Regression

5. Incremental and iterative methods

Olivier Sigaud

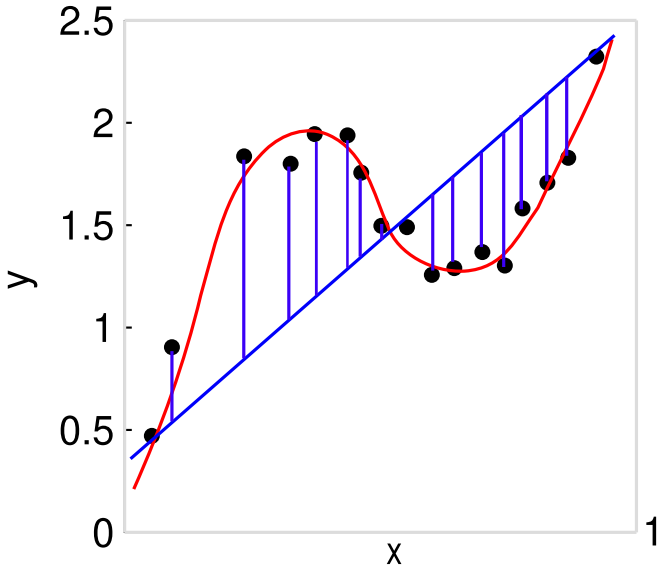
Sorbonne Université
http://people.isir.upmc.fr/sigaud

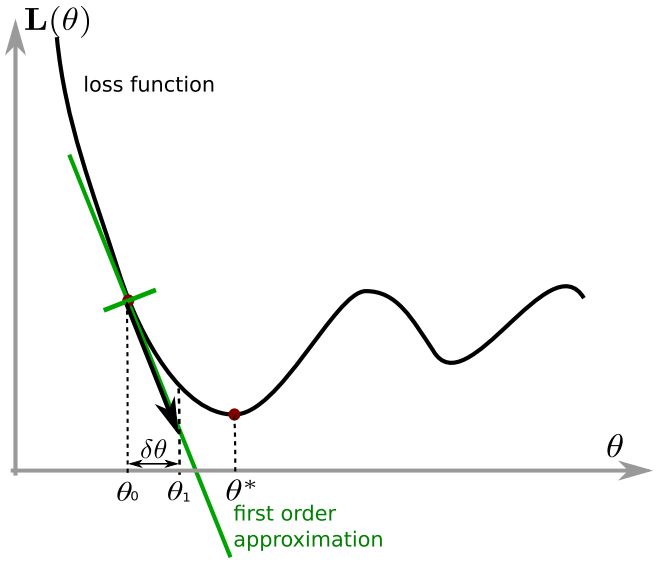
Gradient descent

Limiting the cost of regression

- Solving $\boldsymbol{\theta}^* = (\mathbf{G}^\intercal \mathbf{G})^{-1} \mathbf{G}^\intercal \mathbf{y}$ (or any variant) requires inverting $(\mathbf{G}^\intercal \mathbf{G})$
- That is in $O(N^3)$ in the number of dimensions
- For large linear architectures or deep neural networks, this is too expensive
- Incremental solutions:
  - Complexity can be reduced to $O(N^2)$ by using the Sherman-Morrisson formula $\rightarrow$ incremental update of the inverse,
  - But sensitive to rounding errors.
  - Numerically more stable option: updating the Cholesky factor of the matrix using the QR algorithm.
  - In the linear case: Recursive Least Squares

Regression through batch gradient descent

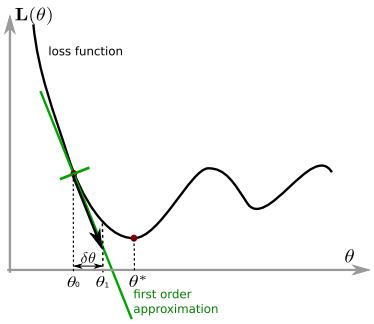




- We want to minimize a regression error (left)
- Thus find the minimum of a **loss function** $\mathbf{L}(\boldsymbol{\theta})$ (right)
- Instead of solving $\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} \mathbf{L}(\boldsymbol{\theta})$ (in one step, but expensive)
- Rather perform local steps using an iterative approach
- The iteration equation is $\boldsymbol{\theta}_{i+1} \leftarrow \boldsymbol{\theta}_i + \delta\boldsymbol{\theta}_i$
- This is the essence of gradient descent

Gradient descent is steepest descent

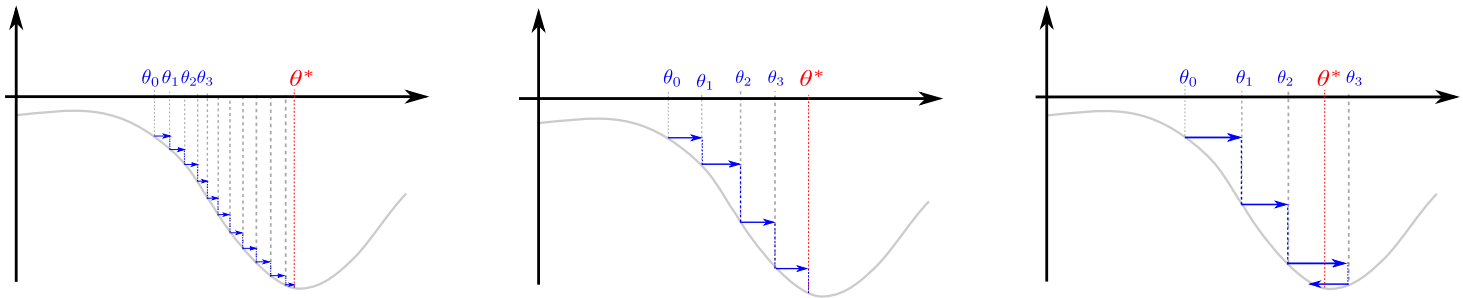
- We want to minimize $\mathbf{L}(\boldsymbol{\theta}_i + \delta\boldsymbol{\theta}_i)$ over $\delta\boldsymbol{\theta}_i$
- How to choose $\delta\boldsymbol{\theta}_i$?



