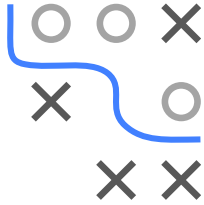
Optimization in Machine Learning

First order methods
Comparison of first order methods

Learning goals

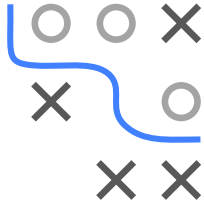
- Gradient Descent
- Stochastic Gradient Descent
- Momentum
- Step size decay

COMPARISON OF FIRST ORDER METHODS

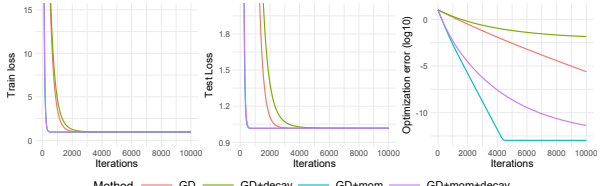


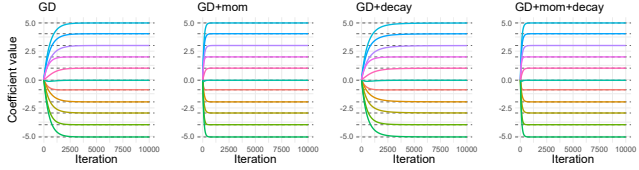
- Comparison of (S)GD, (S)GD + momentum, and (S)GD + momentum + step size control on simulated data
- We do not use analytical solution on purpose although one exists:
- Linear regression (squared loss) simulation $\boldsymbol{y} = \mathbf{X}\boldsymbol{\theta}^* + \boldsymbol{\epsilon}$ with $n = 500$ samples and $p = 11$ features, where $\boldsymbol{\theta}^* = (-5, -4, \ldots 0, \ldots, 4, 5)^\top$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \Sigma)$ for $\Sigma = \boldsymbol{I}$ (i.i.d. features) or $\Sigma_{i,j} = 0.99^{|i-j|}$ (corr. features)
- Indep. features result in a condition number of ≈ 2.9, whereas the corr. feature set-up produces a (moderately) bad condition number of ≈ 600
- We set the momentum parameter to 0.8 and the decay step size using schedule $\alpha^{[t]} = \alpha^{[0]} \cdot \texttt{decay}^{t/t_{\text{max}}}$ for $\texttt{decay} = 0.1$
- For GD and SGD we use different step sizes to show that benefit of momentum/decay depends heavily on step size
- ERM has unique global minimizer given by $\hat{\boldsymbol{\theta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \boldsymbol{y}$
- We also track the optimization error $\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|_2$

LIN-REG (GD + MED STEP SIZE, DEFAULT CASE)



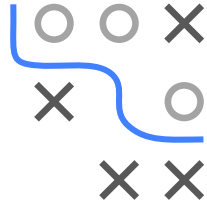
- GD with medium $\alpha = 2 \cdot 10^{-3}$ and indep. features:



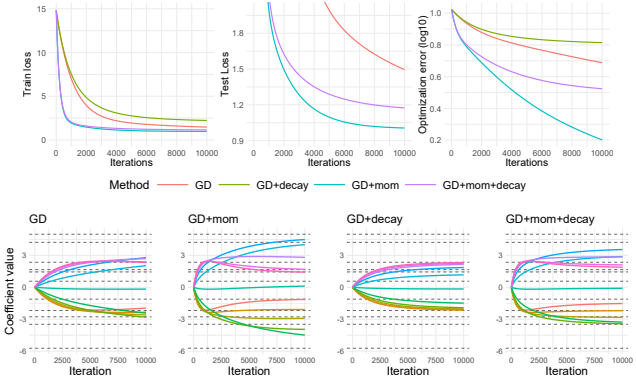


- Irreducible error due to additive noise is $\sigma = 1$. Dotted lines indicate global minimizers
- All variants converge to global min. **Momentum** accelerates optimization and **decay** slows down optimization under that step size

