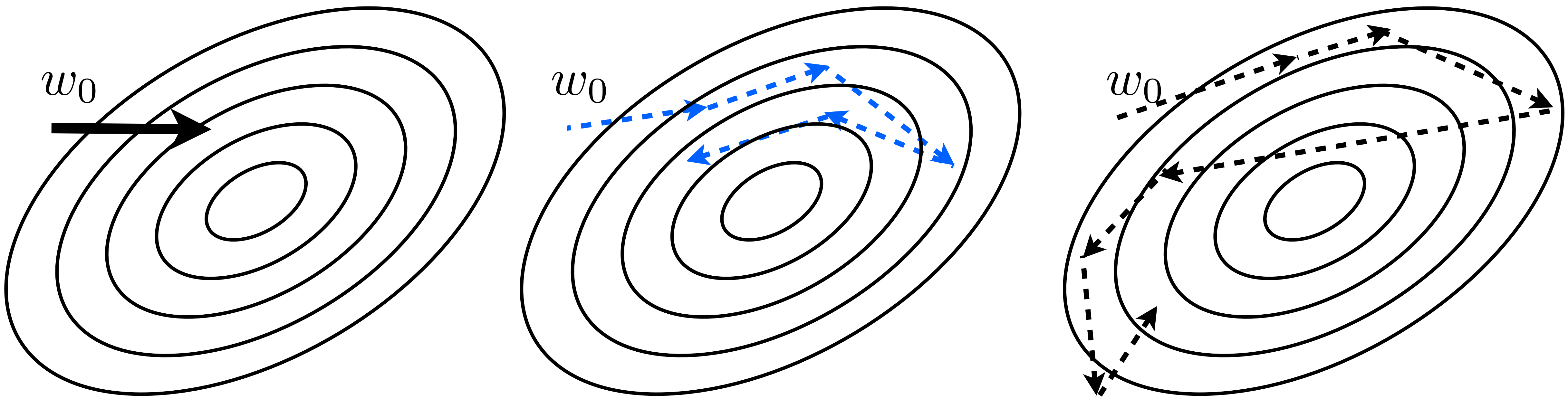
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

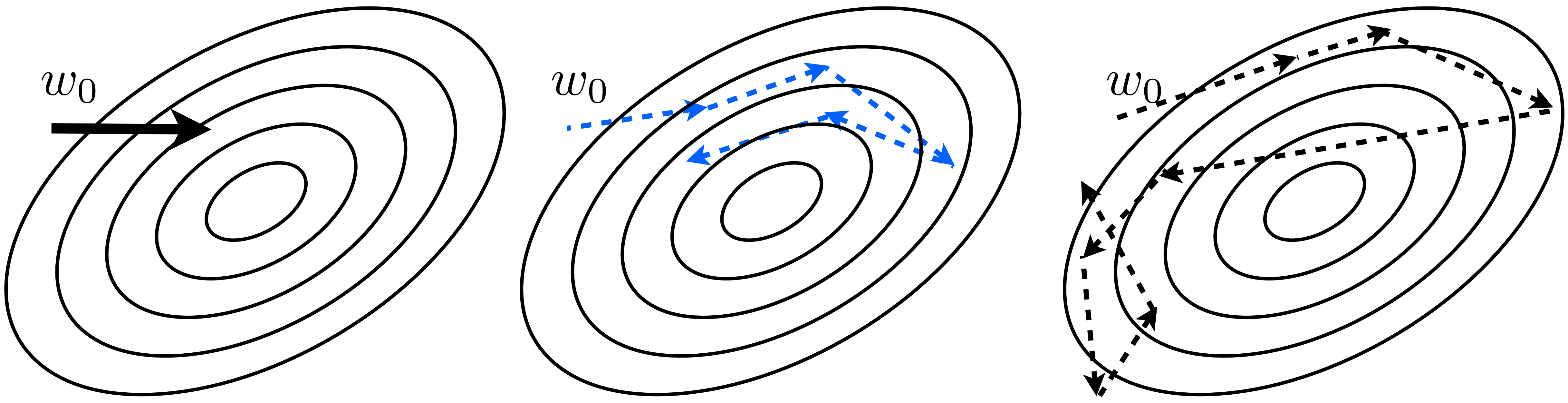
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha\, \mathbb{E}[\nabla L_{j \in B}]$$

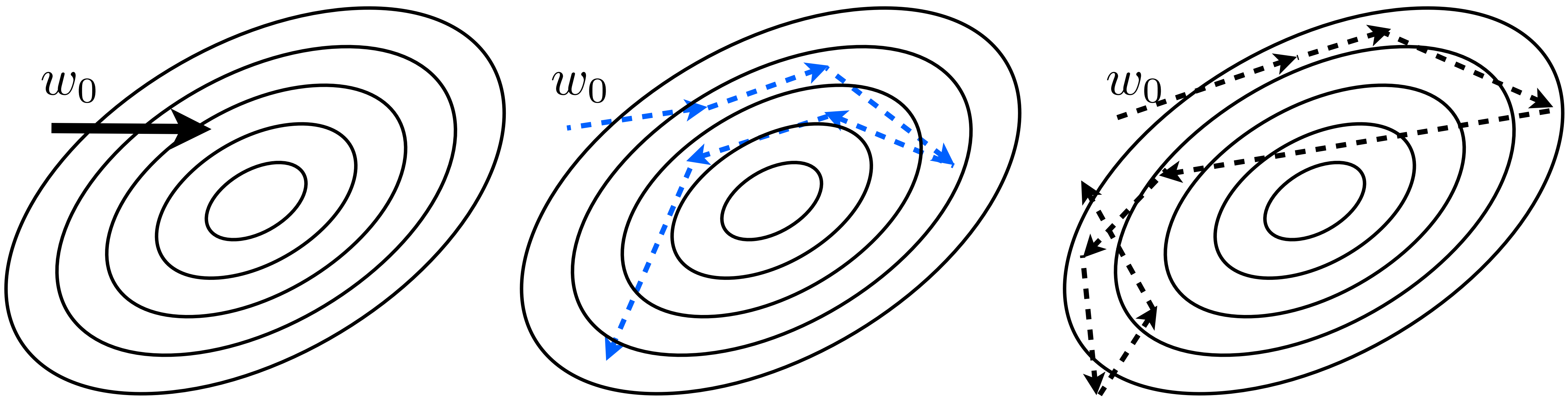
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha\, \mathbb{E}[\nabla L_{j \in B}]$$