- The optimum of $\mathbf{J}(\boldsymbol{\theta} + \delta\boldsymbol{\theta})$ over $\delta\boldsymbol{\theta}$ is reached when $\dfrac{\partial \mathbf{J}(\boldsymbol{\theta} + \delta\boldsymbol{\theta})}{\partial \delta\boldsymbol{\theta}} = 0$
- First order approx: $\mathbf{J}(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}\mathbf{J}(\boldsymbol{\theta})^T \delta\boldsymbol{\theta} + \nu\delta\boldsymbol{\theta}^T\delta\boldsymbol{\theta} +$ higher order terms
- $\dfrac{\partial \mathbf{J}(\boldsymbol{\theta} + \delta\boldsymbol{\theta})}{\partial \delta\boldsymbol{\theta}} \sim \nabla_{\boldsymbol{\theta}}\mathbf{J}(\boldsymbol{\theta})^T \delta\boldsymbol{\theta} + 2\nu\delta\boldsymbol{\theta}$
- $\delta\boldsymbol{\theta}^* = -\alpha\nabla_{\boldsymbol{\theta}}\mathbf{J}(\boldsymbol{\theta})$

- Thus the steepest descent direction is given by the first order derivative

Tuning the step size

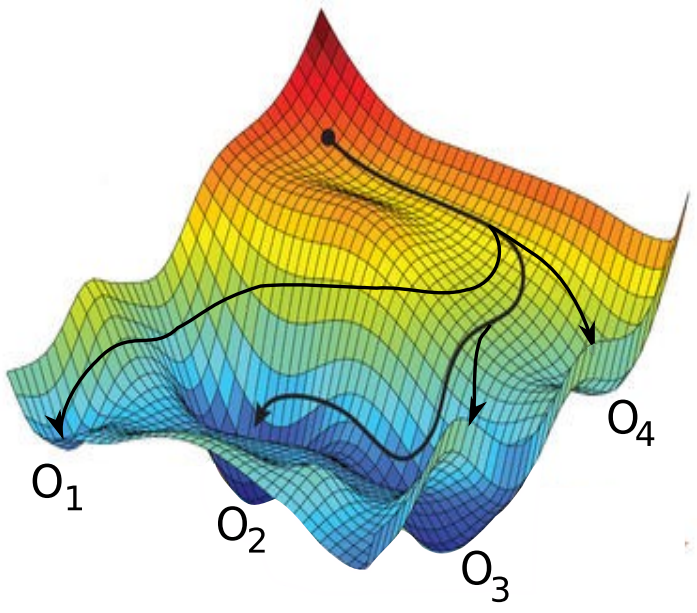


(a) (b) (c)

- Minimizing the first order derivative is not lower bounded
- We need a step size α to determine how far to go
- (a) α_i are too small, (b) α_i are adequate, (c) α_i are too large
- If too small, too many steps. If too large, may miss a local optimum
- Line search: iterate to find the best step size (used e.g. in TRPO)

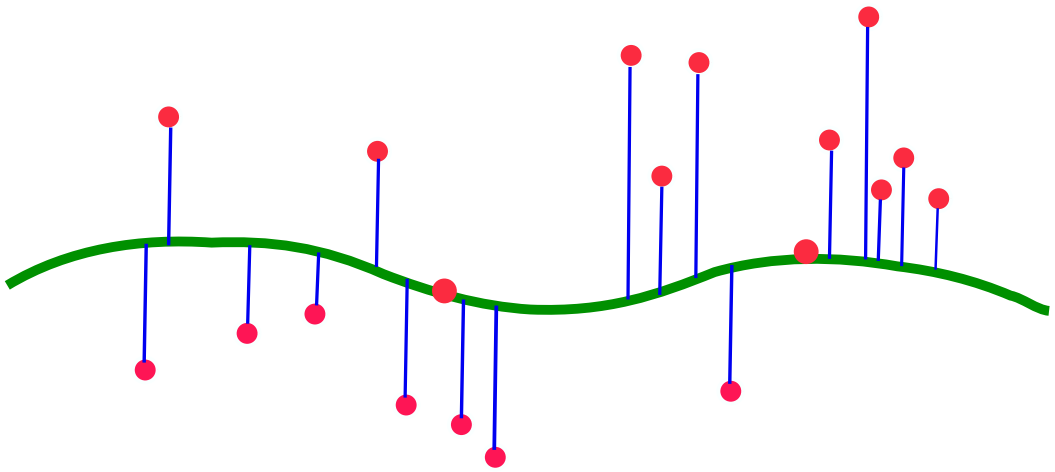
Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015) Trust region policy optimization. *CoRR, abs/1502.05477*

Local Optima



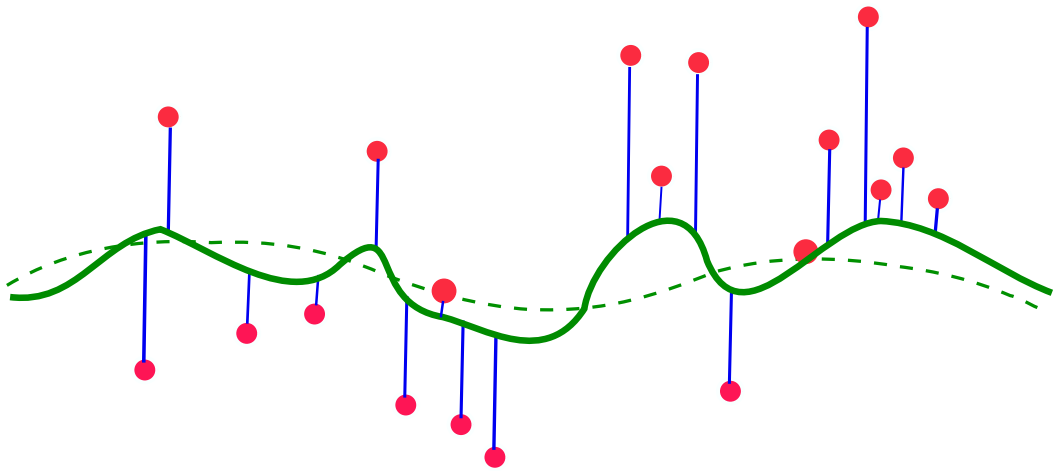
- Gradient descent is a local improvement approach where, at each step, we follow the steepest descent direction
- But we don't know where is the optimum we are targetting
- Unless the cost function is convex in the parameter space, gradient descent can end-up in different local optima depending on the starting point or noise
- Anything that basically goes down will end up somewhere low!

