PAC-Bayesian Learning of Optimization Algorithms

Peter Ochs
Mathematical Optimization for Data Science
Saarland University
— 07.09.2023 —

joint work: Michael Sucker
Inverse Problems are often Modelled as an Optimization Problem

Example 1:

\[
\min_x f(x), \quad f(x) := \frac{1}{2} \|Ax - b\|^2.
\]
Inverse Problems are often Modelled as an Optimization Problem

Example 1:

$$\min_x f(x), \quad f(x) := \frac{1}{2} \|Ax - b\|^2.$$ 

How do we solve the problem?
Inverse Problems are often Modelled as an Optimization Problem

Example 1:

$$\min_x f(x), \quad f(x) := \frac{1}{2} \|Ax - b\|^2.$$ 

How do we solve the problem?

- Inspect the properties of the problem.

*Example: Smooth/Quadratic problem with $L = \|A\|^2$-Lipschitz gradient.*
Inverse Problems are often Modelled as an Optimization Problem

Example 1:

$$\min_x f(x), \quad f(x) := \frac{1}{2} \|Ax - b\|^2.$$  

How do we solve the problem?

- Inspect the properties of the problem.
  
  **Example:** Smooth/Quadratic problem with $L = \|A\|^2$-Lipschitz gradient.

- Embed the problem into a **class of problems** for which algorithms are available.
  
  **Example:** Use Gradient Descent with step size $\alpha = 1/L$

  $$x^{(k+1)} = x^{(k)} - \alpha \nabla f(x^{(k)}).$$

  **Worst case convergence guarantee:**

  $$f(x^{(k)}) - \min f \leq O(1/k).$$
Hidden Structures

If we knew ...

\[
\begin{pmatrix}
1 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1
\end{pmatrix}
\]

We would write down a different algorithm (that directly returns the solution).
If we knew ...

that $A$ is actually of the form

$$A = \begin{pmatrix}
10 & 0 & \cdots & 0 \\
0 & 1 & \ddots & \\
\vdots & \ddots & \ddots & 0 \\
0 & \cdots & 0 & 1
\end{pmatrix}$$

We would write down a different algorithm (that directly returns the solution).
If we knew ...

- that $A$ is actually of the form

$$
A = \begin{pmatrix}
1 & 0 & \cdots & 0 \\
0 & 1 & \ddots & \vdots \\
\vdots & \ddots & \ddots & 0 \\
0 & \cdots & 0 & 1
\end{pmatrix}
$$

- We would write down a different algorithm (that directly returns the solution).

**Game-changer**, if many such problems for different $b$ need to be solved.

- Sometimes the "best" class of problems is not obvious!
If we knew ...

- that $A$ is actually of the form

$$A = \begin{pmatrix} 10 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

- We would write down a different algorithm (that directly returns the solution).

**Game-changer**, if many such problems for different $b$ need to be solved.

- Sometimes the “best” class of problems is not obvious!

Can we construct an algorithm that adapts to hidden problem structures?
Example 2: Solve many problems of the form

$$\min_x f_A(x), \quad f_A(x) := \frac{1}{2} \|Ax - b\|^2 \quad \text{where} \quad A = \bar{A} + \text{noise}.$$
Example 2: Solve many problems of the form

$$\min_x f_A(x), \quad f_A(x) := \frac{1}{2} \|Ax - b\|^2 \quad \text{where} \quad A = \bar{A} + \text{noise}.$$ 

Using Gradient Descent:

- For each problem $f_A$, we need to compute $L = \|A\|^2$,
- and run Gradient Descent with $\alpha = 1/L$ to solve the problem.

Computation of $\|A\|$ can be expensive.
Example 2: Solve many problems of the form

$$\min_x f_A(x), \quad f_A(x) := \frac{1}{2} \| Ax - b \|^2$$

where $$A = \bar{A} + \text{noise}$$.

**Using Gradient Descent:**
- For each problem $$f_A$$, we need to compute $$L = \| A \|^2$$,
- and run Gradient Descent with $$\alpha = 1/L$$ to solve the problem.

- Computation of $$\| A \|$$ can be expensive.

Or ...
- if the noise is bounded, we can use a worst case estimate for $$L$$.

- Results in **small step sizes**.
- Upper bound may be **too pessimistic for most problems** in practice.
Example 2: Solve many problems of the form
\[
\min_x f_A(x), \quad f_A(x) := \frac{1}{2} \|Ax - b\|^2 \quad \text{where} \quad A = \bar{A} + \text{noise}.
\]

Using Gradient Descent:
- For each problem \(f_A\), we need to compute \(L = \|A\|^2\),
- and run Gradient Descent with \(\alpha = 1/L\) to solve the problem.

