Lecture 3

Stochastic gradient descent, Stein’s unbiased risk estimator

Stochastic gradient descent

Suppose that we want to minimize an objective function that is written as a sum of differentiable functions.

$Q(w) = \sum_{i=1}^{n} Q_i(w)$

Each term Q_i is usually associated with the i_th data point.

Standard gradient descent (batch gradient descent):
$w = w - \eta \nabla Q(w) = w - \eta \sum_{i=1}^{n} \nabla Q_i(w)$

where η is the learning rate (step size).

Stochastic gradient descent

Stochastic gradient descent (SGD) considers only a subset of summand functions at every iteration.

This can be quite effective for large-scale problems.

Bottou, Leon; Bousquet, Olivier (2008). The Tradeoffs of Large Scale Learning. Advances in Neural Information Processing Systems 20. pp. 161168.

The gradient of $Q(w)$ is approximated by a gradient at a single example: $w = w - \eta \nabla Q_i(w)$.

This update needs to be done for each training example.

Several passes might be necessary over the training set until the algorithm converges.

η might be adaptive.

Stochastic gradient descent

- Choose an initial value for w and η.
- Repeat until converged
  - Randomly shuffle data points in the training set.
  - For $i = 1, 2, \ldots, n$, do:
  - $w = w - \eta \nabla Q_i(w)$.

Example

Suppose $y = w_1 + w_2 x$

The objective function is:

$$Q(w) = \sum_{i=1}^{n} Q_i(w) = \sum_{i=1}^{n} (w_1 + w_2 x_i - y_i)^2.$$

Update rule will become:

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} := \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \eta \begin{bmatrix} 2(w_1 + w_2 x_i - y_i) \\ 2x_i(w_1 + w_2 x_i - y_i) \end{bmatrix}.$$

Example from Wikipedia

Why Stochastic Gradient Descent (SGD) Works

Problem Statement:

- We want to minimize a loss function $Q(\boldsymbol{w})$ where $\boldsymbol{w}$ is the parameter vector.

- The loss function is assumed to be differentiable.

- Show that Stochastic Gradient Descent (SGD) converges to a local minimum of $Q(\boldsymbol{w})$.

Why Stochastic Gradient Descent (SGD) Works

- **Initialization:** Start with an initial guess w_0
- **Update Rule:**

$$w_{\{t+1\}} = w_t - \rho \nabla Q_{i(w_t)}.$$

- **Expectation of the Gradient:**

$$E\left[\nabla Q_{i(w_t)}\right] = \nabla Q(w_t).$$

- **Convergence:** If ρ is sufficiently small, then each update moves w_t closer to the minimum of $Q(w)$.

Why Stochastic Gradient Descent (SGD) Works

- Expectation of Update Rule:

$$E\left[w_{\{t+1\}}\right] = E[w_t] - \rho E\left[\nabla Q_{i(w_t)}\right]$$

- Substitute Expectation of Gradient:

$$\mathrm{E}\left[w_{\{t+1\}}\right] = E[w_t] - \rho \nabla Q(w_t)).$$

- Convergence: This equation implies that the expected value of w moves in the direction of the negative gradient, thus moving towards the minimum of $Q(w)$

Why Stochastic Gradient Descent (SGD) Works

- The proof shows that the expected update of w using SGD is in the direction of the true gradient of the loss function $Q(w)$

- Therefore, SGD works in practice to find a local minimum of the loss function, provided the learning rate ρ is appropriately chosen.

Mini-batches

Batch gradient decent uses all n data points in each iteration.

Stochastic gradient decent uses 1 data point in each iteration.

Mini-batch gradient decent uses b data points in each iteration.

b is a parameter called Mini-batch size.

Mini-batches

- Choose an initial value for w and η.
- Say $b = 10$
- Repeat until converged
  - Randomly shuffle data points in the training set.
  - For $i = 1, 11, 21, \ldots, n-9$, do:
  - $w = w - \eta \sum_{k=i}^{i+9} \nabla Q_i(w)$.

Tuning η

If η is too high, the algorithm diverges.

If η is too low, makes the algorithm slow to converge.

A common practice is to make η_t a decreasing function of the iteration number t. e.g. $\eta_t = \frac{constant1}{t+constant2}$

The first iterations cause large changes in the w, while the later ones do only fine-tuning.

Momentum

- Momentum is a technique used in optimization to accelerate convergence.

- Inspired by physical momentum, it helps in navigating the optimization landscape.