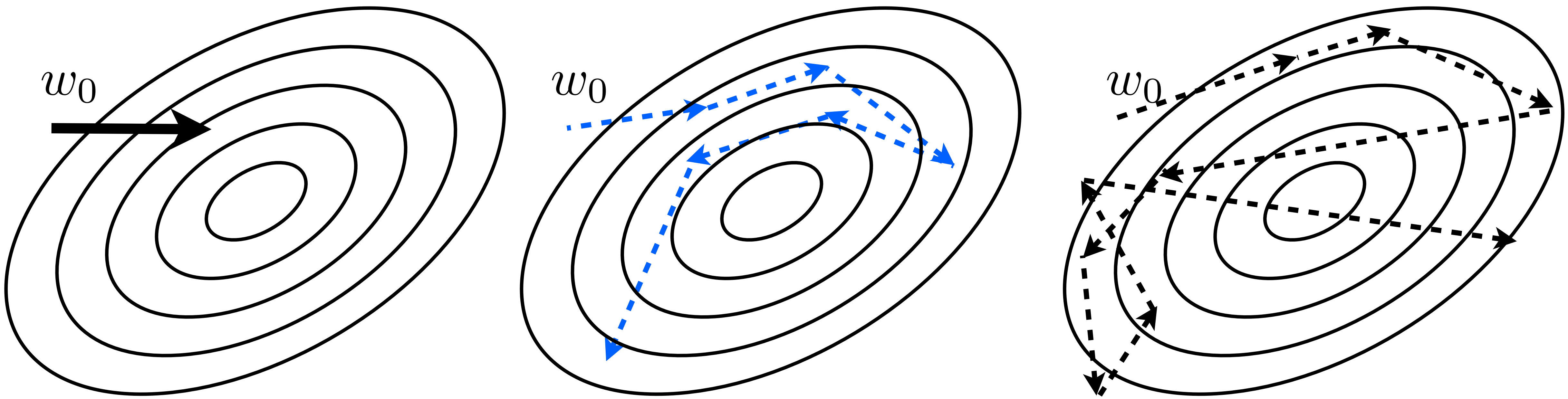
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

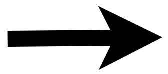
<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

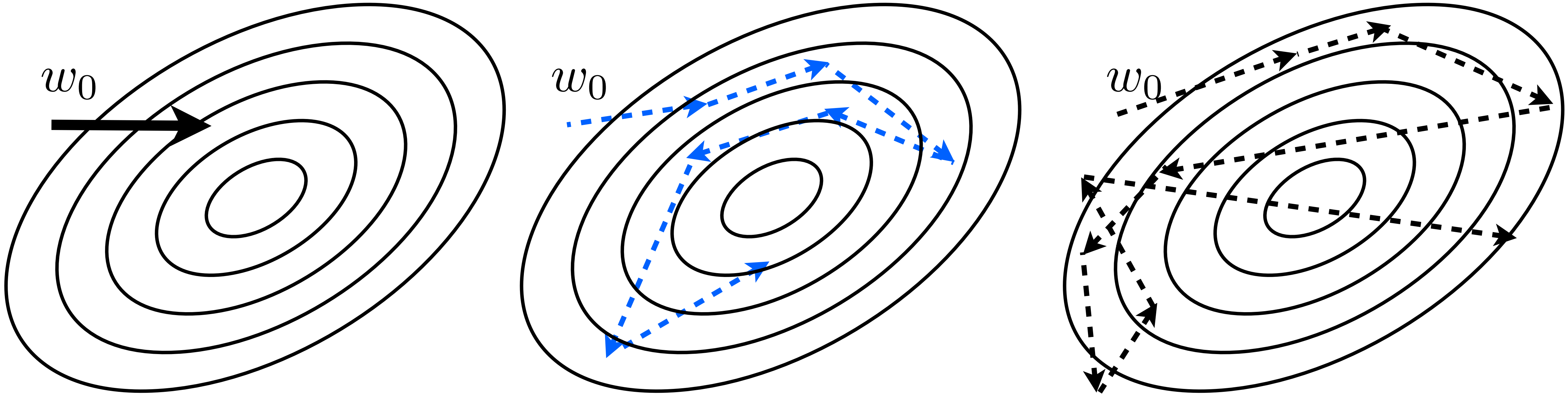
}


Use entire dataset


Use k_i blocks


Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

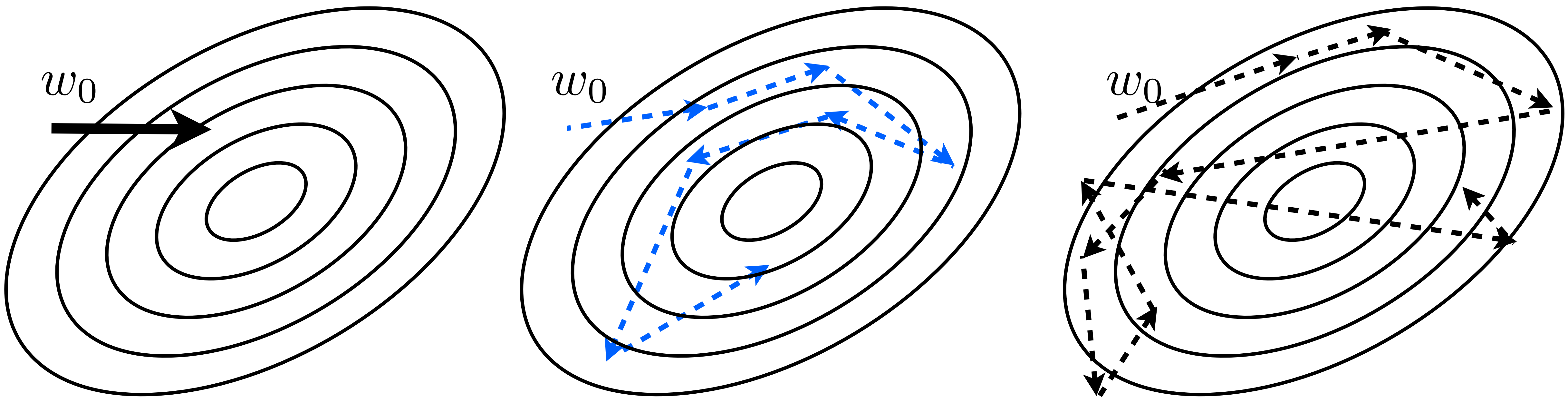
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