LIN-REG (GD + CORR. FEATURES)

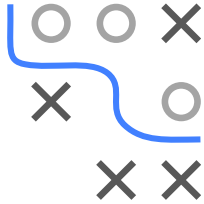


- GD with medium $\alpha = 2 \cdot 10^{-3}$ and bad conditioning (corr. features):

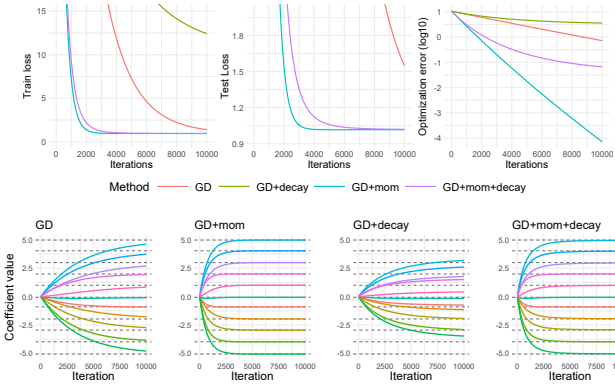


- Irreducible error due to additive noise is $\sigma = 1$. Dotted lines indicate global minimizers
- Moderately bad **conditioning slows down optim** severely! Only **momentum** w/o decay comes close to global min
- **Momentum** causes "overshooting" of some coefs. Corr. features cause corr. global minimizers

LIN-REG (GD + SMALL STEP SIZE)

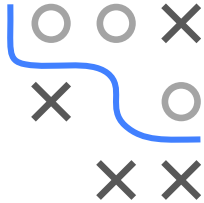


- GD with (too small) $\alpha = 3 \cdot 10^{-4}$ and indep. features:

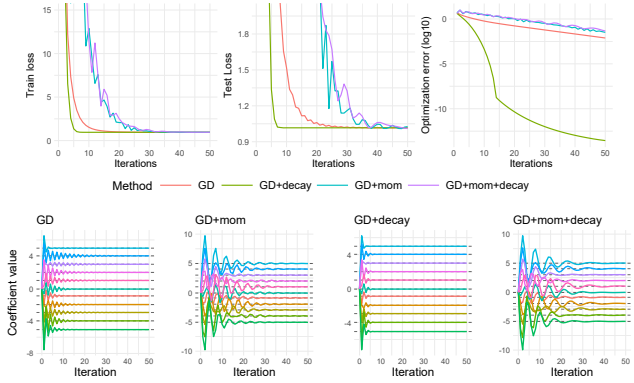


- Irreducible error due to additive noise is $\sigma = 1$. Dotted lines indicate global minimizers
- Only two **momentum** variants come close to global min in $t_{\text{max}} = 10000$
- **Decay** worsens performance as α was already too low

LIN-REG (GD + LARGE STEP SIZE)



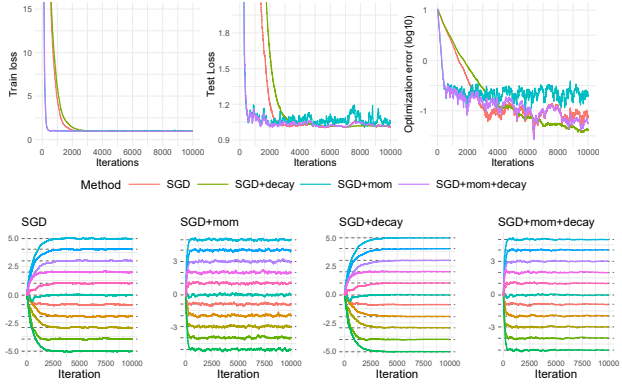
- GD with large $\alpha = 1.5$ and indep. features:

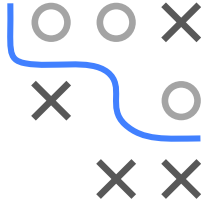


- Irreducible error due to additive noise is $\sigma = 1$. Dotted lines indicate global minimizers
- Super fast convergence in < 20 steps
- **Decay** here accelerates optim while **momentum** becomes slow and unstable
- Coefficients oscillate at beginning which is reduced by decay