Computation of \(\|A\|\) can be expensive.

Or ...
- if the noise is bounded, we can use a worst case estimate for \(L\).

Results in small step sizes.
- Upper bound may be too pessimistic for most problems in practice.

Can we construct an algorithm with good performance for more likely problems?
Data Driven Approach

Yes, using data driven approaches / learning!
Yes, using data driven approaches / learning!

Learning alleviates the bounds of analytical tractability by providing more:

- **Information**: Leverage more structure.
- **Automation**: Less “hand-crafting”.
- **Possibilities**: More building blocks.
Data Driven Approach

Yes, using data driven approaches / learning!

Learning alleviates the bounds of analytical tractability by providing more:

- **Information:** Leverage more structure.
- **Automation:** Less “hand-crafting”.
- **Possibilities:** More building blocks.

Our goals:

- Breaking the barrier of worst-case estimates.
- Adapt algorithms to hidden problem structures.
- Define tight classes of problems.
- We insist on having some theoretical guarantees.
Consider the random parametric optimization problem:

\[
\min_{x \in \mathbb{R}^n} \ell(x, \mathcal{S})
\]

- \(\ell : \mathbb{R}^n \times \Theta \to \mathbb{R}_{\geq 0}\) is a given measurable loss-function.
- \(\mathcal{S} : (\Omega, \mathcal{F}, \mathbb{P}) \to \Theta\) is a random variable.
Consider the random parametric optimization problem:

$$\min_{x \in \mathbb{R}^n} \ell(x, \mathcal{G})$$

- $\ell : \mathbb{R}^n \times \Theta \rightarrow \mathbb{R}_{\geq 0}$ is a given measurable loss-function.
- $\mathcal{G} : (\Omega, \mathcal{F}, \mathbb{P}) \rightarrow \Theta$ is a random variable.

Example:

Regularized Inverse Problem:

$$\ell(x, \lambda) = \frac{1}{2} \|Ax - b\|^2 + \lambda R(x), \quad \text{i.e. } \theta := \lambda, \Theta = [0, 1].$$
Formalize the Problem

Consider the random parametric optimization problem:

\[
\min_{x \in \mathbb{R}^n} \ell(x, \mathcal{G})
\]

- \( \ell : \mathbb{R}^n \times \Theta \rightarrow \mathbb{R}_{\geq 0} \) is a given measurable loss-function.
- \( \mathcal{G} : (\Omega, \mathcal{F}, \mathbb{P}) \rightarrow \Theta \) is a random variable.

Use a parametric optimization algorithm, i.e. a measurable function:

\[
\mathcal{A} : \mathcal{H} \times \mathbb{R}^n \times \Theta \rightarrow \mathbb{R}^n, \quad (\alpha, x^{(0)}, \theta) \mapsto \mathcal{A}(\alpha, x^{(0)}, \theta)
\]

Example:

- Regularized Inverse Problem:

\[
\ell(x, \lambda) = \frac{1}{2} \|Ax - b\|^2 + \lambda R(x), \quad \text{i.e.} \quad \theta := \lambda, \Theta = [0, 1].
\]
Formalize the Problem

Consider the random parametric optimization problem:

\[
\min_{x \in \mathbb{R}^n} \ell(x, \mathcal{G})
\]

- \( \ell : \mathbb{R}^n \times \Theta \rightarrow \mathbb{R}_{\geq 0} \) is a given measurable loss-function.
- \( \mathcal{G} : (\Omega, \mathcal{F}, \mathbb{P}) \rightarrow \Theta \) is a random variable.

Use a parametric optimization algorithm, i.e. a measurable function:

\[
\mathcal{A} : \mathcal{H} \times \mathbb{R}^n \times \Theta \rightarrow \mathbb{R}^n, \quad (\alpha, x^{(0)}, \theta) \mapsto \mathcal{A}(\alpha, x^{(0)}, \theta)
\]

Example:

- Regularized Inverse Problem:
  \[
  \ell(x, \lambda) = \frac{1}{2} \|Ax - b\|^2 + \lambda R(x), \quad \text{i.e. } \theta := \lambda, \Theta = [0, 1].
  \]

- Concatenation of a fixed number of Preconditioned Gradient Descent steps:
  \[
  x^{(k+1)} = x^{(k)} - P \nabla \ell(x^{(k)}, \theta), \quad \text{i.e. } \alpha := P, \mathcal{H} := \mathbb{R}^{n \times n}.
  \]
Formalize the Problem

Consider the random parametric optimization problem:

\[ \min_{x \in \mathbb{R}^n} \ell(x, G) \]

- \( \ell : \mathbb{R}^n \times \Theta \rightarrow \mathbb{R}_{\geq 0} \) is a given measurable loss-function.
- \( G : (\Omega, \mathcal{F}, P) \rightarrow \Theta \) is a random variable.