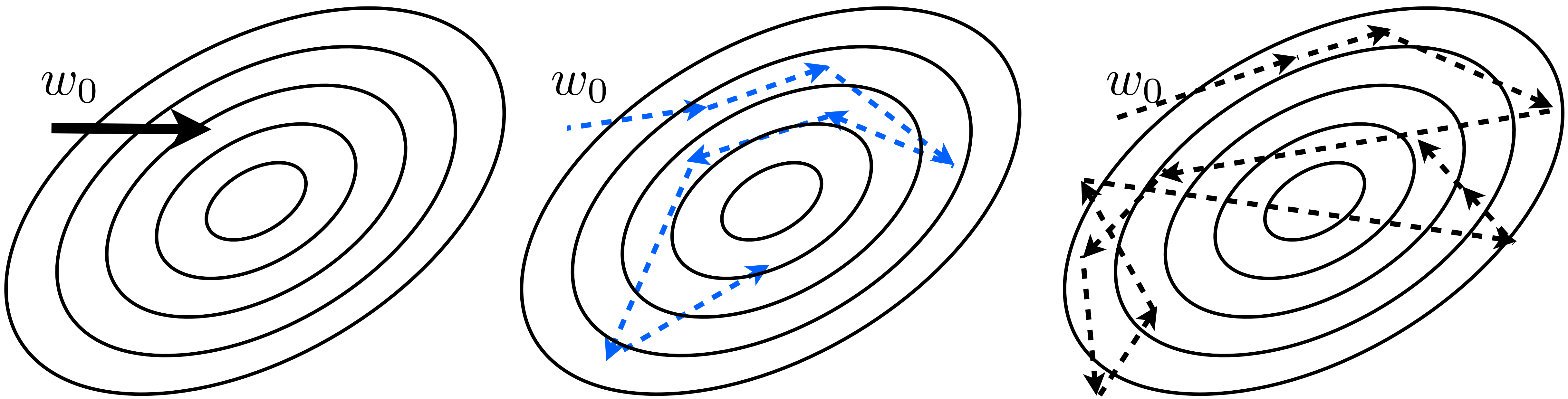
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

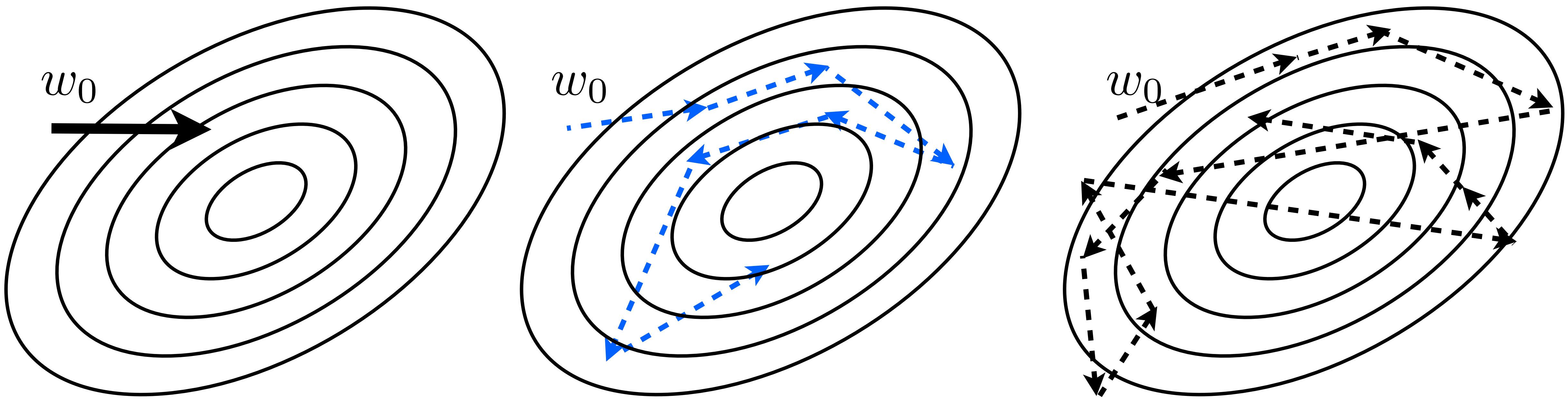
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

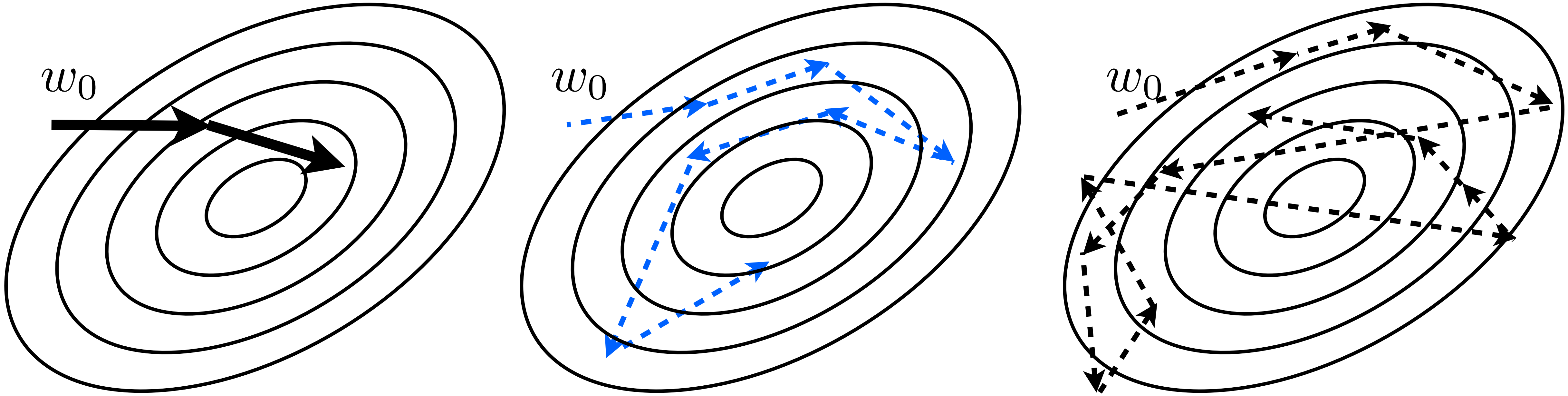
}

→ Use entire dataset

→ Use k_i blocks

⇢ Use one block

Effect of #blocks on performance



Batch GD   Hybrid approach   Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha\,\mathbb{E}[\nabla L_{j \in B}]$$