LIN-REG (SGD + MED STEP SIZE)

- SGD with medium $\alpha = 2 \cdot 10^{-3}$ and indep. features:

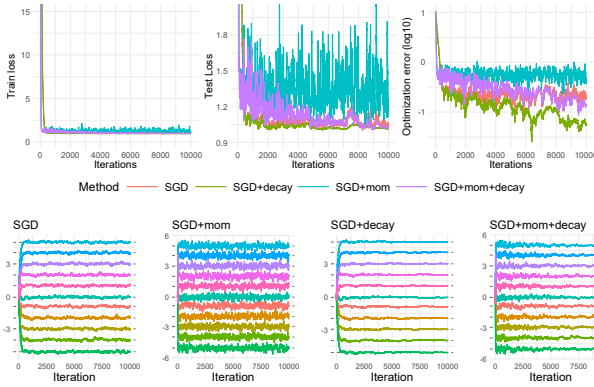




- **Momentum** accelerates optim initially but is eventually outperformed by other variants
- **Momentum+decay** is both fast initially and has small final error
- **Decay** performs best overall but is slowest initially

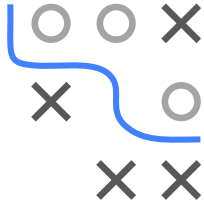
LIN-REG (SGD + LARGE STEP SIZE)

- SGD with large $\alpha = 1 \cdot 10^{-2}$ and indep. features:



- Irreducible error due to additive noise is $\sigma = 1$
- High variance in SGD dynamics
- **Momentum** becomes unstable while **decay** (necessary to eliminate noise) performs best and is fastest

DIGRESSION: SOLVING OLS WITH QR DECOMP.

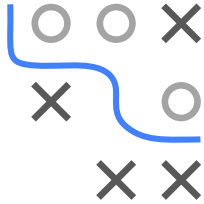


- Solving linear least squares via (S)GD is rarely done in practice
- But inversion of $\mathbf{X}^\top \mathbf{X}$ is numerically unstable and to be avoided
- A standard numerical approach applies **QR Decomposition**:
  - Factorize $\mathbf{X} \in \mathbb{R}^{n \times p}$ as $\mathbf{X} = \boldsymbol{QR}$, where $\boldsymbol{Q} \in \mathbb{R}^{n \times p}$ (thin form) so that $\boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}$ and $\boldsymbol{R} \in \mathbb{R}^{p \times p}$ is upper triangular
  - Purpose: replace solving $\hat{\boldsymbol{\theta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \boldsymbol{y}$ with a more numerically stable method by avoiding direct inversion
  - The QR decomposition can be computed using Gram-Schmidt orthogonalization or Householder transformations
- **Steps**:
  - Decompose $\mathbf{X}$ into $\boldsymbol{Q}$ and $\boldsymbol{R}$
  - Compute $\boldsymbol{Q}^\top \boldsymbol{y}$
  - Solve triangular system $\boldsymbol{R}\hat{\boldsymbol{\theta}} = \boldsymbol{Q}^\top \boldsymbol{y}$ via **back substitution**
  - Why this system? Remember normal equation for least squares problem is $\mathbf{X}^\top \mathbf{X}\hat{\boldsymbol{\theta}} = \mathbf{X}^\top \boldsymbol{y}$. Now replace $\mathbf{X} = \boldsymbol{QR}$:

$$\mathbf{X}^\top \mathbf{X}\hat{\boldsymbol{\theta}} = \mathbf{X}^\top \boldsymbol{y} \iff \boldsymbol{R}^\top (\boldsymbol{Q}^\top \boldsymbol{Q}) \boldsymbol{R}^\top \hat{\boldsymbol{\theta}} = \boldsymbol{R}^\top \boldsymbol{Q}^\top \boldsymbol{y}$$

$$\boldsymbol{R}^\top \boldsymbol{R}\hat{\boldsymbol{\theta}} = \boldsymbol{R}^\top \boldsymbol{Q}^\top \boldsymbol{y} \iff \boldsymbol{R}\hat{\boldsymbol{\theta}} = \boldsymbol{Q}^\top \boldsymbol{y}$$

DIGRESSION: SOLVING OLS WITH QR DECOMP.