Use a parametric optimization algorithm, i.e. a measurable function:

\[ A : \mathcal{H} \times \mathbb{R}^n \times \Theta \rightarrow \mathbb{R}^n, \quad (\alpha, x^{(0)}, \theta) \mapsto A(\alpha, x^{(0)}, \theta) \]

Example:

- Regularized Inverse Problem:
  \[ \ell(x, \lambda) = \frac{1}{2} \|Ax - b\|^2 + \lambda R(x), \quad \text{i.e. } \theta := \lambda, \Theta = [0, 1]. \]

- Concatenation of a fixed number of Preconditioned Gradient Descent steps:
  \[ x^{(k+1)} = x^{(k)} - P \nabla \ell(x^{(k)}, \theta), \quad \text{i.e. } \alpha := P, \mathcal{H} := \mathbb{R}^{n \times n}. \]

Learning boils down to hyperparameter optimization, i.e. how to choose \( \alpha \in \mathcal{H}. \)
**Deterministic/Analytic Approach:** Worst case performance

\[
\min_{\alpha \in \mathcal{H}} \sup_{\theta \in \Theta} \ell(A(\alpha, \theta), \theta). 
\]

Only possible for certain classes of problems.
Quest for Theoretical Convergence Guarantees

**Deterministic/Analytic Approach:** Worst case performance

\[
\min_{\alpha \in \mathcal{H}} \sup_{\theta \in \Theta} \ell(A(\alpha, \theta), \theta).
\]

Only possible for certain classes of problems.

**Learning Based Approach:** Expected case performance:

- **Minimize the risk** \( R(\alpha) \), defined as the **expected loss**:

\[
\min_{\alpha \in \mathcal{H}} R(\alpha), \quad R(\alpha) := \mathbb{E}[\ell(A(\alpha, S), S)].
\]
Deterministic/Analytic Approach: Worst case performance

\[
\min_{\alpha \in \mathcal{H}} \sup_{\theta \in \Theta} \ell(A(\alpha, \theta), \theta).
\]

Only possible for certain classes of problems.

Learning Based Approach: Expected case performance:

- Minimize the risk \( R(\alpha) \), defined as the expected loss:

\[
\min_{\alpha \in \mathcal{H}} R(\alpha), \quad R(\alpha) := \mathbb{E}[\ell(A(\alpha, S), S)].
\]

This is intractable, since the distribution \( \mathbb{P}_S \) is unknown.
Quest for Theoretical Convergence Guarantees

**Deterministic/Analytic Approach**: Worst case performance

\[
\min_{\alpha \in \mathcal{H}} \sup_{\theta \in \Theta} \ell(A(\alpha, \theta), \theta).
\]

Only possible for certain classes of problems.

**Learning Based Approach**: Expected case performance:

- Minimize the **risk** \( R(\alpha) \), defined as the **expected loss**:
  \[
  \min_{\alpha \in \mathcal{H}} R(\alpha), \quad R(\alpha) := \mathbb{E}[\ell(A(\alpha, S), S)].
  \]

  This is intractable, since the distribution \( P_S \) is unknown.

- Hence, resort to minimizing the **empirical risk** \( \hat{R}(\alpha, \mathcal{D}_N) \) over some dataset \( \mathcal{D}_N := \{ S_i \}_{i=1}^N \):
  \[
  \min_{\alpha \in \mathcal{H}} \hat{R}(\alpha, \mathcal{D}_N), \quad \hat{R}(\alpha, \mathcal{D}_N) := \frac{1}{N} \sum_{i=1}^N \ell(A(\alpha, S_i), S_i).
  \]
Why do we need Generalization Guarantees?

Is the performance on $\hat{R}$ representative for the overall performance $R$?

Yes, if we have uniform generalization bounds, i.e. bounds of the form:

$$\forall \epsilon > 0: P_{\hat{R}}(\alpha^*(D_N)) \leq \inf_{\alpha \in H} \hat{R}(\alpha, D_N) + K(N, \alpha, \epsilon) \geq 1 - \epsilon.$$ 

Such bounds are called PAC-bounds, which is an acronym for: Probably With high probability, Approximately the empirical risk is close to Correct the true risk.
Why do we need Generalization Guarantees?