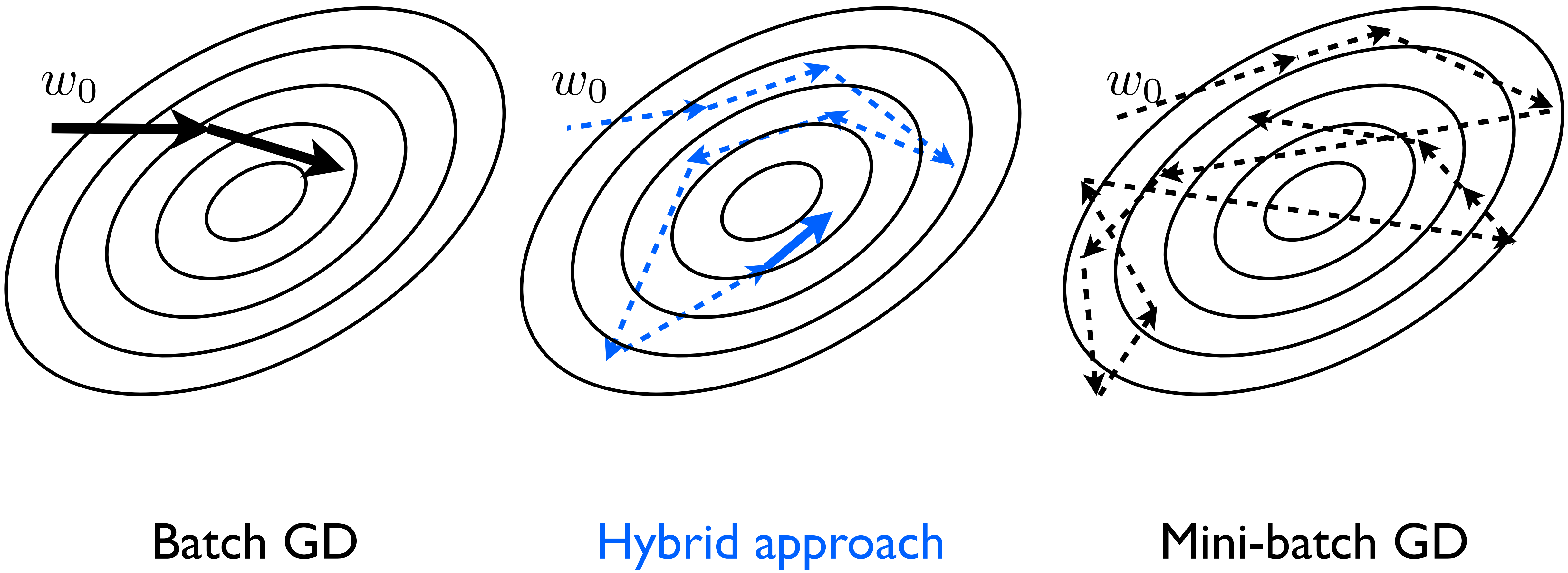
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