Regression through batch gradient descent: illustration



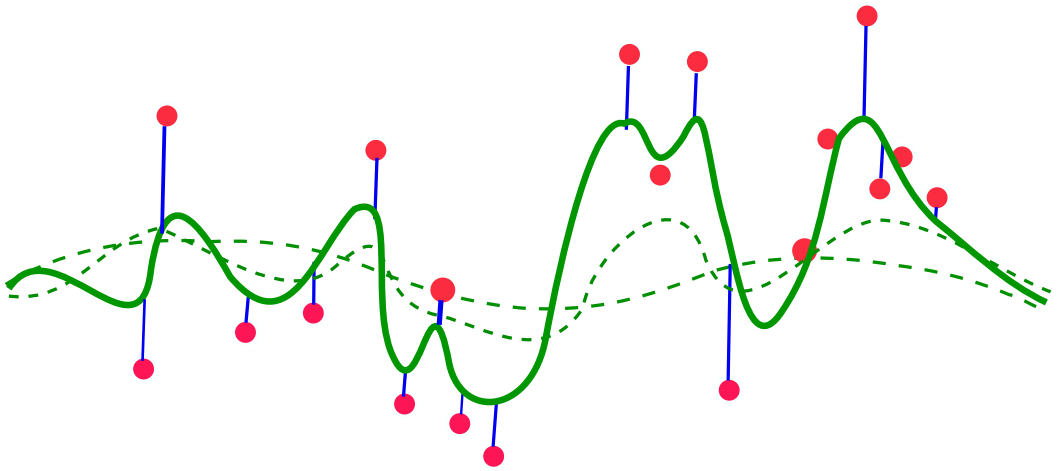
- As usual, the loss is the squared sum of all residual errors $\mathbf{L}(\boldsymbol{\theta}) = \|\mathbf{y} - \hat{f}(\mathbf{X})\|^2$

Regression through batch gradient descent: illustration



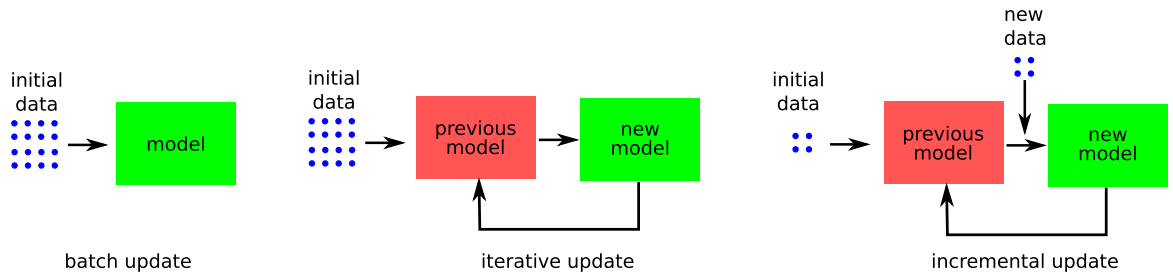
- As usual, the loss is the squared sum of all residual errors $\mathbf{L}(\boldsymbol{\theta}) = \|\mathbf{y} - \hat{f}(\mathbf{X})\|^2$

Regression through batch gradient descent: illustration



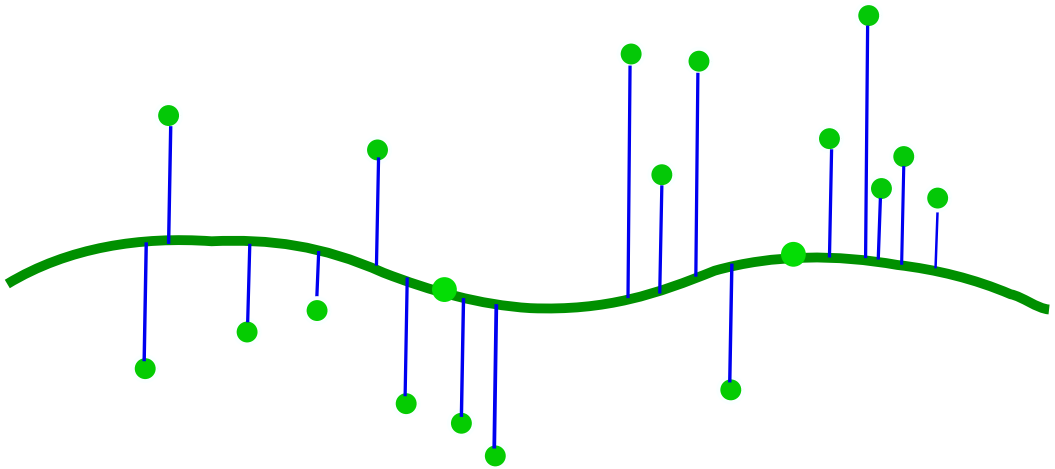
- As usual, the loss is the squared sum of all residual errors $\mathbf{L}(\boldsymbol{\theta}) = \|\mathbf{y} - \hat{f}(\mathbf{X})\|^2$

Batch, Iterative, Incremental



- So far, we have focused on batch regression
- Iterative methods: you improve a model through steps, using the same data (your batch) at all steps
- Incremental methods: you improve a model through steps, with additional data at each step
- Incremental implies more than iterative
- Batch gradient descent is only iterative: no need for additional data
- Stochastic gradient descent is a specific case of incremental ($\mathbf{y}$ and $\mathbf{X}$ change for each mini-batch)
- The new datapoints are taken randomly from the batch

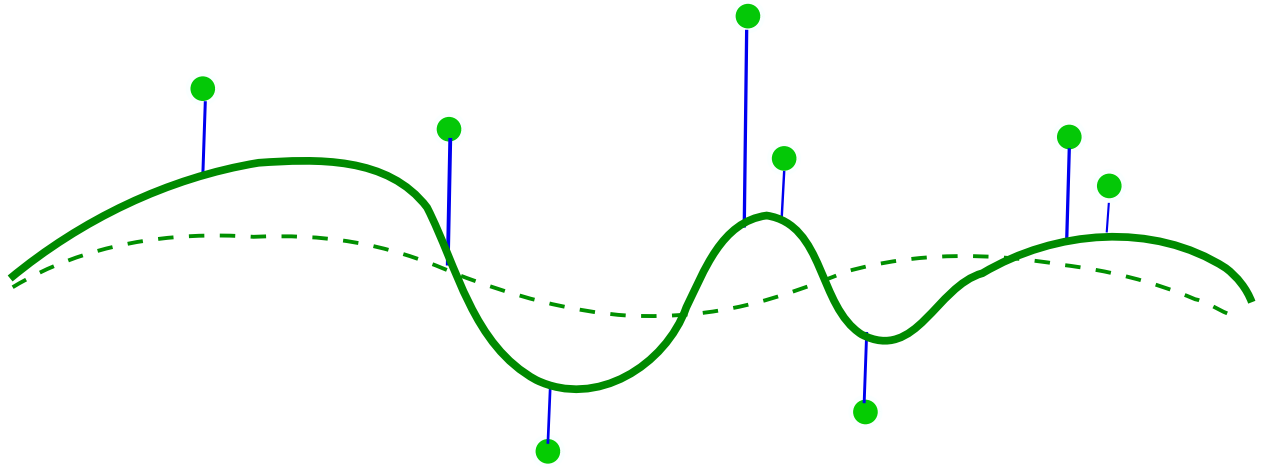
Stochastic gradient descent (SGD)



- Batch gradient descent is expensive when the batch is large
- We can decrease the cost by using several mini-batches (extra gain with parallel computations)
- In SGD, each mini-batch gives a different (inaccurate) estimate of the local gradient
- Combining several inaccurate estimates can help being accurate
- Note that SGD converges more slowly. Use it only for large batches.