Is the performance on $\hat{R}$ representative for the overall performance $R$?

Yes, if we have uniform generalization bounds, i.e. bounds of the form: $\forall \epsilon > 0$:

$$
P\{ R(\alpha^* (\mathcal{D}_N)) \leq \inf_{\alpha \in \mathcal{H}} \hat{R}(\alpha, \mathcal{D}_N) + K(N, \alpha, \epsilon) \} \geq 1 - \epsilon .$$
Why do we need Generalization Guarantees?

Is the performance on $\hat{R}$ representative for the overall performance $R$?

Yes, if we have **uniform generalization bounds**, i.e. bounds of the form: $\forall \varepsilon > 0$:

$$
P\{ \mathcal{R}(\alpha^*(\mathcal{D}_N)) \leq \inf_{\alpha \in \mathcal{H}} \hat{R}(\alpha, \mathcal{D}_N) + K(N, \alpha, \varepsilon) \} \geq 1 - \varepsilon .$$

Such bounds are called **PAC-bounds**, which is an acronym for: 

- **P**robably
- **A**proximately
- **C**orrect
Why do we need Generalization Guarantees?

Is the performance on $\hat{\mathcal{R}}$ representative for the overall performance $\mathcal{R}$?

Yes, if we have **uniform generalization bounds**, i.e. bounds of the form: $\forall \epsilon > 0$:

$$
P\left\{ \mathcal{R}(\alpha^*(\mathcal{D}_N)) \leq \inf_{\alpha \in \mathcal{H}} \hat{\mathcal{R}}(\alpha, \mathcal{D}_N) + K(N, \alpha, \epsilon) \right\} \geq 1 - \epsilon.
$$

Such bounds are called **PAC-bounds**, which is an acronym for: Probably \(\{\text{z}\}\) With high probability, Approximately \(\{\text{z}\}\) the empirical risk is close to Correct \(\{\text{z}\}\) the true risk.
Why do we need Generalization Guarantees?

Is the performance on $\hat{\mathcal{R}}$ representative for the overall performance $\mathcal{R}$?

Yes, if we have **uniform generalization bounds**, i.e. bounds of the form: $\forall \epsilon > 0$:

$$
\mathbb{P}\{ \mathcal{R}(\alpha^*(\mathcal{D}_N)) \leq \inf_{\alpha \in \mathcal{H}} \hat{\mathcal{R}}(\alpha, \mathcal{D}_N) + K(N, \alpha, \epsilon) \} \geq 1 - \epsilon.
$$

Such bounds are called **PAC-bounds**, which is an acronym for:

- **P**robably
- **A**dditively
- **C**orrect
Why do we need Generalization Guarantees?

Is the performance on $\hat{R}$ representative for the overall performance $R$?

- Yes, if we have **uniform generalization bounds**, i.e. bounds of the form: $\forall \epsilon > 0$:

  $\mathbb{P}\{R(\alpha^*(\mathcal{D}_N)) \leq \inf_{\alpha \in \mathcal{H}} \hat{R}(\alpha, \mathcal{D}_N) + K(N, \alpha, \epsilon)\} \geq 1 - \epsilon$.

- Such bounds are called **PAC-bounds**, which is an acronym for:

  $$\text{Probably} \quad \text{Approximately} \quad \text{Correct}.$$  

  With high probability, the empirical risk is close to the true risk.
PAC-Bayes extends this to the Bayes-risk:

Such bounds hold for **posterior distributions** \( Q \in \mathcal{M}(\mathbb{P}_f) \):

\[
\mathbb{P}\left\{ \mathbb{E}_{Q^*}(\mathcal{D}_N)[\mathcal{R}] \leq \inf_{Q \in \mathcal{M}(\mathbb{P}_f)} \mathbb{E}_Q[\hat{\mathcal{R}}(\mathcal{D}_N)] + K(Q, N, \epsilon) \right\} \geq 1 - \epsilon ,
\]

where \( \mathcal{M}(\mathbb{P}_f) \) denotes some class of (probability) measures on \( \mathcal{H} \) that satisfy a certain property w.r.t. the **prior distribution** \( \mathbb{P}_f \).

*This is a naming convention! Not to be confused with prior and posterior in Bayesian analysis, which are linked by a likelihood.*
For good reviews of two long lines of work see...