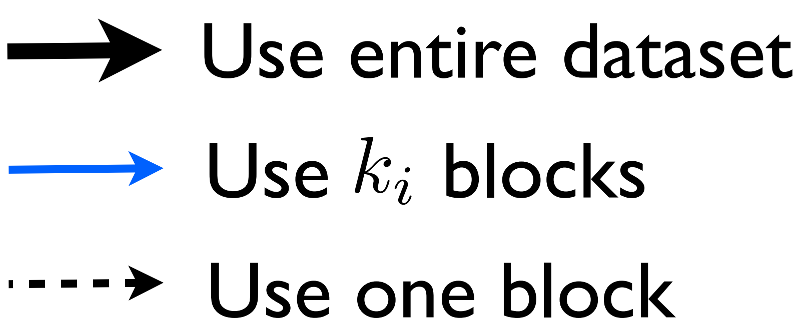
<u>Gradient descent setup:</u>

Initialize w_0

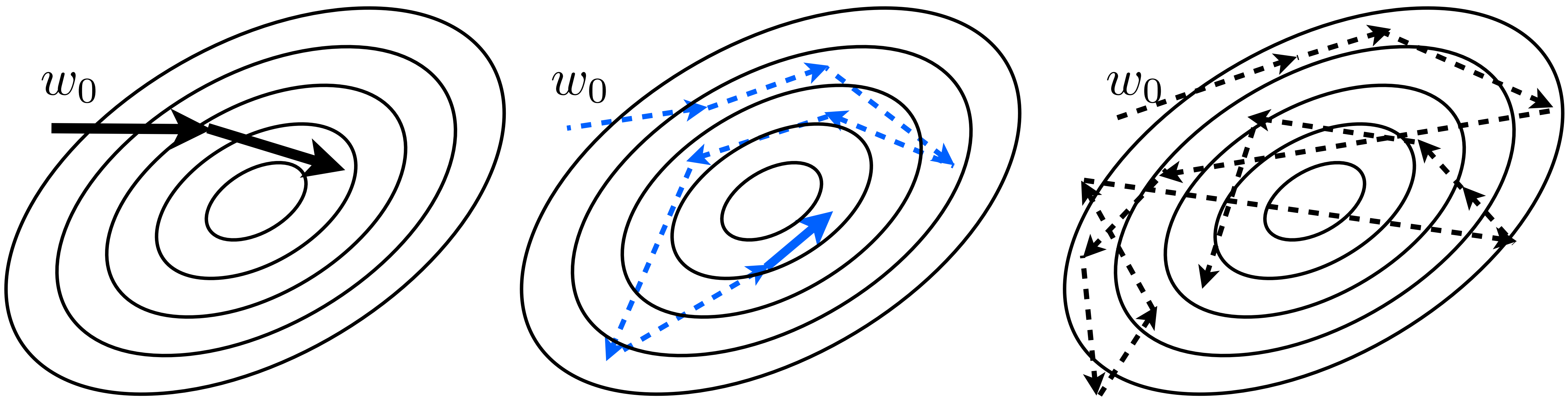
while not converged {

$$w_{i+1} = w_i - \alpha\, \mathbb{E}[\nabla L_{j \in B}]$$

}



Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

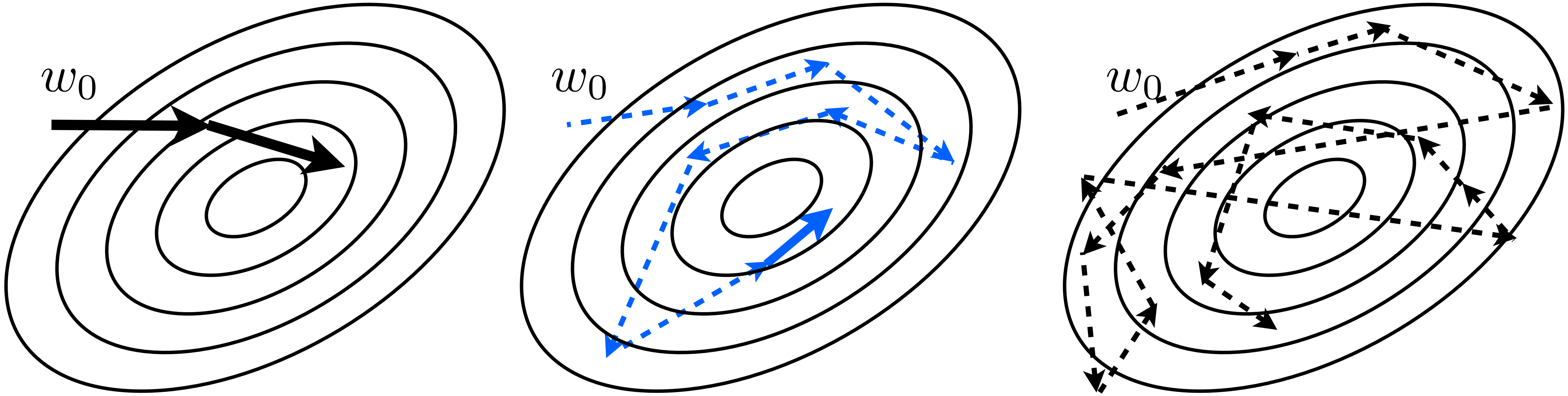
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

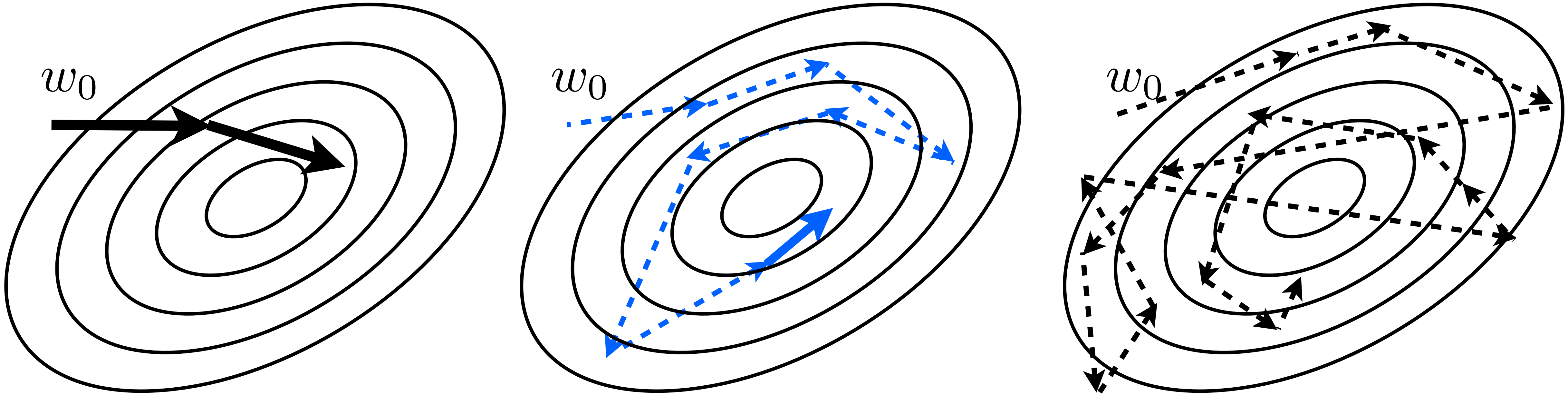
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

<u>Gradient descent setup:</u>

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

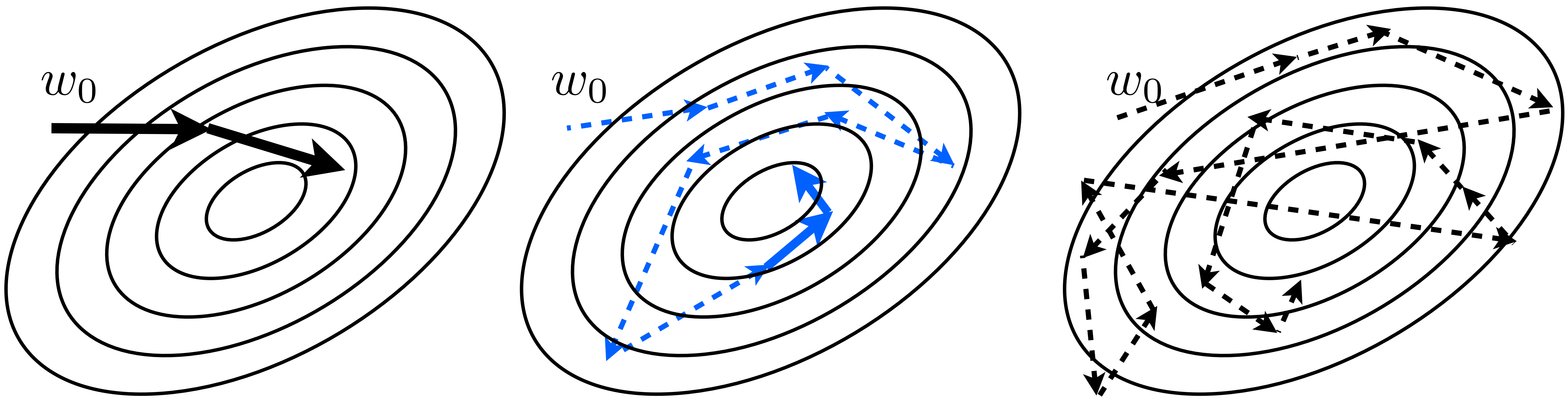
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD | Hybrid approach | Mini-batch GD