Bottou, L. (2012) Stochastic gradient descent tricks. *Neural networks: Tricks of the trade*, pp. 421–436. Springer

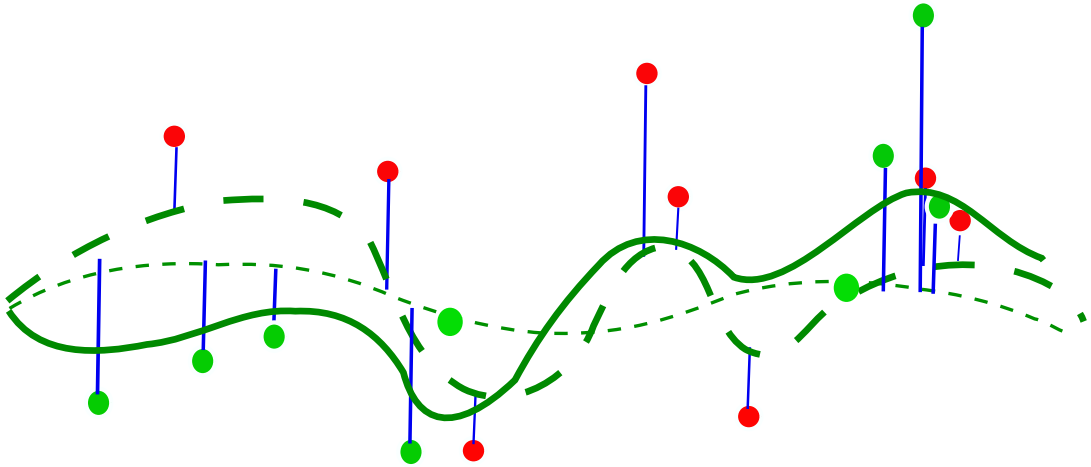
Stochastic gradient descent (SGD)



- Batch gradient descent is expensive when the batch is large
- We can decrease the cost by using several mini-batches (extra gain with parallel computations)
- In SGD, each mini-batch gives a different (inaccurate) estimate of the local gradient
- Combining several inaccurate estimates can help being accurate
- Note that SGD converges more slowly. Use it only for large batches.

Bottou, L. (2012) Stochastic gradient descent tricks. *Neural networks: Tricks of the trade*, pp. 421–436. Springer

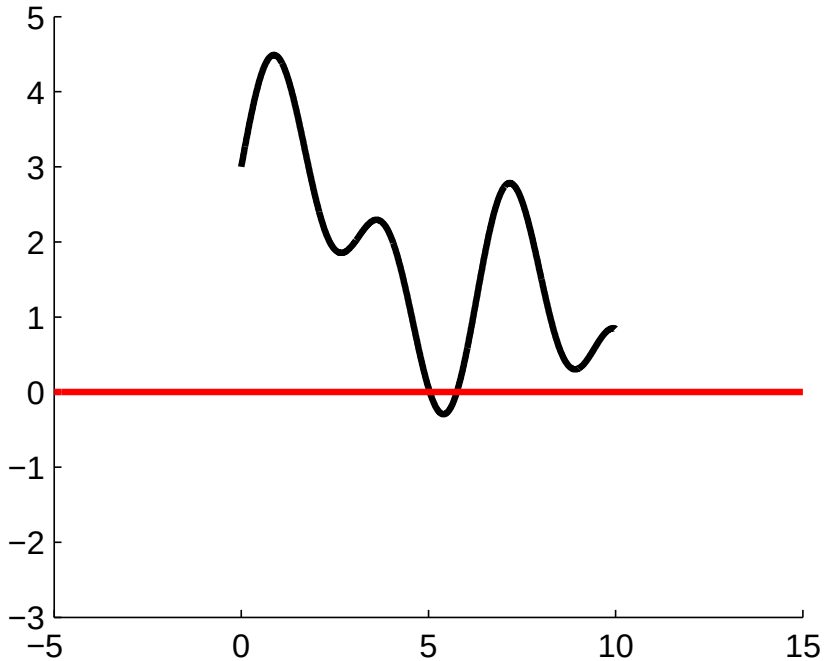
Stochastic gradient descent (SGD)



- Batch gradient descent is expensive when the batch is large
- We can decrease the cost by using several mini-batches (extra gain with parallel computations)
- In SGD, each mini-batch gives a different (inaccurate) estimate of the local gradient
- Combining several inaccurate estimates can help being accurate
- Note that SGD converges more slowly. Use it only for large batches.

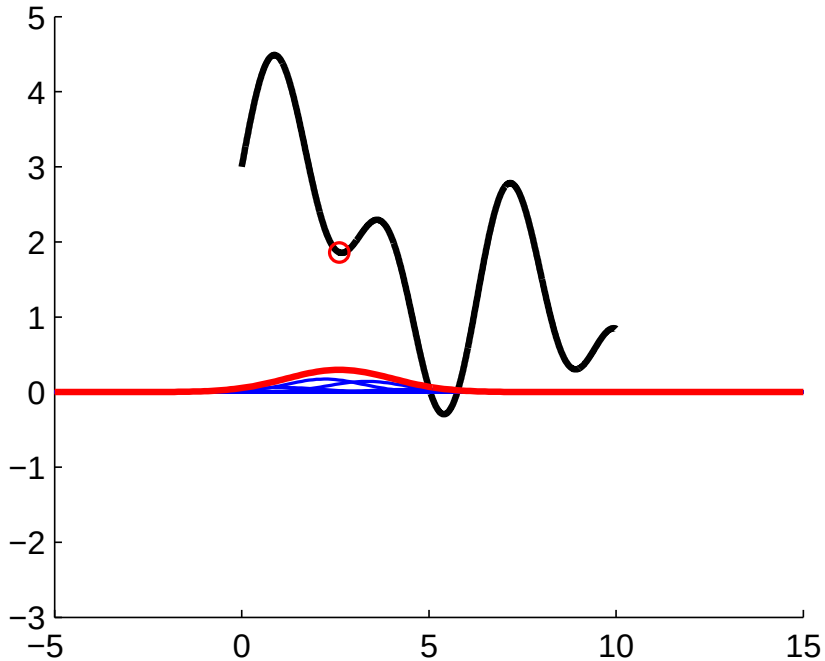
Bottou, L. (2012) Stochastic gradient descent tricks. *Neural networks: Tricks of the trade*, pp. 421–436. Springer