Mathematical Formulation

- Standard GD Update:

$$w_{\{t+1\}} = w_t - \rho \nabla Q(w_t)$$

- Momentum Update:

$$v_{\{t+1\}} = \beta v_t + (1 - \beta) \nabla Q(w_t)$$

$$w_{\{t+1\}} = w_t - \rho v_{\{t+1\}}$$

- β: Momentum coefficient (0.9 to 0.99)
- v_t: Velocity term, a running average of gradients

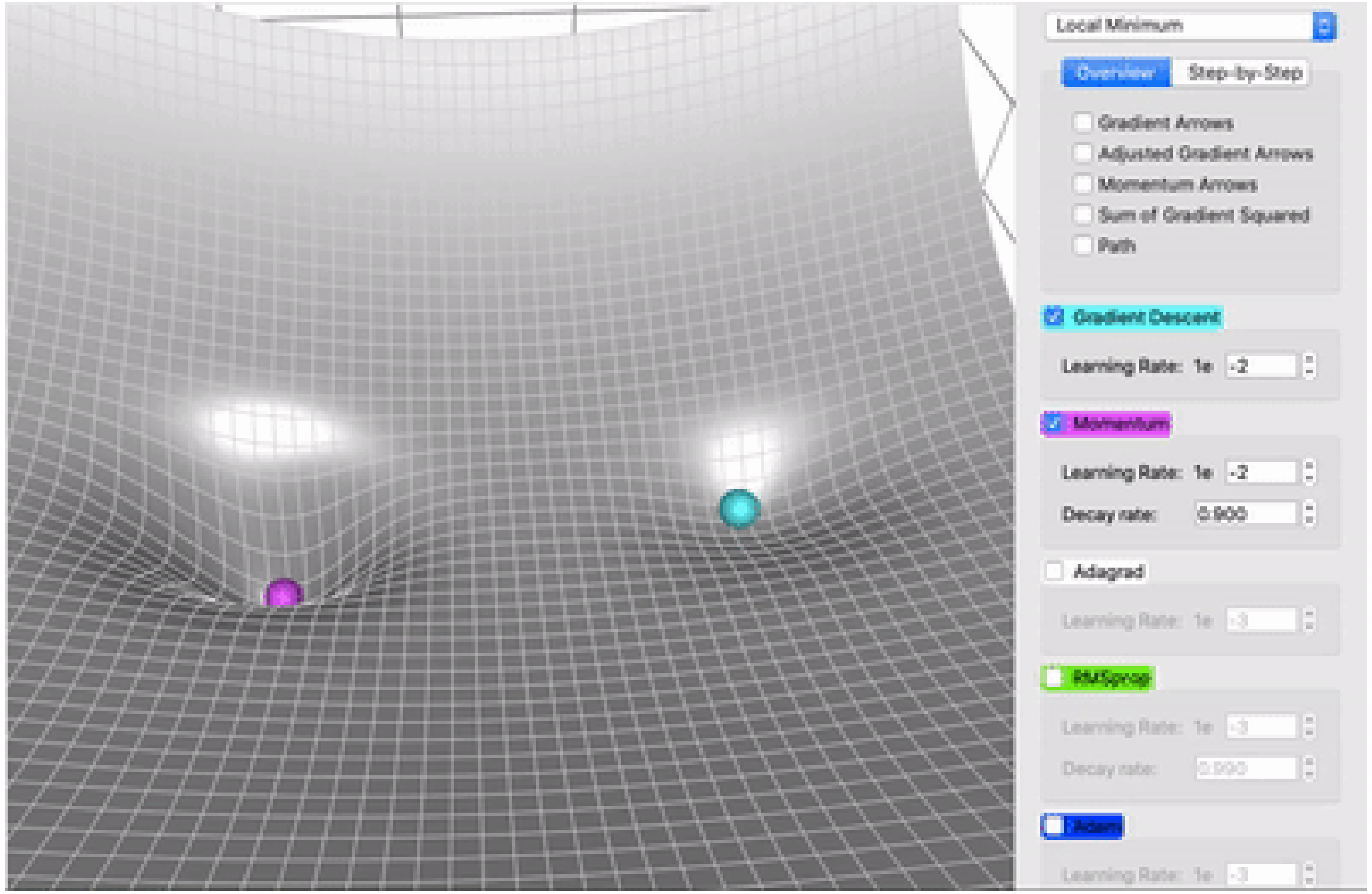
Understanding the Velocity Term

- The velocity term v_t is a running average of past gradients.
- It accumulates information from past updates to inform the next step.
- Mathematical Update:

$$v_{\{t+1\}} = \beta v_t + (1 - \beta)\nabla Q(w_t)$$

Role in Optimization:

- Smoothing out updates
- Accelerating through flat regions
- Providing stability



Momentum (magenta) vs. Gradient Descent (cyan) on a surface with a global minimum (the left well) and local minimum (the right well)

Optimization Algorithms in Deep Learning

- Adam is currently the most popular optimization algorithm in deep learning.
- However, there are some concerns about its generalization performance compared to stochastic gradient descent (SGD).
- Other optimization algorithms, such as AMSGrad, AdamW, QHAdam, YellowFin, Demon Adam, Momentum, and Aggmo QHM, have been proposed and evaluated on various test problems.
- SGD is slower to converge but generally performs better in terms of generalization.

Challenges of Second-Order Optimization in Deep Learning

Computational Complexity

- Requires Hessian matrix: $O(n^2)$ complexity
- Matrix inversion: $O(n^3)$ complexity

Memory Requirements

- Hessian storage: $O(n^2)$ memory
- **Non-Convexity**
- Risk of saddle points and local minima
-

Stein’s unbiased risk estimator

Model Selection

- The general task in machine learning is estimating a function.
  - We want to estimate :

$$\hat{f}(x) \quad \text{estimated function}$$

Model Selection

- The general task in machine learning is estimating a function.
  - We want to estimate :

$$\hat{f}(x) \quad \text{estimated function}$$

  - Where there is a true underlying function :

$$f(x) \quad \text{true function}$$

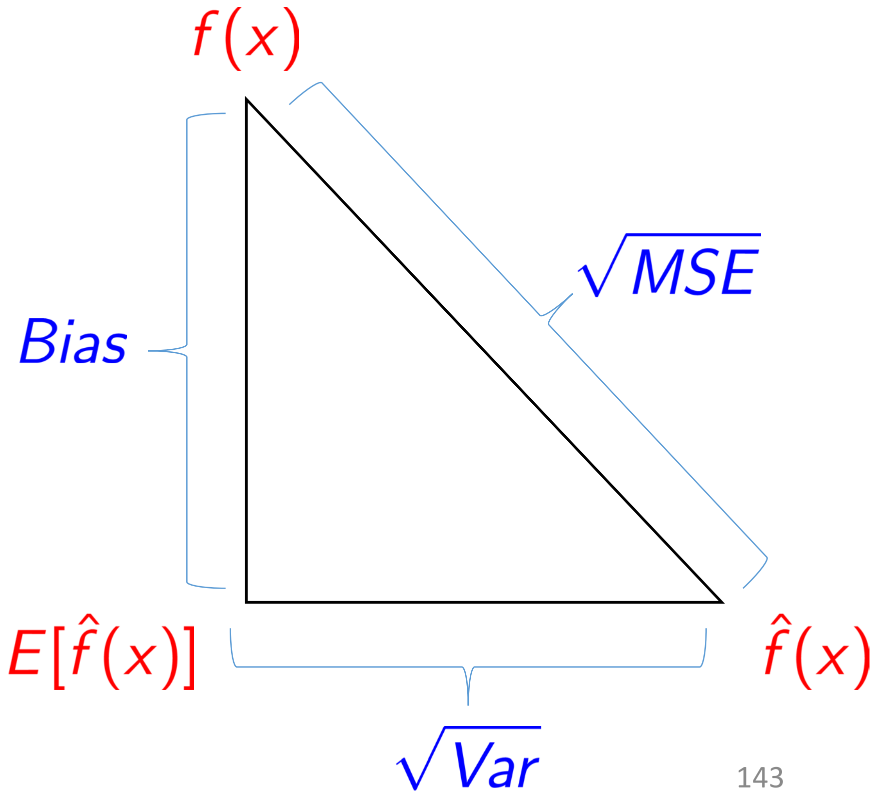
Model Selection

- The general task in machine learning is estimating a function.
  - We want to estimate :

$\hat{f}(x)$ estimated function

  - Where there is a true underlying function :

$f(x)$ true function



Definitions and Notations

assume $T = \{(x_i, y_i)\}_{i=1}^{n}$ be the training set.

Definitions and Notations

assume $T = \{(x_i, y_i)\}_{i=1}^{n}$ be the training set.

$f(.) \rightarrow$ True function

Definitions and Notations

assume $T = \{(x_i, y_i)\}_{i=1}^{n}$ be the training set.