Gradient descent setup:

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

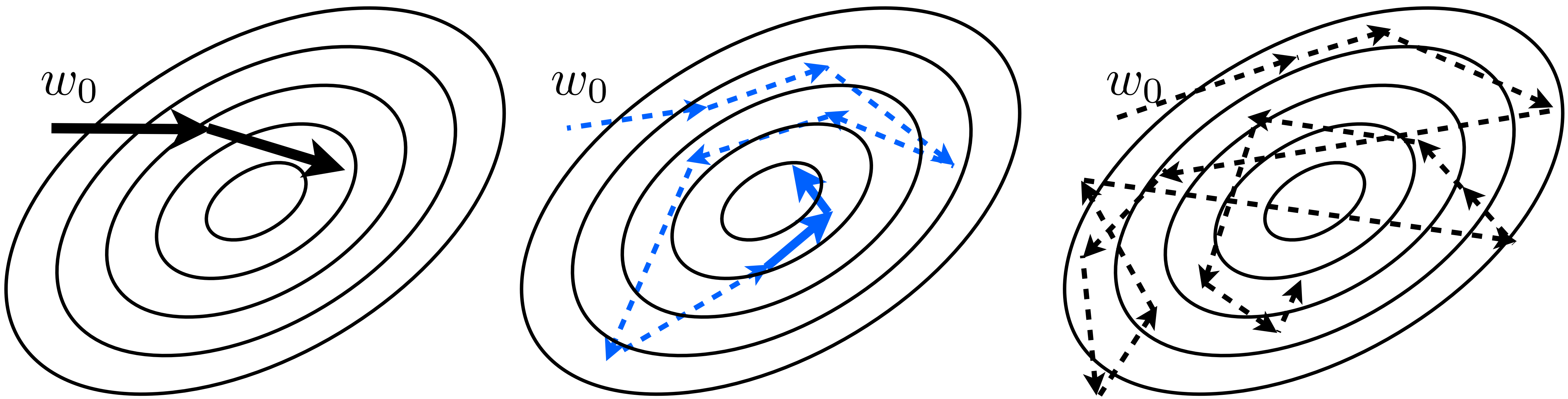
}

Use entire dataset

Use k_i blocks

Use one block

Effect of #blocks on performance



Batch GD

Hybrid approach

My contribution: Model this stopping criteria in principled approach

Gradient descent setup:

Initialize w_0

while not converged {

$$w_{i+1} = w_i - \alpha \, \mathbb{E}[\nabla L_{j \in B}]$$

}

Use entire dataset

Use k_i blocks

Use one block

High-level intuition about "principled approach"

User issues aggregate query $\longrightarrow$ MapReduce

High-level intuition about “principled approach”

User issues aggregate query → MapReduce

Eg: avg(employee salary)

High-level intuition about “principled approach”

User issues aggregate query → MapReduce

Eg: avg(employee salary)

Typical MapReduce pgm returns

10 seconds

2 minutes

1000

10 hours

High-level intuition about "principled approach"

User issues aggregate query → MapReduce

Eg: avg(employee salary)

Typical MapReduce pgm returns

OLA returns

10 seconds

2 minutes

1000

10 hours

High-level intuition about "principled approach"

User issues aggregate query → MapReduce

Eg: avg(employee salary)

Typical MapReduce pgm returns

OLA returns

10 seconds — OLA returns [500, 1600]

2 minutes

10 hours — Typical MapReduce pgm returns 1000

High-level intuition about "principled approach"

User issues aggregate query → MapReduce

Eg: avg(employee salary)

Typical MapReduce
pgm returns

OLA returns

10 seconds: [500, 1600]

2 minutes: [900, 1100]

10 hours: 1000

High-level intuition about "principled approach"

User issues aggregate query → MapReduce

Eg: avg(employee salary)

Typical MapReduce pgm returns

OLA returns

10 seconds [500, 1600]

[900, 1100]

2 minutes [999, 1001]

1000

10 hours

High-level intuition about "principled approach"

User issues aggregate query → MapReduce

Eg: avg(employee salary)

Typical MapReduce pgm returns

OLA returns

10 seconds [500, 1600]

[900, 1100]

2 minutes [999, 1001]

...

10 hours 1000

High-level intuition about "principled approach"

User issues aggregate query → MapReduce

Eg: avg(employee salary)

Typical MapReduce pgm returns

OLA returns

10 seconds [500, 1600]

[900, 1100]

2 minutes [999, 1001]

...

[999.5, 1000.4]

1000 10 hours

High-level intuition about "principled approach"

User issues aggregate query → MapReduce

Eg: avg(employee salary)

Typical MapReduce pgm returns		OLA returns
	10 seconds	[500, 1600]
		[900, 1100]
	2 minutes	[999, 1001]
		...
		[999.5, 1000.4]
		[999.8, 1000.3]
1000	10 hours	

High-level intuition about "principled approach"

User issues aggregate query → MapReduce

Eg: avg(employee salary)

Typical MapReduce pgm returns

OLA returns

10 seconds
[500, 1600]
[900, 1100]

2 minutes
[999, 1001]
...
[999.5, 1000.4]
[999.8, 1000.3]
[999.9, 1000.1]

10 hours

1000

High-level intuition about "principled approach"

User issues aggregate query → MapReduce

Eg: avg(employee salary)

Typical MapReduce pgm returns

OLA returns

10 seconds [500, 1600]

[900, 1100]

2 minutes [999, 1001]

...