Stochastic gradient descent with RBFNs: Illustration



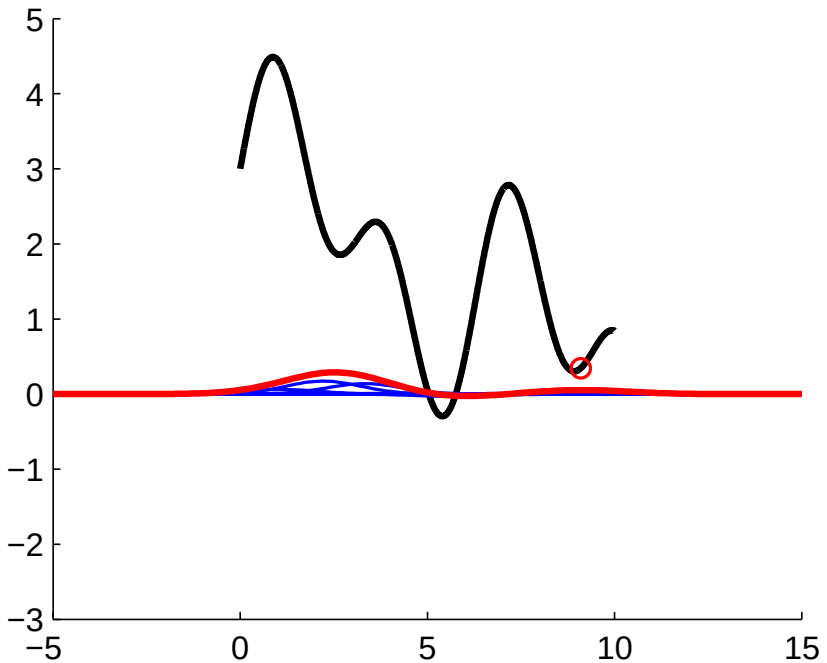
- Extreme case where we take one point of the batch at each iteration

Stochastic gradient descent with RBFNs: Illustration



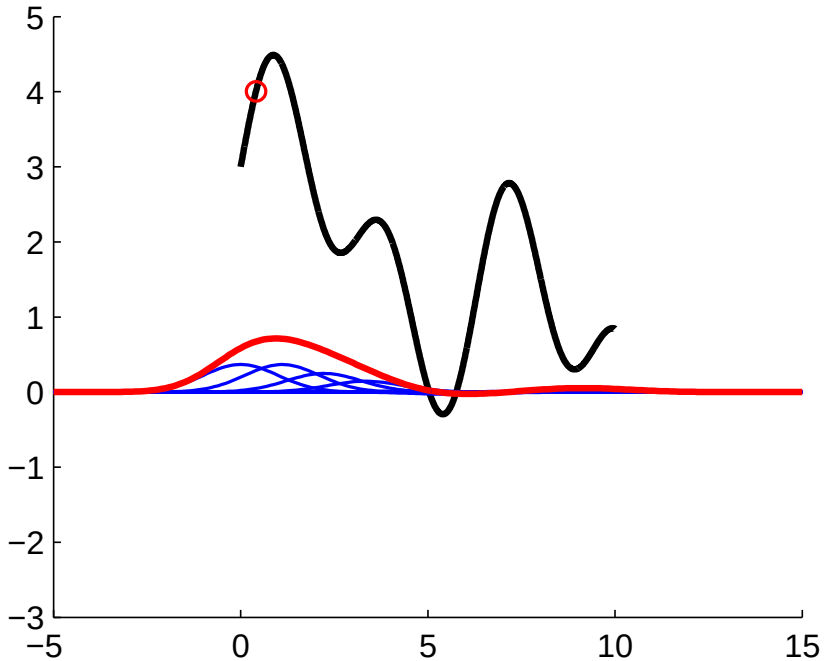
- Extreme case where we take one point of the batch at each iteration

Stochastic gradient descent with RBFNs: Illustration



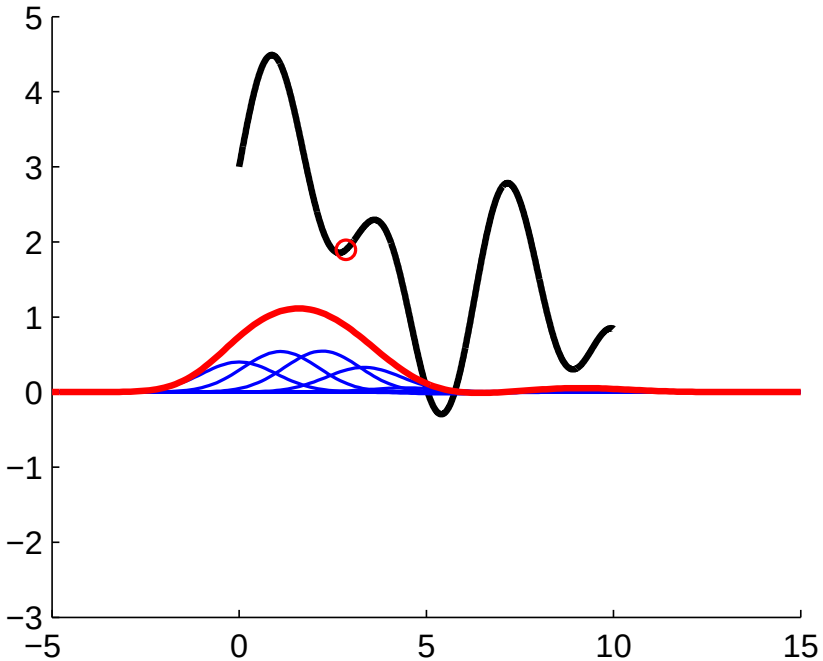
- Extreme case where we take one point of the batch at each iteration

Stochastic gradient descent with RBFNs: Illustration



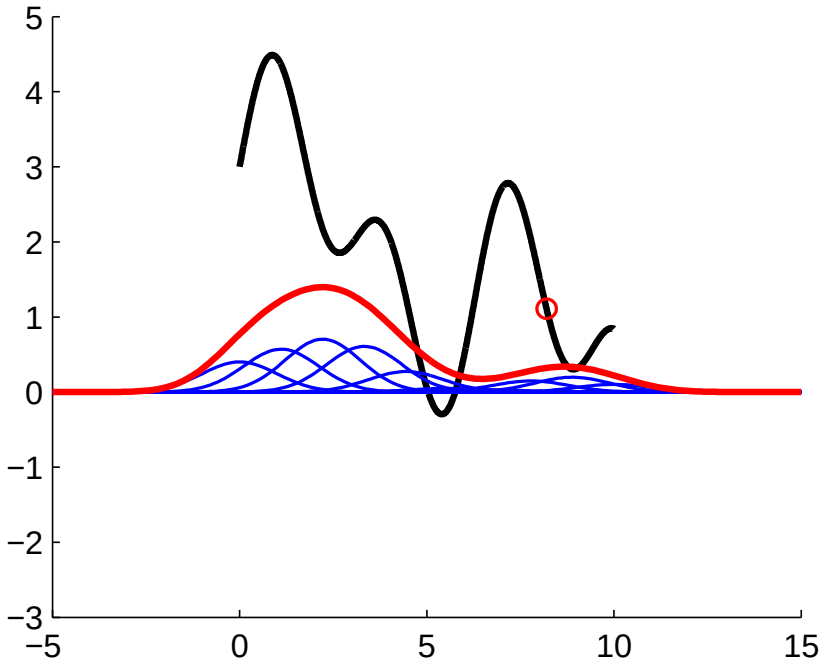
- Extreme case where we take one point of the batch at each iteration