$f(.) \rightarrow$ True function

$\hat{f}(.) \rightarrow$ Estimated function

Definitions and Notations

assume $T = \{(x_i, y_i)\}_{i=1}^{n}$ be the training set.

$f(.) \rightarrow$ True function

$\hat{f}(.) \rightarrow$ Estimated function

also assume:

$y_i = f(x_i) + \epsilon_i$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$

Definitions and Notations

assume $T = \{(x_i, y_i)\}_{i=1}^{n}$ be the training set.

$f(.) \rightarrow$ True function

$\hat{f}(.) \rightarrow$ Estimated function

also assume:

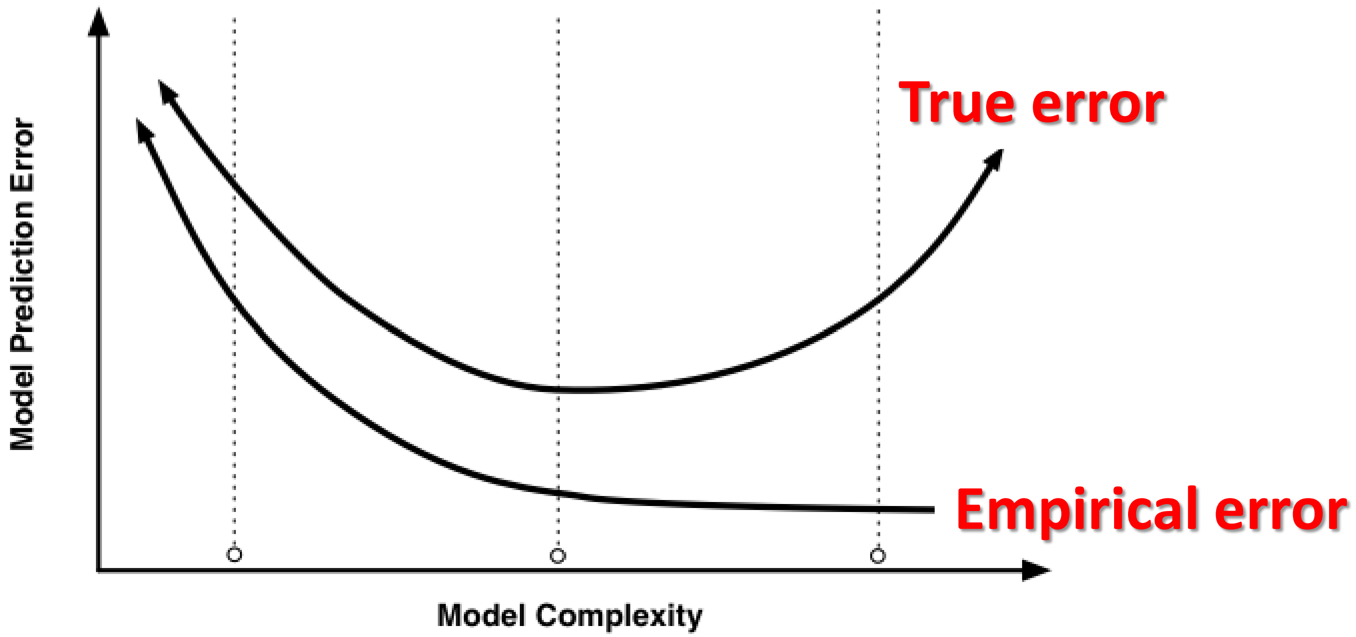
$y_i = f(x_i) + \epsilon_i$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$

$\hat{y}_i = \hat{f}(x_i)$

$f_i \equiv f(x_i)$
$\hat{f}_i \equiv \hat{f}(x_i)$

- Empirical error: $$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



For point (x_0, y_0) we are interested in:

$$E[(\hat{y}_0 - y_0)^2] = E[(\hat{f}_0 - f_0 - \epsilon_0)^2]$$

For point (x_0, y_0) we are interested in:

$$\begin{aligned} E[(\hat{y}_0 - y_0)^2] &= E[(\hat{f}_0 - f_0 - \epsilon_0)^2] \\ &= E\left[\left((\hat{f}_0 - f_0) - \epsilon_0\right)^2\right] \end{aligned}$$