PAC-Bayes [Alquier ’21]

Learning-to-Optimize [Chen et al. ’22]

PAC-Bayesian Learning of Optimization Algorithms [Sucker, O. 22]

Lemma: Consider an exponential family \((Q_\lambda)_{\lambda \in \Lambda}\) w.r.t. the prior \(P_S\), i.e. distributions of the form:

\[ Q_\lambda \propto \exp(\langle \eta(\lambda), T \rangle) \cdot P_S, \quad \lambda \in \Lambda \]

and denote \(c(\lambda) := \mathbb{E}_{P_S}[\exp(\langle \eta(\lambda), T \rangle)]\). Then it holds:

\[ \log(c(\lambda)) = \sup_{Q \ll P_S} \mathbb{E}_Q[\langle \eta(\lambda), T \rangle] - D_{KL}(Q \parallel P_S) \]

Furthermore, for every \(\lambda \in \Lambda\), the supremum is attained at \(Q_\lambda\).
**Lemma:** Consider an exponential family \( (Q_{\lambda})_{\lambda \in \Lambda} \) w.r.t. the prior \( P_\mathcal{F} \), i.e. distributions of the form:

\[
Q_\lambda \propto \exp(\langle \eta(\lambda), T \rangle) \cdot P_\mathcal{F}, \quad \lambda \in \Lambda
\]

and denote \( c(\lambda) := \mathbb{E}_{P_\mathcal{F}} [\exp(\langle \eta(\lambda), T \rangle)] \). Then it holds:

\[
\log(c(\lambda)) = \sup_{Q \ll P_\mathcal{F}} \mathbb{E}_Q [\langle \eta(\lambda), T \rangle] - D_{KL}(Q \parallel P_\mathcal{F})
\]

Furthermore, for every \( \lambda \in \Lambda \), the supremum is attained at \( Q_\lambda \).
Lemma: Consider an exponential family \((Q_\lambda)_{\lambda \in \Lambda}\) w.r.t. the prior \(P_\delta\), i.e. distributions of the form:

\[
Q_\lambda \propto \exp(\langle \eta(\lambda), T \rangle) \cdot P_\delta, \quad \lambda \in \Lambda
\]

and denote \(c(\lambda) := E_{P_\delta} \left[ \exp(\langle \eta(\lambda), T \rangle) \right]\). Then it holds:

\[
\log(c(\lambda)) = \sup_{Q \ll P_\delta} E_Q[\langle \eta(\lambda), T \rangle] - D_{KL}(Q \parallel P_\delta)
\]

Furthermore, for every \(\lambda \in \Lambda\), the supremum is attained at \(Q_\lambda\).
Towards a PAC-Bayes Theorem for Exponential Families

A Form of the Donsker–Varadhan Variational Formulation

**Lemma:** Consider an exponential family $(Q_\lambda)_{\lambda \in \Lambda}$ w.r.t. the prior $\mathbb{P}_S$, i.e. distributions of the form:

$$Q_\lambda \propto \exp(\langle \eta(\lambda), T \rangle) \cdot \mathbb{P}_S, \quad \lambda \in \Lambda$$

and denote $c(\lambda) := \mathbb{E}_{\mathbb{P}_S} \left[ \exp(\langle \eta(\lambda), T \rangle) \right]$. Then it holds:

$$\log(c(\lambda)) = \sup_{Q \ll \mathbb{P}_S} \mathbb{E}_Q[\langle \eta(\lambda), T \rangle] - D_{KL}(Q \parallel \mathbb{P}_S)$$

Furthermore, for every $\lambda \in \Lambda$, the supremum is attained at $Q_\lambda$. 

© 2023 — Peter Ochs

Mathematical Optimization for Data Science
Theorem: If $\mathbb{E}_{\mathcal{D}_N} [c(\lambda)] \leq 1$, then for all $\varepsilon > 0$:

$$\mathbb{P}_{\mathcal{D}_N} \left\{ \forall \lambda \in \Lambda: \forall Q \ll P_\mathcal{H}: \mathbb{E}_Q [\langle \eta(\lambda), T \rangle] \leq D_{KL}(Q \parallel P_\mathcal{H}) + \log(|\Lambda|/\varepsilon) \right\} \geq 1 - \varepsilon$$
Towards a PAC-Bayes Theorem for Exponential Families

Theorem: If $\mathbb{E}_{\mathcal{D}} \left[ c(\lambda) \right] \leq 1$, then for all $\varepsilon > 0$:

$$\mathbb{P}_{\mathcal{D}} \{ \forall \lambda \in \Lambda : \forall Q \ll \mathbb{P}_S : \mathbb{E}_Q \left[ \langle \eta(\lambda), T \rangle \right] \leq D_{\text{KL}}(Q \parallel \mathbb{P}_S) + \log(|\Lambda|/\varepsilon) \} \geq 1 - \varepsilon$$

Sketch of Proof.