Stochastic gradient descent with RBFNs: Illustration



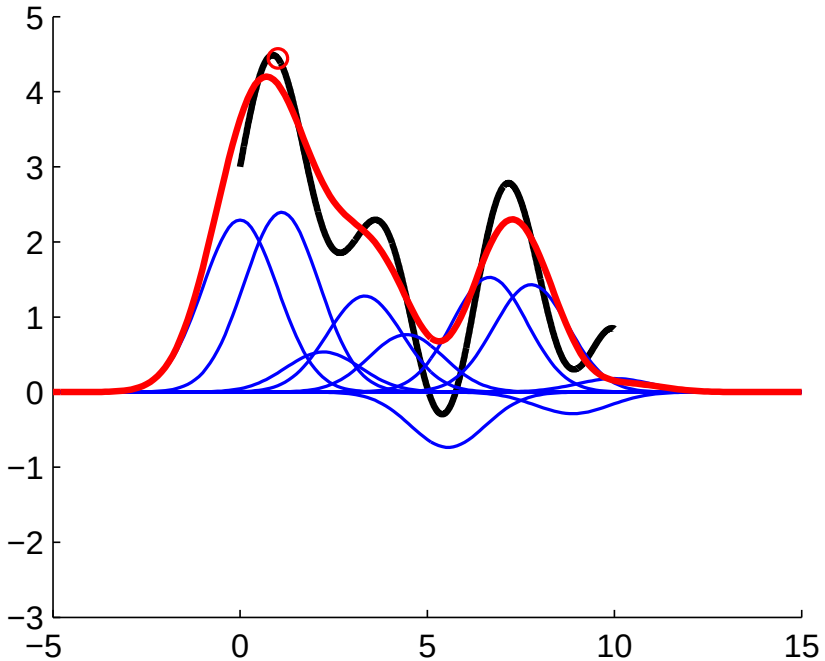
- Extreme case where we take one point of the batch at each iteration

Stochastic gradient descent with RBFNs: Illustration



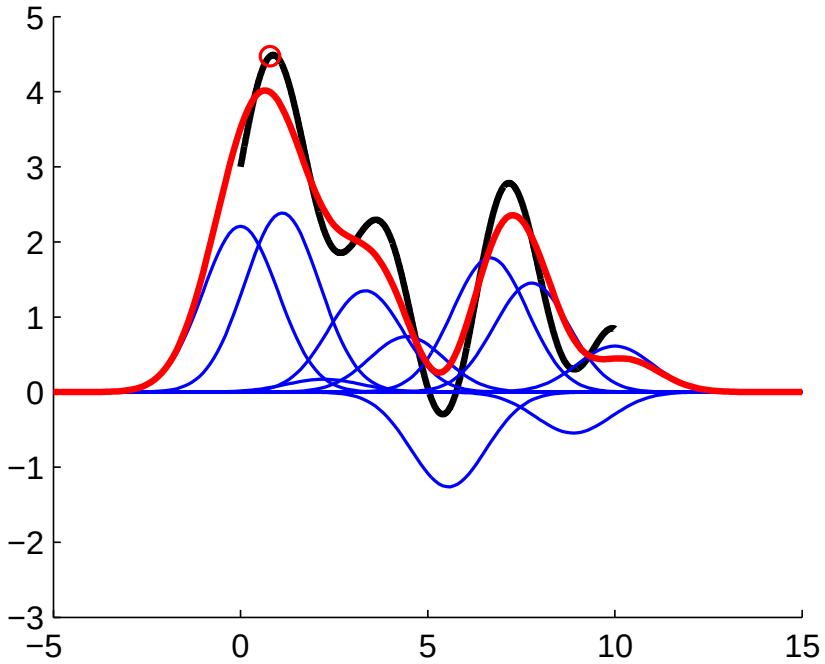
- Extreme case where we take one point of the batch at each iteration

Stochastic gradient descent with RBFNs: Illustration



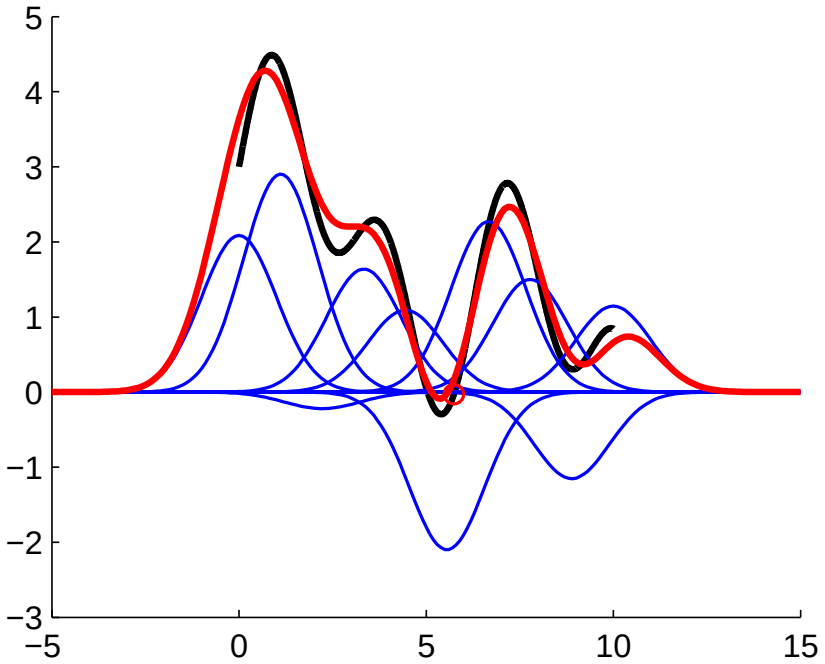
- Extreme case where we take one point of the batch at each iteration