For point (x_0, y_0) we are interested in:

$$
\begin{aligned}
E[(\hat{y}_0 - y_0)^2] &= E[(\hat{f}_0 - f_0 - \epsilon_0)^2] \\
&= E\left[\left((\hat{f}_0 - f_0) - \epsilon_0\right)^2\right] \\
&= E[(\hat{f}_0 - f_0)^2 + \epsilon_0^2 - 2\epsilon_0(\hat{f}_0 - f_0)]
\end{aligned}
$$

For point (x_0, y_0) we are interested in:

$$
\begin{aligned}
E[(\hat{y}_0 - y_0)^2] &= E[(\hat{f}_0 - f_0 - \epsilon_0)^2] \\
&= E\left[\left((\hat{f}_0 - f_0) - \epsilon_0\right)^2\right] \\
&= E[(\hat{f}_0 - f_0)^2 + \epsilon_0^2 - 2\epsilon_0(\hat{f}_0 - f_0)] \\
&= E[(\hat{f}_0 - f_0)^2] + E[\epsilon_0^2] - 2E[\epsilon_0(\hat{f}_0 - f_0)]
\end{aligned}
$$

For point (x_0, y_0) we are interested in:

$$
\begin{aligned}
E[(\hat{y}_0 - y_0)^2] &= E[(\hat{f}_0 - f_0 - \epsilon_0)^2] \\
&= E\left[\left((\hat{f}_0 - f_0) - \epsilon_0\right)^2\right] \\
&= E[(\hat{f}_0 - f_0)^2 + \epsilon_0^2 - 2\epsilon_0(\hat{f}_0 - f_0)] \\
&= E[(\hat{f}_0 - f_0)^2] + E[\epsilon_0^2] - 2E[\epsilon_0(\hat{f}_0 - f_0)] \\
&= E[(\hat{f}_0 - f_0)^2] + \sigma^2 - 2E[\epsilon_0(\hat{f}_0 - f_0)]
\end{aligned}
$$

case 1

assume: $(x_0, y_0) \notin T$

In this case, since $\hat{f}$ is estimated only based on points in training set, therefore it is completely independent from (x_0, y_0)

case 1

assume: $(x_0, y_0) \notin T$

In this case, since $\hat{f}$ is estimated only based on points in training set, therefore it is completely independent from (x_0, y_0)

$$\Rightarrow E[(y_0 - f)(\hat{f} - f)] = cov(y_0, \hat{f}_0) = 0$$

case 1

assume: $(x_0, y_0) \notin T$

In this case, since $\hat{f}$ is estimated only based on points in training set, therefore it is completely independent from (x_0, y_0)

$$\Rightarrow E[(y_0 - f)(\hat{f} - f)] = cov(y_0, \hat{f}_0) = 0$$

If summing up all m pints that are not in T.

case 1

assume: $(x_0, y_0) \notin T$

In this case, since $\hat{f}$ is estimated only based on points in training set, therefore it is completely independent from (x_0, y_0)

$$\Rightarrow E[(y_0 - f)(\hat{f} - f)] = cov(y_0, \hat{f}_0) = 0$$

If summing up all m pints that are not in T.

$$\underbrace{\sum_{i=1}^{m}(\hat{y}_i - y_i)^2}_{\text{err}} = \underbrace{\sum_{i=1}^{m}(\hat{f}_i - f_i)^2}_{\text{Err}} + m\sigma^2$$

true error **empirical error**

$$Err = err - m\sigma^2$$

true error

empirical error

$$Err = err - m\sigma^2$$

Empirical error (err) is a good estimator of true error (Err) if the point (x_0, y_0) is not in the training set.

case 2

assume: $(x_0, y_0) \in T$

then: $2E[\epsilon_0(\hat{f}_0 - f_0)] \neq 0$

case 2

assume: $(x_0, y_0) \in T$

then: $2E[\epsilon_0(\hat{f}_0 - f_0)] \neq 0$

Stein's Lemma

If :

$x \sim \mathcal{N}(\theta, \sigma^2)$
and $g(x)$ is differentiable.

then

$$E[g(x)(x - \theta)] = \sigma^2 E[\frac{\partial g(x)}{\partial x}]$$

- <u>Our problem:</u> $$E[\epsilon_0(\hat{f}_0 - f_0)] = \sigma^2 E[\frac{\partial(\hat{f}_0 - f_0)}{\partial \epsilon_0}]$$

- <u>Our problem:</u>

$$E[\epsilon_0(\hat{f}_0 - f_0)] = \sigma^2 E[\frac{\partial(\hat{f}_0 - f_0)}{\partial \epsilon_0}]$$
$$= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0} - \frac{\partial f_0}{\partial \epsilon_0}]$$