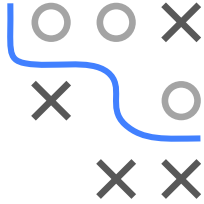


- $\boldsymbol{R}\hat{\boldsymbol{\theta}} = \boldsymbol{Q}^\top \boldsymbol{y}$ is a triangular system easily solvable by back substitution:

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1p} \\ 0 & r_{22} & \dots & r_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & r_{pp} \end{bmatrix}, \quad \hat{\boldsymbol{\theta}} = \begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \\ \vdots \\ \hat{\theta}_p \end{bmatrix}, \quad \boldsymbol{Q}^\top \boldsymbol{y} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{bmatrix}$$

- **Steps for Back Substitution**:
  - Start with the last equation (1 unknown): $\hat{\theta}_p = \frac{b_p}{r_{pp}}$
  - Move upwards to the $(p-1)$-th equation: $\hat{\theta}_{p-1} = \frac{b_{p-1} - r_{p-1,p}\hat{\theta}_p}{r_{p-1,p-1}}$
  - Continue this process up to the first row:
    $\hat{\theta}_i = \frac{b_i - \sum_{j=i+1}^{p} r_{ij}\hat{\theta}_j}{r_{ii}}$ for each $i = p-1, \dots, 1$
  - Back substitution leverages triangular structure of $\boldsymbol{R}$, moving upward from the last row and inserting already known values of $\hat{\boldsymbol{\theta}}$ as we go
    Back substitution leverages triangular structure of $\boldsymbol{R}$, moving upward from the last row and inserting already known values of $\hat{\boldsymbol{\theta}}$ as we go

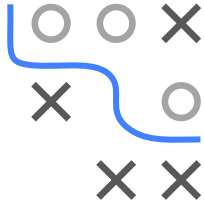
(S)GD FOR LOGISTIC REGRESSION



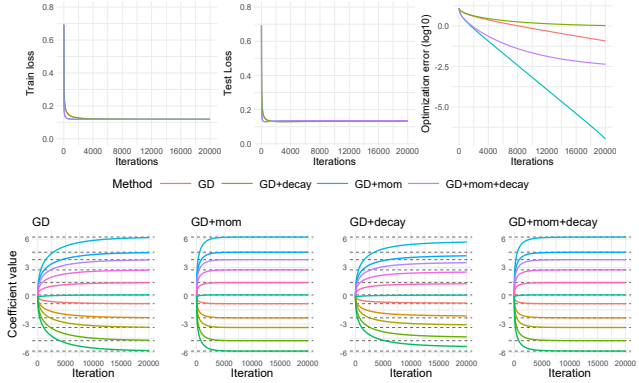
Comparison of (S)GD, (S)GD + momentum, and (S)GD + momentum + step size control on simulated data:

- Logistic regression (log loss) with same simulation setup as for linear regression
- To simulate response, we set $y^{(i)} \sim \mathcal{B}(\pi^{(i)}), \pi^{(i)} = \frac{1}{1+e^{-(\mathbf{x}^{(i)})^\top \boldsymbol{\theta}^*}}$
- We set the momentum parameter to 0.8 and $\texttt{decay} = 0.1$
- We again use different step sizes to illustrate benefit of momentum and decay depends on step size
- ERM has unique global minimizer but no closed-form solution. We can approximate $\hat{\boldsymbol{\theta}}$ using the `glm` solution (second-order optim)
- We also track the optimization error $\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|_2$

LOG-REG (GD + MED STEP SIZE)