Stochastic gradient descent with RBFNs: Illustration



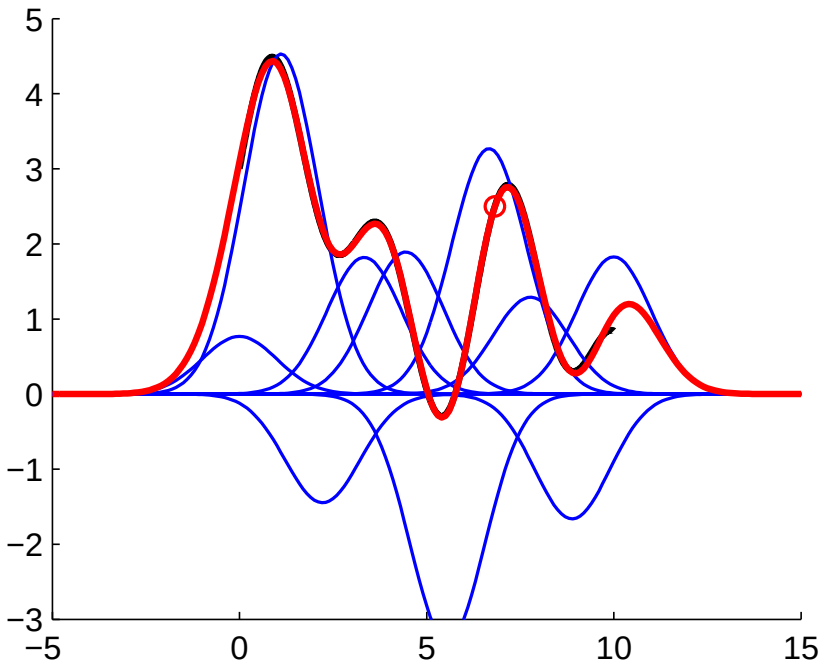
- Extreme case where we take one point of the batch at each iteration

Stochastic gradient descent with RBFNs: Illustration



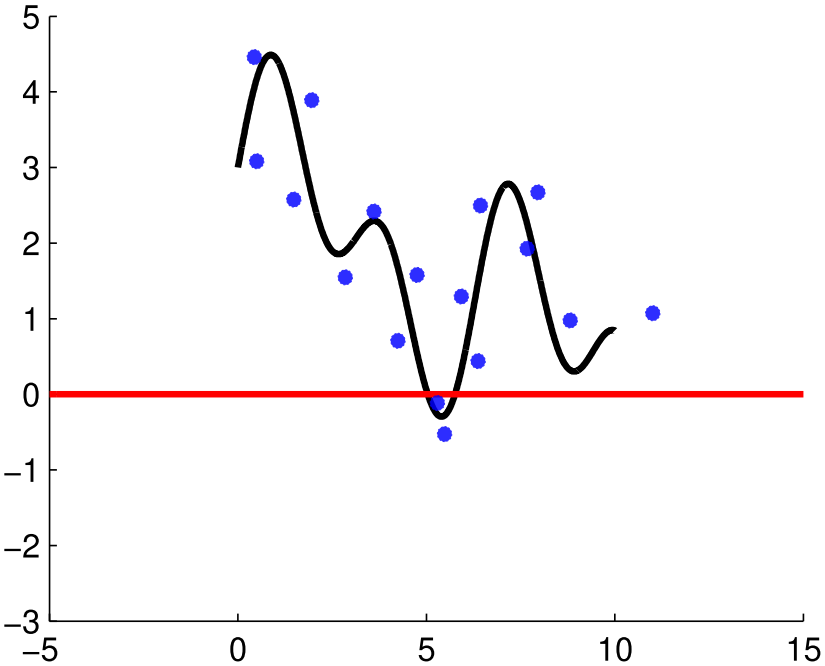
- Extreme case where we take one point of the batch at each iteration

Stochastic gradient descent with RBFNs: Illustration



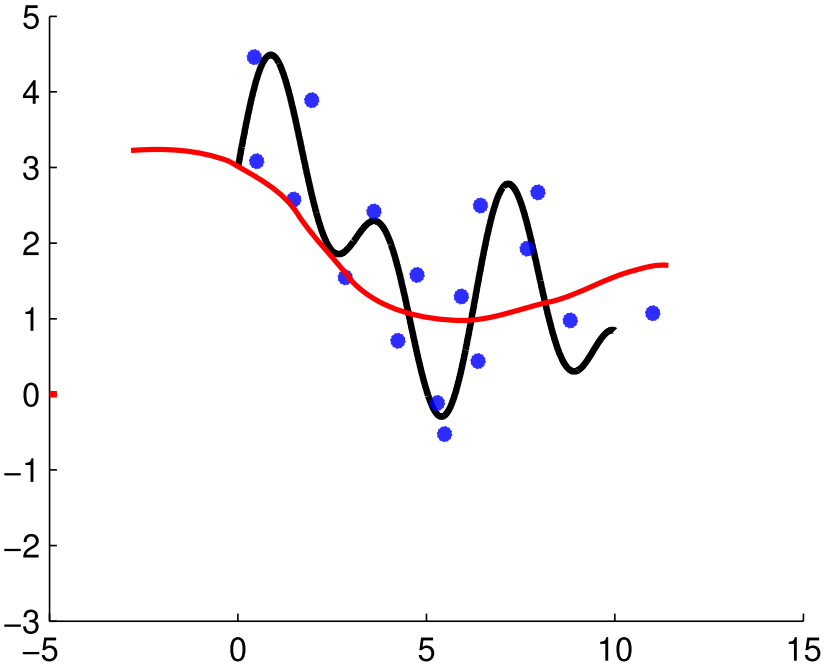
- Extreme case where we take one point of the batch at each iteration

Risk of overfitting



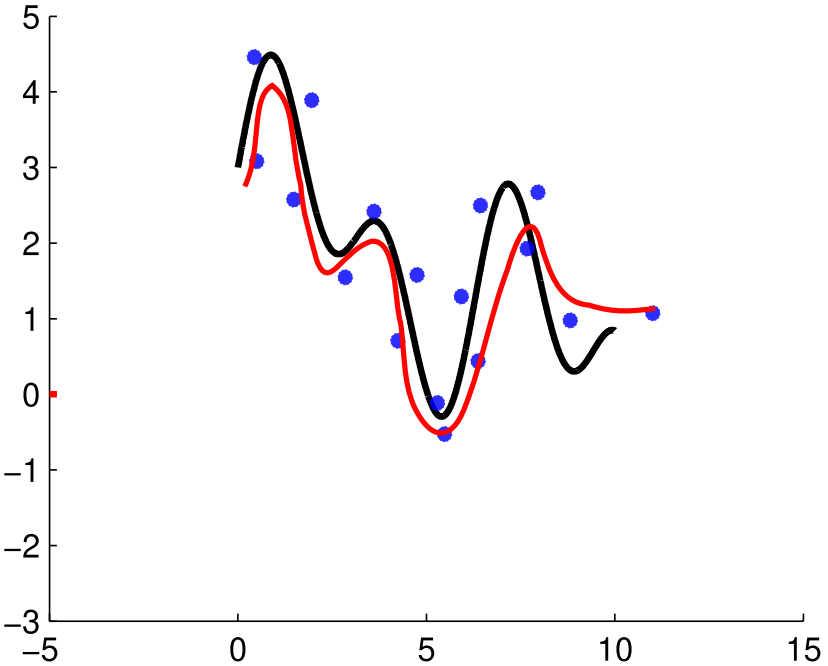
- In the previous simulation, measurements were noiseless
- What if we get noisy measurements?
- The risk is overfitting to data
- The question is when to stop