- <u>Our problem:</u>

$$E[\epsilon_0(\hat{f}_0 - f_0)] = \sigma^2 E[\frac{\partial(\hat{f}_0 - f_0)}{\partial \epsilon_0}]$$

$$= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0} - \cancelto{0}{\frac{\partial f_0}{\partial \epsilon_0}}]$$

- Our problem:

$$
\begin{aligned}
E[\epsilon_0(\hat{f}_0 - f_0)] &= \sigma^2 E[\frac{\partial(\hat{f}_0 - f_0)}{\partial \epsilon_0}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0} - \cancelto{0}{\frac{\partial f_0}{\partial \epsilon_0}}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0}]
\end{aligned}
$$

- <u>Our problem:</u>

$$
\begin{aligned}
E[\epsilon_0(\hat{f}_0 - f_0)] &= \sigma^2 E[\frac{\partial(\hat{f}_0 - f_0)}{\partial \epsilon_0}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0} - \cancelto{0}{\frac{\partial f_0}{\partial \epsilon_0}}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial y_0} \cdot \frac{\partial y_0}{\partial \epsilon_0}]
\end{aligned}
$$

- **Our problem:**

$$E[\epsilon_0(\hat{f}_0 - f_0)] = \sigma^2 E[\frac{\partial(\hat{f}_0 - f_0)}{\partial \epsilon_0}]$$

$$= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0} - \overbrace{\cancel{\frac{\partial f_0}{\partial \epsilon_0}}}^{0}]$$

$$= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0}]$$

Chain rule

$$= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial y_0} \cdot \frac{\partial y_0}{\partial \epsilon_0}]$$

- <u>Our problem:</u>

$$
\begin{aligned}
E[\epsilon_0(\hat{f}_0 - f_0)] &= \sigma^2 E[\frac{\partial(\hat{f}_0 - f_0)}{\partial \epsilon_0}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0} - \cancelto{0}{\frac{\partial f_0}{\partial \epsilon_0}}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial y_0} \cdot \cancelto{1}{\frac{\partial y_0}{\partial \epsilon_0}}]
\end{aligned}
$$

Chain rule

- Our problem:

$$
\begin{aligned}
E[\epsilon_0(\hat{f}_0 - f_0)] &= \sigma^2 E[\frac{\partial(\hat{f}_0 - f_0)}{\partial \epsilon_0}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0} - \cancelto{0}{\frac{\partial f_0}{\partial \epsilon_0}}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial y_0} \cdot \cancelto{1}{\frac{\partial y_0}{\partial \epsilon_0}}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial y_0}]
\end{aligned}
$$

Chain rule

- **Our problem:**

$$
\begin{aligned}
E[\epsilon_0(\hat{f}_0 - f_0)] &= \sigma^2 E[\frac{\partial(\hat{f}_0 - f_0)}{\partial \epsilon_0}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0} - \cancelto{0}{\frac{\partial f_0}{\partial \epsilon_0}}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial \epsilon_0}] \\
&= \sigma^2 E[\frac{\partial \hat{f}_0}{\partial y_0} \cdot \cancelto{1}{\frac{\partial y_0}{\partial \epsilon_0}}] \\
&= \sigma^2 E[\underbrace{\frac{\partial \hat{f}_0}{\partial y_0}}_{D_0}]
\end{aligned}
$$

Chain rule

$$E[(\hat(y)_0 - y_0)^2] = E[(\hat(f)_0 - f_0)^2] + \sigma^2 - 2\sigma^2 E[D_0]$$

$$E[(\hat(y)_0 - y_0)^2] = E[(\hat(f)_0 - f_0)^2] + \sigma^2 - 2\sigma^2 E[D_0]$$

Sum over all n data points:

$$\sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{n} (\hat{f}_i - f_i)^2 + n\sigma^2 - 2\sigma^2 \sum_{i=1}^{n} D_i$$

$$E[(\hat{(y)}_0 - y_0)^2] = E[(\hat{(f)}_0 - f_0)^2] + \sigma^2 - 2\sigma^2 E[D_0]$$

Sum over all n data points:

$$\underbrace{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}_{\text{err}} = \sum_{i=1}^{n} (\hat{f}_i - f_i)^2 + n\sigma^2 - 2\sigma^2 \sum_{i=1}^{n} D_i$$

$$E[(\hat{(y)}_0 - y_0)^2] = E[(\hat{(f)}_0 - f_0)^2] + \sigma^2 - 2\sigma^2 E[D_0]$$