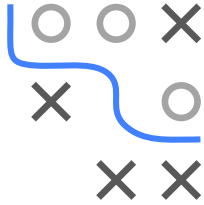


- GD with medium $\alpha = 0.25$ and indep. features:

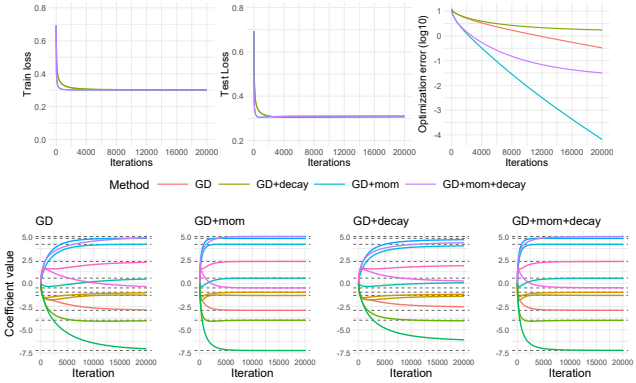


- Dotted lines indicate global minimizers
- All variants converge to global min
- **Momentum** accelerates and **decay** slows down optimization. **GD+mom** performs best

LOG-REG (GD + CORR. FEATURES)

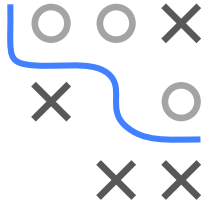


- GD with medium $\alpha = 0.25$ and bad conditioning (corr. features):

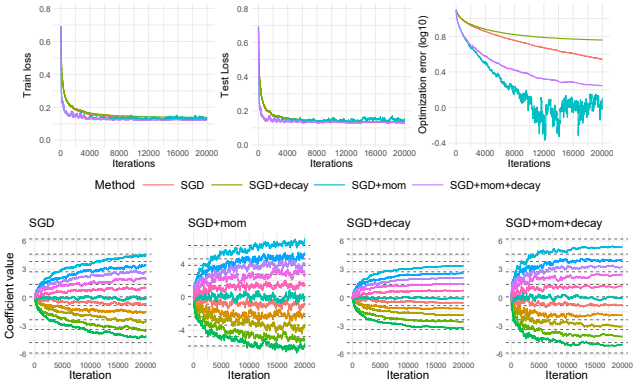


- Dotted lines indicate global minimizers
- Moderately bad conditioning slows down optim slightly and affects ERM solutions
- Only **momentum w/o decay** converges exactly to global min in $t_{\max}$
- Momentum causes "**overshooting**" of some coefs

LOG-REG (SGD + MED STEP SIZE)

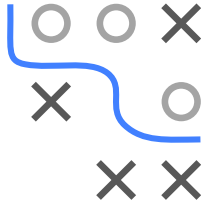


- SGD with medium $\alpha = 0.03$ and indep. features:

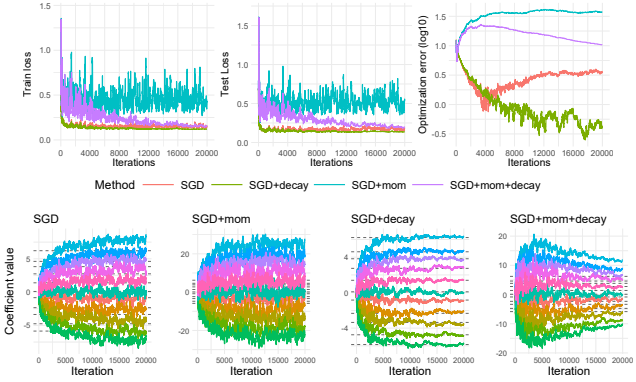


- Under SGD dynamics become more noisy
- **Momentum** accelerates while **decay** slows down optimization
- Only the coefs of **momentum w/o decay** come close to global minimizers

LOG-REG (SGD + LARGE STEP SIZE)

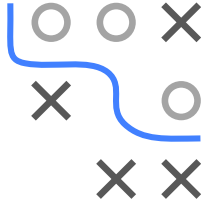


- SGD with large $\alpha = 0.3$ and indep. features:



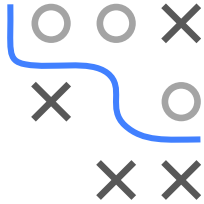
- **High variance** in optim dynamics
- Plain SGD and **Momentum** become unstable and oscillate at suboptimal loss values/diverge while **decay** (necessary to eliminate oscillations) performs best and is fastest
- **Momentum+decay** initially diverges but recovers once step size reduces (would converge with many more steps)

CLASSIFICATION ON MNIST (SGD)

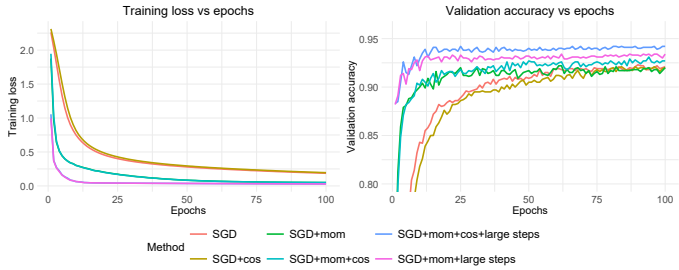


- For a more realistic application, we compare optimizers on training an NN on subset of MNIST image classification task
- DNN has two hidden layers with 128 and 64 ReLU-activated units and Kaiming normal init. The loss is cross-entropy
- Instead of SGD we use mini-batch SGD with 100 images per batch
- For the step size schedule, we use cosine decay $\alpha^{[t]} = \alpha^{[0]} \left[(1 - r_{\text{min}}) \cdot \frac{1}{2} \left(1 + \cos \left(\pi \cdot \frac{t}{t_{\text{max}}} \right) \right) + r_{\text{min}} \right]$ with final step size fraction $r_{\text{min}} = 0.01$
- In case of momentum we set the parameter to 0.8
- Regular initial step size is 0.01 and to 0.1 for large step size

CLASSIFICATION ON MNIST (SGD)



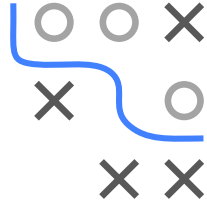
- Mini-batch SGD with $\alpha \in \{0.01, 0.1\}$ for 100 epochs (5000 iterations):



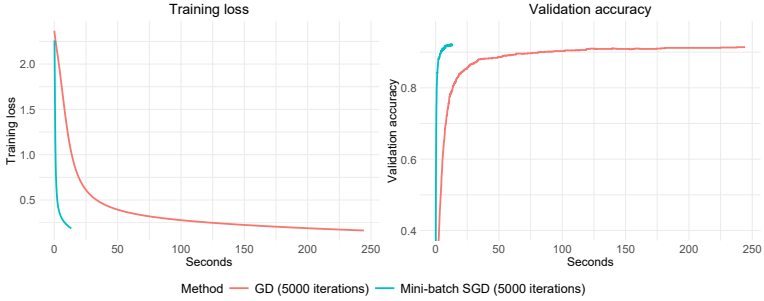
Observations (green/cyan + blue/purple train losses overlap but not val. acc.):

- **Momentum** drastically speeds up optimization in all settings
- **Plain SGD**: step size decay slows optimization slightly
- SGD, SGD+cos and SGD+mom achieve **same** val. acc. $\Rightarrow$ no generalization benefit for medium α
- **Cosine decay+momentum** improves generalization for medium α slightly
- **Large step size** with momentum and decay performs best

CLASSIFICATION ON MNIST (GD VS. SGD)



- Why is it not a good idea to use GD in most DL applications? SGD is much faster
- Compare runtime of mini-batch SGD (batch size=100) with GD (constant $\alpha = 0.01$ without momentum for $t_{\text{max}} = 5000$ iterations):



Observations:

- Mini-batch SGD is over an order of magnitude faster
- SGD generalizes better than GD despite being much faster