- Use Markov's inequality

$$\mathbb{P}_{\mathcal{D}} \left\{ c(\lambda) \geq \exp(s) \right\} \leq \frac{\mathbb{E}_{\mathcal{D}} \left[ c(\lambda) \right]}{\exp(s)} \leq 1/\exp(s) =: 1/s'.$$
Towards a PAC-Bayes Theorem for Exponential Families

**Theorem:** If $\mathbb{E}_{\mathcal{D}_N}[c(\lambda)] \leq 1$, then for all $\varepsilon > 0$:

$$\mathbb{P}_{\mathcal{D}_N}\{\forall \lambda \in \Lambda : \forall Q \ll P_{\mathcal{F}} : \mathbb{E}_Q[\langle \eta(\lambda), T \rangle] \leq D_{KL}(Q \parallel P_{\mathcal{F}}) + \log(|\Lambda|/\varepsilon)\} \geq 1 - \varepsilon$$

**Sketch of Proof.**

- Use Markov's inequality

  $$\mathbb{P}_{\mathcal{D}_N}\{c(\lambda) \geq \exp(s)\} \leq \frac{\mathbb{E}_{\mathcal{D}_N}[c(\lambda)]}{\exp(s)} \leq 1/\exp(s) =: 1/s'.$$

- Union-bound argument: (use covering argument for compact continuous $\Lambda$)

  $$\mathbb{P}_{\mathcal{D}_N}\{\sup_{\lambda \in \Lambda} c(\lambda) > s'\} = \mathbb{P}_{\mathcal{D}_N}\{\bigcup_{\lambda \in \Lambda} \{c(\lambda) > s'\}\} \leq \sum_{\lambda \in \Lambda} \mathbb{P}_{\mathcal{D}_N}\{\{c(\lambda) > s'\}\} \leq |\Lambda|/s' =: \varepsilon.$$
Towards a PAC-Bayes Theorem for Exponential Families

**Theorem:** If \( \mathbb{E}_{\mathcal{D}_N} [c(\lambda)] \leq 1 \), then for all \( \varepsilon > 0 \):

\[
P_{\mathcal{D}_N} \{ \forall \lambda \in \Lambda : \forall Q \ll P_{\mathcal{F}} : \mathbb{E}_Q [\langle \eta(\lambda), T \rangle] \leq D_{KL}(Q \parallel P_{\mathcal{F}}) + \log(|\Lambda|/\varepsilon) \} \geq 1 - \varepsilon
\]

**Sketch of Proof.**

- **Use Markov’s inequality**

  \[
P_{\mathcal{D}_N} \{ c(\lambda) \geq \exp(s) \} \leq \frac{\mathbb{E}_{\mathcal{D}_N} [c(\lambda)]}{\exp(s)} \leq 1/\exp(s) =: 1/s'.
  \]

- **Union-bound argument:** (use covering argument for compact continuous \( \Lambda \))

  \[
P_{\mathcal{D}_N} \{ \sup_{\lambda \in \Lambda} c(\lambda) > s' \} = P_{\mathcal{D}_N} \{ \bigcup_{\lambda \in \Lambda} \{ c(\lambda) > s' \} \} \leq \sum_{\lambda \in \Lambda} P_{\mathcal{D}_N} \{ \{ c(\lambda) > s' \} \} \leq |\Lambda|/s' =: \varepsilon
  \]

- **Apply Donsker–Varadhan variational formulation in**

  \[
P_{\mathcal{D}_N} \{ \sup_{\lambda \in \Lambda} \log(c(\lambda)) \leq \log(|\Lambda|/\varepsilon) \} \geq 1 - \varepsilon.
  \]
Specify $\eta$ and $T$ accordingly to construct a PAC-Bayesian generalization bound:

$$\eta(\lambda) = (\eta_1(\lambda), \eta'(\lambda))$$

and

$$T(\alpha, \mathcal{D}_N) = (\mathcal{R}(\alpha) - \hat{\mathcal{R}}(\alpha, \mathcal{D}_N), T'(\alpha, \mathcal{D}_N)).$$
Specify $\eta$ and $T$ accordingly to construct a PAC-Bayesian generalization bound:

$$\eta(\lambda) = (\eta_1(\lambda), \eta'(\lambda))$$

and

$$T(\alpha, \mathcal{D}_N) = (\mathcal{R}(\alpha) - \hat{\mathcal{R}}(\alpha, \mathcal{D}_N), T'(\alpha, \mathcal{D}_N)).$$

This provides a bound of the following form:

**Theorem:** Under mild assumptions, it holds for $\epsilon > 0$:

$$\mathbb{P}_{\mathcal{D}_N} \left\{ \forall \lambda \in \Lambda, \forall Q \ll \mathbb{P}_Y : \mathbb{E}_Q[\mathcal{R}] \leq \mathbb{E}_Q[\hat{\mathcal{R}}] + G(N, \lambda, Q, \epsilon) \right\} \geq 1 - \epsilon.$$

**Note:**

(i) By the definition of the risk and the algorithm, this bound gives a guarantee for the function value of the algorithm’s output.