Sum over all n data points:

$$\underbrace{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}_{\text{err}} = \underbrace{\sum_{i=1}^{n}(\hat{f}_i - f_i)^2}_{\text{Err}} + n\sigma^2 - 2\sigma^2 \sum_{i=1}^{n} D_i$$

$$E[(\hat{(y)}_0 - y_0)^2] = E[(\hat{(f)}_0 - f_0)^2] + \sigma^2 - 2\sigma^2 E[D_0]$$

Sum over all n data points:

$$\underbrace{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}_{\text{err}} = \underbrace{\sum_{i=1}^{n} (\hat{f}_i - f_i)^2}_{\text{Err}} + n\sigma^2 - 2\sigma^2 \sum_{i=1}^{n} D_i$$

$$Err = err - n\sigma^2 + 2\sigma^2 \sum_{i=1}^{n} D_i$$

$$E[(\hat(y)_0 - y_0)^2] = E[(\hat(f)_0 - f_0)^2] + \sigma^2 - 2\sigma^2 E[D_0]$$

Sum over all n data points:

$$\underbrace{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}_{\text{err}} = \underbrace{\sum_{i=1}^{n}(\hat{f}_i - f_i)^2}_{\text{Err}} + n\sigma^2 - 2\sigma^2 \sum_{i=1}^{n} D_i$$

$$Err = err - n\sigma^2 + 2\sigma^2 \underbrace{\sum_{i=1}^{n} D_i}_{\text{Complexity of model}}$$

$$E[(\hat{(y)}_0 - y_0)^2] = E[(\hat{(f)}_0 - f_0)^2] + \sigma^2 - 2\sigma^2 E[D_0]$$

Sum over all n data points:

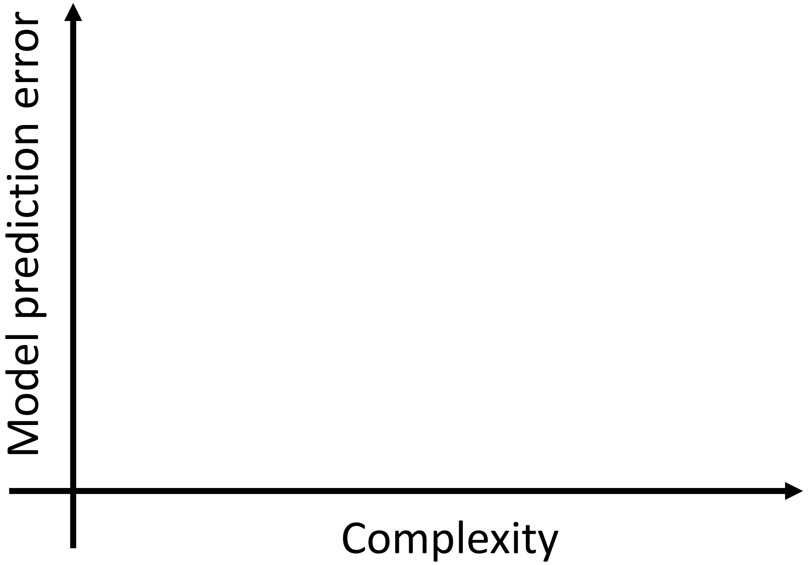
$$\underbrace{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}_{\text{err}} = \underbrace{\sum_{i=1}^{n}(\hat{f}_i - f_i)^2}_{\text{Err}} + n\sigma^2 - 2\sigma^2 \sum_{i=1}^{n} D_i$$

$$\boxed{Err = err - n\sigma^2 + \underbrace{2\sigma^2 \sum_{i=1}^{n} D_i}_{\text{Complexity of model}}}$$

Stein's Unbiased Risk Estimator (SURE)