[999.5, 1000.4]

[999.8, 1000.3]

[999.9, 1000.1]

...

1000 10 hours

High-level intuition about "principled approach"

User issues aggregate query ⟶ MapReduce

Eg: avg(employee salary)

Typical MapReduce pgm returns			OLA returns
		10 seconds	[500, 1600]
			[900, 1100]
		2 minutes	[999, 1001]
			...
			[999.5, 1000.4]
			[999.8, 1000.3]
			[999.9, 1000.1]
			...
1000	↓	10 hours	[1000, 1000]

High-level intuition about "principled approach"

User issues aggregate query → MapReduce

Eg: avg(employee salary)

Typical MapReduce pgm returns		OLA returns
	10 seconds	[500, 1600]
		[900, 1100]
	2 minutes	[999, 1001]
		...
		[999.5, 1000.4]
		[999.8, 1000.3]
		[999.9, 1000.1]
		...
1000	10 hours	[1000, 1000]

Online

High-level intuition about "principled approach"

User issues $\boxed{\text{aggregate}}$ query ⟶ MapReduce

Eg: avg(employee salary)

Typical MapReduce pgm returns			OLA returns	
	↓	10 seconds	[500, 1600]	Online Aggregation
			[900, 1100]	
		2 minutes	[999, 1001]	
			...	
			[999.5, 1000.4]	
			[999.8, 1000.3]	
			[999.9, 1000.1]	
			...	
1000		10 hours	[1000, 1000]	

High-level intuition about "principled approach"

User issues $\boxed{\text{aggregate}}$ query → MapReduce

Eg: avg(employee salary)

My previous work

Typical MapReduce pgm returns

OLA returns

Time	Typical MapReduce pgm returns	OLA returns
10 seconds		[500, 1600]
		[900, 1100]
2 minutes		[999, 1001]
		...
		[999.5, 1000.4]
		[999.8, 1000.3]
		[999.9, 1000.1]
		...
10 hours	1000	[1000, 1000]

Online Aggregation

High-level intuition about "principled approach"

User issues $\boxed{\text{aggregate}}$ query → MapReduce

Eg: avg(employee salary)

My previous work

Online Aggregation

Typical MapReduce pgm returns		OLA returns
	10 seconds	[500, 1600]
		[900, 1100]
	2 minutes	[999, 1001]
		...
		[999.5, 1000.4]
		[999.8, 1000.3]
		[999.9, 1000.1]
		...
1000	10 hours	[1000, 1000]

Idea: If acceptable accuracy reached early, query can be stopped

High-level intuition about "principled approach"

User issues $\boxed{\text{aggregate}}$ query → MapReduce

Eg: avg(employee salary)

My previous work

Typical MapReduce pgm returns

OLA returns

10 seconds [500, 1600]

[900, 1100]

2 minutes [999, 1001]

...

[999.5, 1000.4]

[999.8, 1000.3]

[999.9, 1000.1]

...

1000 10 hours [1000, 1000]

Online Aggregation

Idea: If acceptable accuracy reached early, query can be stopped

$$\theta_{j+1} = \theta_j - \alpha \boxed{\sum_{i=0}^{b} \nabla f_i}$$

Summary: Scalable & Fast Machine Learning

- Work in Progress
- Use OLA to implement GD on MapReduce

  => Improve performance of GD

  => Improve performance of Machine Learning
- Scalability using MapReduce

Modeling challenges:

- Modeling surface
- Modeling random walk

Vary B ... little more detail

Vary B ... little more detail

Batch GD

Initialize θ_0

While not converged $\{$

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{n} \nabla f_i$$

$\}$

Entire dataset

Mini-Batch GD

Initialize θ_0

While not converged $\{$

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{b} \nabla f_i$$

$\}$

Random block b

$b \ll n$

Stochastic GD

Initialize θ_0

While not converged $\{$

$$\theta_{j+1} = \theta_j - \alpha \nabla f_r$$

$\}$

Random datapoint r

* Ignoring divisor & decreasing learning rate for readability
Also, I will be using the variable "f" to represent loss.

Vary B ... little more detail

Batch GD

Initialize θ_0

While not converged {

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{n} \nabla f_i$$

}

Entire dataset

Mini-Batch GD

Initialize θ_0

While not converged {

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{b} \nabla f_i$$

}

Random block b

$b \ll n$

Stochastic GD

Initialize θ_0

While not converged {

$$\theta_{j+1} = \theta_j - \alpha \nabla f_r$$

}

Random datapoint r

Not suitable for MapReduce

* Ignoring divisor & decreasing learning rate for readability
Also, I will be using the variable "f" to represent loss.

Vary B ... little more detail

Batch GD

Initialize θ_0

While not converged {

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{n} \nabla f_i$$

}

Entire dataset

Mini-Batch GD

Initialize θ_0

While not converged {

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{b} \nabla f_i$$

}

Random block b

$$b \ll n$$

Stochastic GD

Initialize θ_0

While not converged {

$$\theta_{j+1} = \theta_j - \alpha \nabla f_r$$

}

Random datapoint r

* Ignoring divisor & decreasing learning rate for readability
Also, I will be using the variable "f" to represent loss.

Vary B ... little more detail

Batch GD

Initialize θ_0

While not converged $\{$

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{n} \nabla f_i$$

$\}$

Entire dataset

Mini-Batch GD

Initialize θ_0

While not converged $\{$

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{b} \nabla f_i$$

$\}$

Random block b

$b \ll n$

* Ignoring divisor & decreasing learning rate for readability
Also, I will be using the variable "f" to represent loss.

Vary B ... little more detail

Batch GD

Initialize θ_0

While not converged $\{$

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{n} \nabla f_i$$

$\}$

Entire dataset

Mini-Batch GD

Initialize θ_0

While not converged $\{$

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{b} \nabla f_i$$

$\}$

Random block b

$b \ll n$

* Ignoring divisor & decreasing learning rate for readability
Also, I will be using the variable "f" to represent loss.