Risk of overfitting



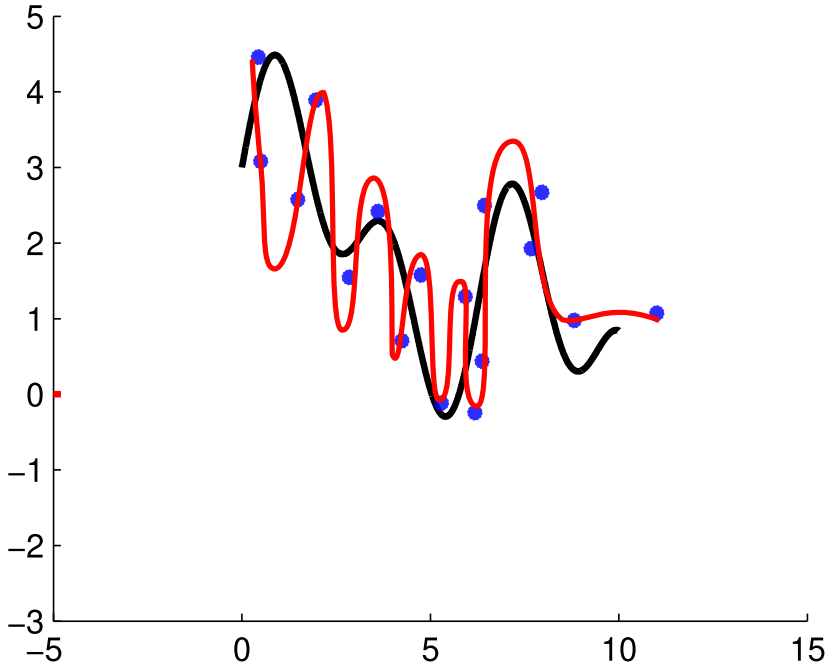
- In the previous simulation, measurements were noiseless
- What if we get noisy measurements?
- The risk is overfitting to data
- The question is when to stop

Risk of overfitting



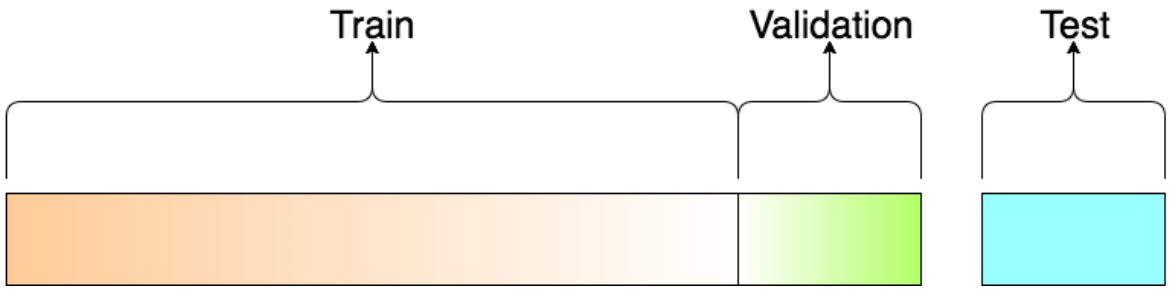
- In the previous simulation, measurements were noiseless
- What if we get noisy measurements?
- The risk is overfitting to data
- The question is when to stop

Risk of overfitting



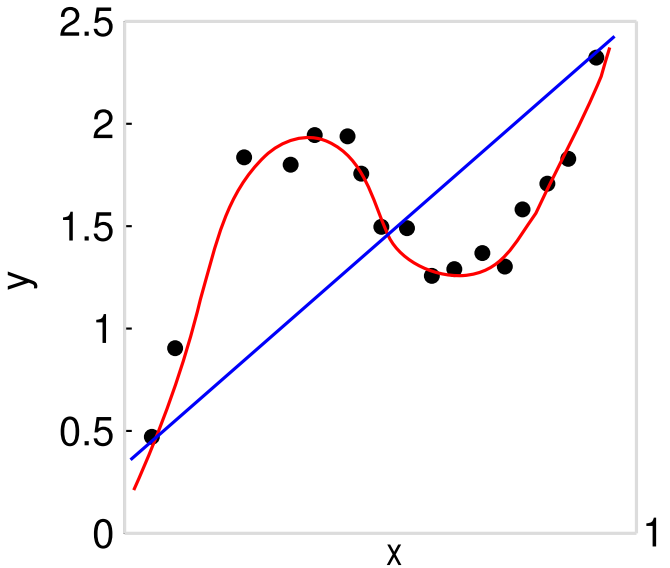
- In the previous simulation, measurements were noiseless
- What if we get noisy measurements?
- The risk is overfitting to data
- The question is when to stop

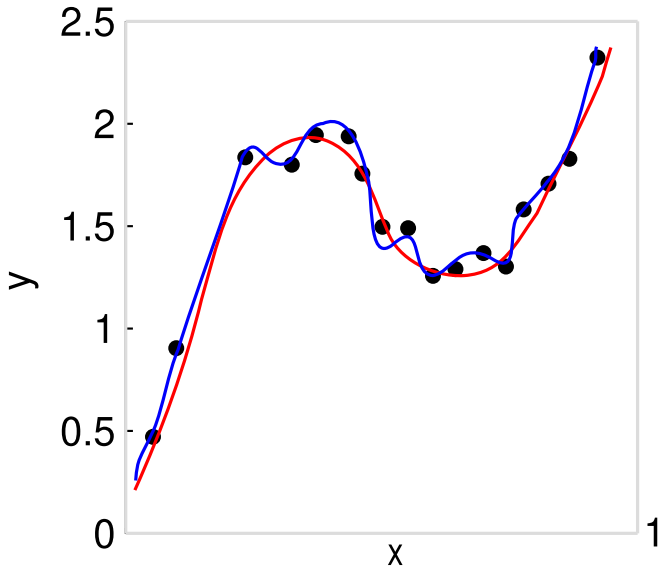
Fighting overfitting: Standard methodology



- Split the training set into training set and validation set
- Train on the training set
- Evaluate on the validation set
- When performance stops improving, stop iterating
- Should generalize well to the test set

Overfitting: influence of the model





- A model with more parameters is more prone to overfitting
- If the model does not have enough free parameters:
  - The training error may not go down towards 0
  - The validation performance may stop improving
- → Select a rich enough model...
- ...and use the validation trick to decide when to stop
- (Deep) neural networks are particularly rich models

Bottou, L. (2012).

Stochastic gradient descent tricks.
In *Neural networks: Tricks of the trade*, pages 421–436. Springer.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. (2015).

Trust region policy optimization.
In Bach, F. R. and Blei, D. M., editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1889–1897. JMLR.org.