(ii) This is a statement about relative values, not absolute ones.
Specify $\eta$ and $T$ accordingly to construct a PAC-Bayesian generalization bound:

$$\eta(\lambda) = (\eta_1(\lambda), \eta'(\lambda))$$

and

$$T(\alpha, D_N) = (R(\alpha) - \hat{R}(\alpha, D_N), T'(\alpha, D_N)).$$

This provides a bound of the following form:

**Theorem:** Under mild assumptions, it holds for $\epsilon > 0$:

$$\mathbb{P}_{D_N} \left\{ \forall \lambda \in \Lambda, \forall Q \ll P_f : \mathbb{E}_Q[R] \leq \mathbb{E}_Q[\hat{R}] + G(N, \lambda, Q, \epsilon) \right\} \geq 1 - \epsilon.$$

**Note:**

(i) By the definition of the risk and the algorithm, this bound gives a guarantee for the function value of the algorithm’s output.

(ii) This is a statement about relative values, not absolute ones.

$\Rightarrow$ Since supremum is attained at $Q_\lambda$, learning can be phrased as an optimization in $\lambda$ (possibly very low-dimensional).
**Issue:** A bad performance on a single problem dominates the average.

- Sometimes, analytic worst-case bounds are sharp.
- Gradient Descent on quadratics diverges for $\alpha > 2/L$.
- Trying to learn the step size (without this bound) yields an extremely large loss for $\alpha > 2/L$, which dominates the cost of the “average performance” (the empirical risk). Therefore, learnable step sizes obey the deterministic step size rule $\alpha \in (0, 2/L)$.

In this case, the analytically best known step size is recovered by learning.
However this might correspond to unlikely special/extreme cases:

We develop a variant that allows to **Trade-Off Guarantees and Speed**.
Trade-Off Guarantees and Speed

However this might correspond to unlikely special/extreme cases:

We develop a variant that allows to **Trade-Off Guarantees and Speed**.

*The algorithm may diverge (for extreme cases), if this happens in rare cases and the Trade-Off can be controlled.*
Trade-Off Guarantees and Speed

However this might correspond to unlikely special/extreme cases:

We develop a variant that allows to Trade-Off Guarantees and Speed.

The algorithm may diverge (for extreme cases), if this happens in rare cases and the Trade-Off can be controlled.

Account for likelihood of e.g. the worst-case.
However this might correspond to unlikely special/extreme cases:

We develop a variant that allows to **Trade-Off Guarantees and Speed**.

*The algorithm may diverge (for extreme cases), if this happens in rare cases and the Trade-Off can be controlled.*

- Account for **likelihood** of e.g. the worst-case.
- **Encode** properties of the algorithm in the convergence set $C \subset H \times \Theta$, e.g.,

\[
C_\alpha := \{ \theta \in \Theta : \ell(A(\alpha, \theta), \theta) \leq \ell(x^{(0)}, \theta) \}
\]
However this might correspond to unlikely special/extreme cases:

We develop a variant that allows to **Trade-Off Guarantees and Speed**.

*The algorithm may diverge (for extreme cases), if this happens in rare cases and the Trade-Off can be controlled.*

- **Account for likelihood** of e.g. the worst-case.
- **Encode** properties of the algorithm in the convergence set \( C \subset \mathcal{H} \times \Theta \), e.g.,

  \[
  C_\alpha := \{ \theta \in \Theta : \ell(\mathcal{A}(\alpha, \theta), \theta) \leq \ell(x^{(0)}, \theta) \}.
  \]

- **Condition** on it to get the convergence risk \( \mathcal{R}_c : \mathcal{H} \rightarrow \mathbb{R}_{\geq 0} : \)

  \[
  \mathcal{R}_c(\alpha) := \mathbb{E}[\ell(\mathcal{A}(\alpha, \mathcal{G}), \mathcal{G}) \mid C_\alpha].
  \]
However this might correspond to unlikely special/extreme cases:

We develop a variant that allows to Trade-Off Guarantees and Speed.

The algorithm may diverge (for extreme cases), if this happens in rare cases and the Trade-Off can be controlled.