Vary B ... little more detail

Batch GD

Initialize θ_0

While not converged $\{$

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{n} \nabla f_i$$

$\}$

Entire dataset

$b \ll n$

Mini-Batch GD

Initialize θ_0

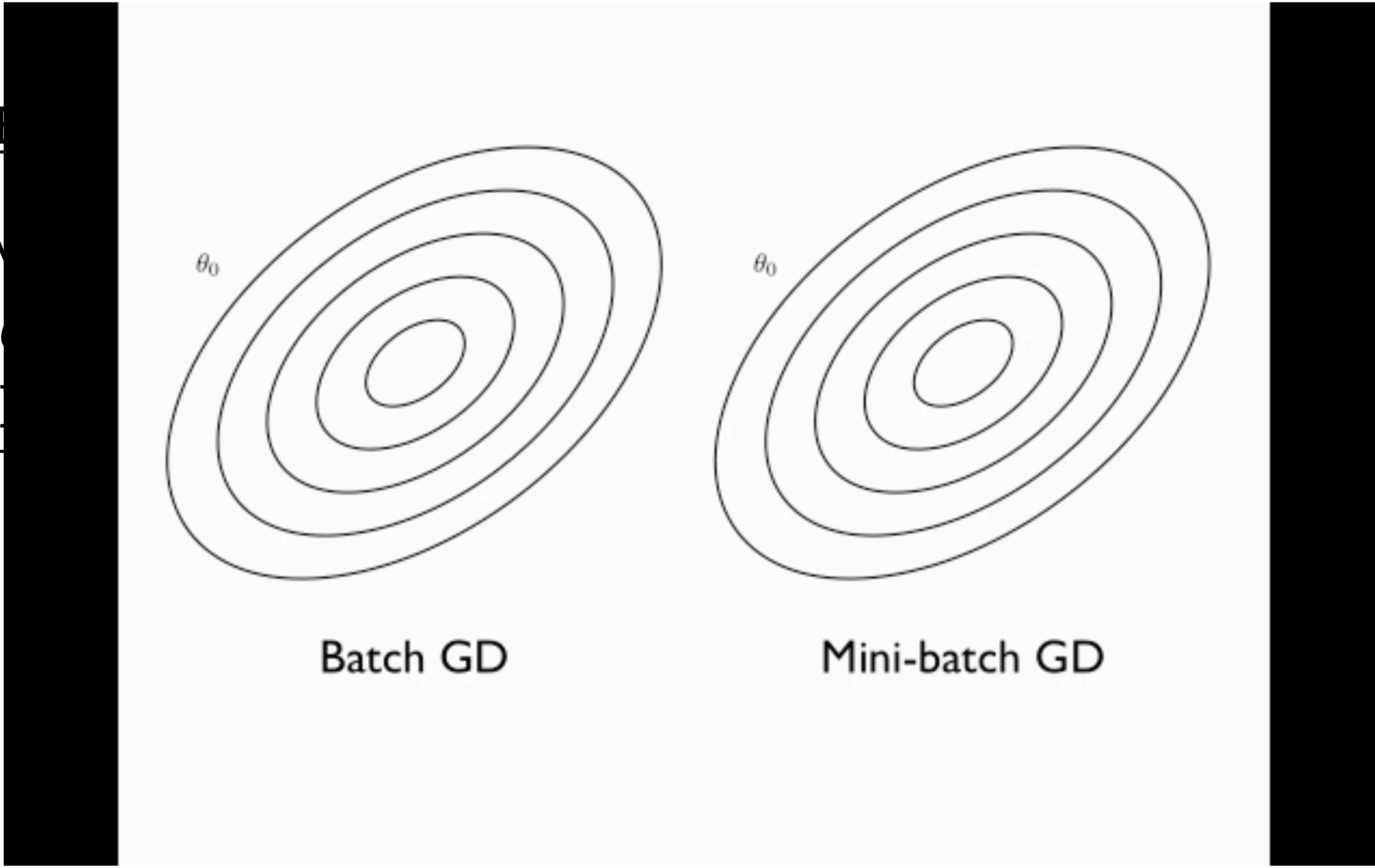
While not converged $\{$

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{b} \nabla f_i$$

$\}$

Random block

Vary B ... little more detail



ed {

∇f_i

Vary B ... little more detail

Batch GD

Initialize θ_0

While not converged $\{$

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{n} \nabla f_i$$

$\}$

Entire dataset

$b \ll n$

Mini-Batch GD

Initialize θ_0

While not converged $\{$

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{b} \nabla f_i$$

$\}$

Random block

Vary B ... little more detail

$b \ll n$

Batch GD

Initialize θ_0

While not converged {

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{n} \nabla f_i$$

}

Entire dataset

- More accurate "f"
- Less iterations of while loops
- Each iteration takes long time

Mini-Batch GD

Initialize θ_0

While not converged {

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{b} \nabla f_i$$

}

Random block

- Less accurate "f"
- More iterations of while loops
- Each iteration takes much less time

Vary B ... little more detail

$b \ll n$

Batch GD

Initialize θ_0
While not converged {
$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{n} \nabla f_i$$
}

Entire dataset

- More accurate "f"
- Less iterations of while loops
- Each iteration takes long time

Mini-Batch GD

Initialize θ_0
While not converged {
$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{b} \nabla f_i$$
}

Random block

- Less accurate "f"
- More iterations of while loops
- Each iteration takes much less time

Key idea: Proceed to next iteration if at least "k" datapoints processed

Vary B ... little more detail

$b \ll n$

Batch GD

Initialize θ_0

While not converged {

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{n} \nabla f_i$$

}

Entire dataset

- More accurate "f"
- Less iterations of while loops
- Each iteration takes long time

Mini-Batch GD

Initialize θ_0

While not converged {

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=0}^{b} \nabla f_i$$

}

Random block

- Less accurate "f"
- More iterations of while loops
- Each iteration takes much less time

Key idea: Proceed to next iteration if at least "k" datapoints processed where k is found using a bayesian model (Stopping criteria)

References

- http://www.eetimes.com/document.asp?doc_id=1273834
- http://www.youtube.com/