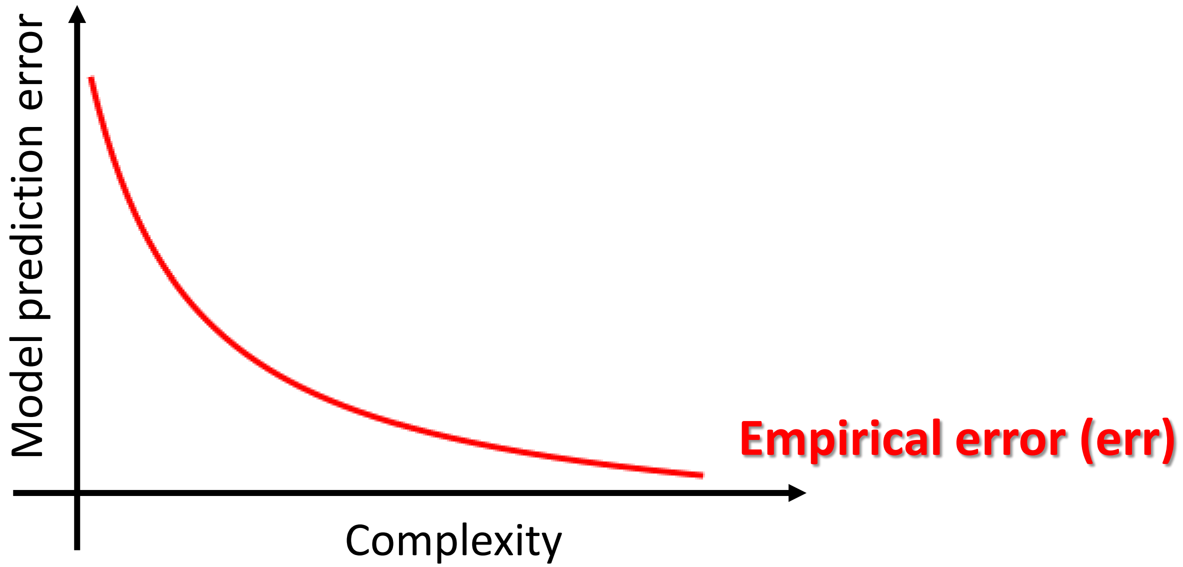
- SURE gives us a very good insight about the behavior of true error

$$Err = err - n\sigma^2 + 2\sigma^2 \sum_{i=1}^{n} D_i$$



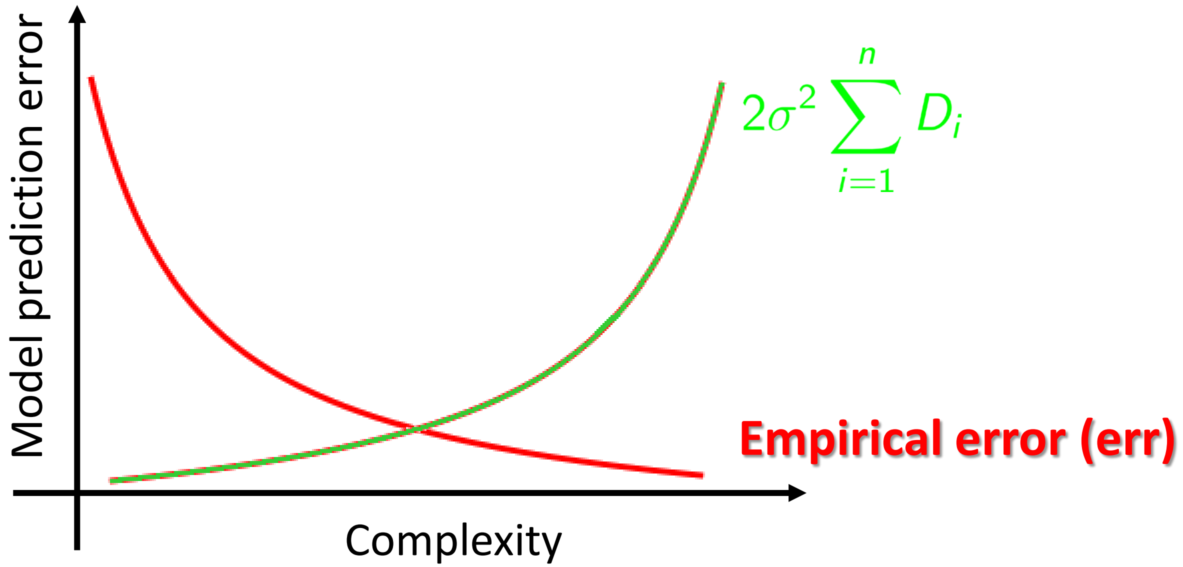
- SURE gives us a very good insight about the behavior of true error

$$Err = err - n\sigma^2 + 2\sigma^2 \sum_{i=1}^{n} D_i$$



- SURE gives us a very good insight about the behavior of true error

$$Err = err - n\sigma^2 + 2\sigma^2 \sum_{i=1}^{n} D_i$$



- SURE gives us a very good insight about the behavior of true error

$$Err = err - n\sigma^2 + 2\sigma^2 \sum_{i=1}^{n} D_i$$