- Account for likelihood of e.g. the worst-case.
- Encode properties of the algorithm in the convergence set $C \subset \mathcal{H} \times \Theta$, e.g.,

$$C_{\alpha} := \{ \theta \in \Theta : \ell(A(\alpha, \theta), \theta) \leq \ell(x^{(0)}, \theta) \}.$$

- Condition on it to get the convergence risk $\mathcal{R}_c : \mathcal{H} \to \mathbb{R}_{\geq 0}$:

$$\mathcal{R}_c(\alpha) := \mathbb{E}\left[\ell(A(\alpha, \mathcal{S}), \mathcal{S}) \mid C_{\alpha}\right].$$

- Guarantees in form of the convergence probability $\mathbb{P}_\mathcal{S}[C_{\alpha}]$ instead of convergence for every sample.
Theorem: Under mild assumptions, it holds for $\varepsilon > 0$:

$$P_{D_N} \left\{ \forall \lambda \in \Lambda, \forall Q \ll P_S : E_Q[R_c] \leq E_Q[\hat{R}_c] + G(N, \lambda, Q, \varepsilon) \right\} \geq 1 - \varepsilon.$$
Applying the same machinery again yields the following **generalization**:

**Theorem:** Under mild assumptions, it holds for \( \varepsilon > 0 \):

\[
P_{\mathcal{D}_N} \{ \forall \lambda \in \Lambda, \forall Q \ll P_{\mathcal{S}} : \mathbb{E}_Q[R_c] \leq \mathbb{E}_Q[\hat{R}_c] + G(N, \lambda, Q, \varepsilon) \} \geq 1 - \varepsilon.
\]

- Learning must achieve that \( \mathcal{A} \) converges for “sufficiently many problems” (according to the convergence probability).
- Therefore, the algorithm can focus on quickly solving the remaining problems.
Applying the same machinery again yields the following generalization:

**Theorem:** Under mild assumptions, it holds for $\varepsilon > 0$:

$$\mathbb{P}_{\mathcal{D}_N} \{ \forall \lambda \in \Lambda, \forall Q \ll P_J : \mathbb{E}_Q[R_c] \leq \mathbb{E}_Q[\hat{R}_c] + G(N, \lambda, Q, \varepsilon) \} \geq 1 - \varepsilon.$$  

Learning must achieve that $\mathcal{A}$ converges for “sufficiently many problems” (according to the convergence probability).

Therefore, the algorithm can focus on quickly solving the remaining problems.

**Example Statement:** With high probability, the algorithm that is trained to optimize 95% of all problems in $\mathcal{D}_N$ quickly, will optimize 95% of all problems quickly.
1) Find a “trainable” initialization by following another algorithm.

2) Find a point inside the constraint with small empirical risk.

3) Run a specifically constrained sampling procedure.

4) Find $\lambda^*$ and perform a reweighting based on closed-form of the posterior.
Learn an Algorithm to Train 2-Layer Regression Networks

Training a 2-layer neural network with ReLU-activations ...

... to perform regression.
Loss over Iterations, Conv. Prob. = 100.0 %

Loss Histogram and PAC-Bound

Cumulative Time to Solve the Test Set
Learning gets faster...

- Iteration 1
- Iteration 3
- Iteration 6
- Iteration 9
- Iteration 12
- Iteration 15
Breaking the barrier of worst-case estimates

\[ \min_{\alpha \in \mathcal{H}} \mathcal{R}(\alpha), \quad \mathcal{R}(\alpha) := \mathbb{E}[\ell(\mathcal{A}(\alpha, \mathcal{S}), \mathcal{S})]. \]

by learning specialized optimization algorithms

\[ \min_{\alpha \in \mathcal{H}} \hat{\mathcal{R}}(\alpha, \mathcal{D}_N), \quad \hat{\mathcal{R}}(\alpha, \mathcal{D}_N) := \frac{1}{N} \sum_{i=1}^{N} \ell(\mathcal{A}(\alpha, \mathcal{S}_i), \mathcal{S}_i). \]

with theoretical guarantees via PAC-Bayes generalization bounds:

\[ \mathbb{P}_{\mathcal{D}_N} \left\{ \forall \lambda \in \Lambda, \forall \mathcal{Q} \ll \mathbb{P}_\mathcal{D} : \mathbb{E}_\mathcal{Q}[\mathcal{R}] \leq \mathbb{E}_\mathcal{Q}[\hat{\mathcal{R}}] + G(N, \lambda, \mathcal{Q}, \epsilon) \right\} \geq 1 - \epsilon. \]