PhD Candidacy Exam — Study Notes

Machine Learning & Algorithms · Computer Science

Candidate: Arjun Mehta Advisor: Prof. Chen Exam Date: September 2025

Focus areas: **Statistical Learning Theory** · **Optimization** · **Probabilistic Graphical Models** · **Algorithms & Complexity**

Statistical Learning Theory

Overview

The PAC (Probably Approximately Correct) framework formalizes when and why empirical risk minimization generalizes. Key objects: hypothesis class $\mathcal{H}$, loss ℓ, true risk $R(h)$, empirical risk $\hat{R}_n(h)$.

Theorem (Hoeffding / Uniform Convergence)

For a finite class $|\mathcal{H}| < \infty$ and bounded loss $\ell \in [0,1]$, with probability at least $1-\delta$:

$$\sup_{h \in \mathcal{H}} \left| R(h) - \hat{R}_n(h) \right| \;\le\; \sqrt{\frac{\ln |\mathcal{H}| + \ln(2/\delta)}{2n}}.$$

The ERM solution $\hat{h}$ satisfies $R(\hat{h}) \le R(h^*) + 2\sqrt{\dfrac{\ln |\mathcal{H}| + \ln(2/\delta)}{2n}}$.

VC Dimension

The **VC dimension** $\mathrm{VC}(\mathcal{H})$ is the largest set size that $\mathcal{H}$ can *shatter*. For binary classification with loss in $\{0,1\}$:

$$R(\hat{h}) - R(h^*) = \mathcal{O}\left(\sqrt{\frac{d \ln(n/d) + \ln(1/\delta)}{n}} \right), \quad d = \mathrm{VC}(\mathcal{H}).$$

Outline:

- Bias–variance trade-off: $\mathbb{E}[\hat{R}] = \text{bias}^2 + \text{variance} + \sigma^2$
- Rademacher complexity $\mathfrak{R}_n(\mathcal{H})$: tighter than VC for specific function classes (e.g. linear classifiers with margin)
- Structural risk minimization: balance $\hat{R}_n(h)$ with a complexity penalty
- Double-descent phenomenon: overparameterized regime can still generalize

Exam Tip

Committees often ask to *derive* the union-bound generalization bound from scratch. Remember: apply Hoeffding per hypothesis, then union bound over $|\mathcal{H}|$, then set the RHS equal to δ and solve. Also know the sample complexity: $n \ge \dfrac{\ln |\mathcal{H}| + \ln(1/\delta)}{2\varepsilon^2}$.

Convex Optimization

Overview

Gradient-based methods minimize $f : \mathbb{R}^d \to \mathbb{R}$. Key properties: convexity, L-smoothness ($\|\nabla f(x) - \nabla f(y)\| \le L\|x-y\|$), μ-strong convexity ($f(y) \ge f(x) + \nabla f(x)^\top (y-x) + \frac{\mu}{2}\|y-x\|^2$).

Theorem (GD Convergence Rates)

With step size $\eta = 1/L$:

$$\text{Convex, } L\text{-smooth:} \quad f(x_T) - f^* \leq \frac{L\|x_0 - x^*\|^2}{2T} = \mathcal{O}(1/T)$$

$$\mu\text{-strongly convex, } L\text{-smooth:} \quad \|x_T - x^*\|^2 \leq \left(1 - \frac{\mu}{L}\right)^T \|x_0 - x^*\|^2 \quad \text{(linear)}$$

The ratio $\kappa = L/\mu$ is the **condition number**; ill-conditioned problems ($\kappa \gg 1$) converge slowly.

Stochastic Gradient Descent (SGD)

Use noisy gradient $g_t = \nabla f_{i_t}(x_t)$ where $i_t \sim \text{Uniform}([n])$. Under $\mathbb{E}[g_t \mid x_t] = \nabla f(x_t)$ and $\mathbb{E}[\|g_t\|^2] \leq G^2$, with step $\eta_t = c/\sqrt{T}$:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[f(x_t)] - f^* = \mathcal{O}\left(\frac{G\|x_0 - x^*\|}{\sqrt{T}}\right).$$

Acceleration & Variants:

- **Nesterov momentum**: achieves $\mathcal{O}(1/T^2)$ for convex smooth f — optimal for first-order methods
- **Adam**: adaptive per-coordinate learning rates; not always convergent for convex f but widely used
- **Frank-Wolfe / Conditional Gradient**: projects onto convex set via linear oracle; $\mathcal{O}(1/T)$ without matrix inversion

~~Exam Tip~~

Know the *lower bounds*: no first-order method can do better than $\Omega(1/T^2)$ for convex smooth problems (Nesterov lower bound). Be able to state why SGD has an $\mathcal{O}\left(1/\sqrt{T}\right)$ rate even for smooth strongly-convex objectives (variance term dominates).

Probabilistic Graphical Models

Overview

PGMs factorize a joint distribution $p(x_1, \ldots, x_n)$ using conditional independence encoded by a graph $\mathcal{G}$.

Bayesian Network & Markov Network

Bayesian net (DAG): $p(\mathbf{x}) = \prod_i p(x_i \mid x_{\text{pa}(i)})$, where $\text{pa}(i)$ are parents of node i.
Markov network (undirected): $p(\mathbf{x}) = \frac{1}{Z}\prod_{C \in \text{cliques}} \psi_C(\mathbf{x}_C)$, $Z = \sum_{\mathbf{x}} \prod_C \psi_C(\mathbf{x}_C)$ (partition function).

Belief Propagation (Sum-Product)

On a tree-structured factor graph, *exact* marginals are computed in $\mathcal{O}\left(n \cdot |\mathcal{X}|^2\right)$ time via message passing:

$$\mu_{f \to x}(x) = \sum_{\sim x}\left[f(\mathbf{x}_f) \prod_{y \in \text{ne}(f)\setminus x} \mu_{y \to f}(y)\right].$$

On loopy graphs, Loopy BP is approximate but often effective in practice (e.g. turbo codes, MRFs for image segmentation).

Inference algorithms:

- Exact: Variable Elimination, Junction Tree ($\mathcal{O}\big(n \cdot |\mathcal{X}|^{w+1}\big)$, w = treewidth)
- Approximate: Loopy BP, Mean-Field variational inference (minimize $\mathrm{KL}[q\|p]$), MCMC (Gibbs sampling)
- Latent variable models: EM algorithm maximizes $\mathcal{L}(\theta) = \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z}; \theta)]$

Algorithms & Complexity

Complexity Classes

$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$; whether $\mathbf{P} = \mathbf{NP}$ is open. A language $L \in \mathbf{NP}$ iff there is a poly-time verifier. A problem is **NP-hard** if every NP problem reduces to it in polynomial time.

Cook–Levin Theorem

SAT is NP-complete. By reduction, 3-SAT, Clique, Vertex Cover, Independent Set, Hamiltonian Cycle are all NP-complete. Approximability:

- Vertex Cover: 2-approximation via maximal matching
- Set Cover: $\ln n$-approximation via greedy (tight assuming $\mathbf{P} \neq \mathbf{NP}$)
- Max-Clique: no poly-time $n^{1-\varepsilon}$-approximation (assuming ETH)

Randomized Complexity

$\mathbf{BPP}$: poly-time randomized, error $\leq 1/3$. $\mathbf{RP}$: one-sided error. $\mathbf{ZPP} = \mathbf{RP} \cap \text{co-RP}$ (zero error, expected poly time). Inclusions: $\mathbf{P} \subseteq \mathbf{ZPP} \subseteq \mathbf{RP} \subseteq \mathbf{BPP}$; conjectured $\mathbf{BPP} = \mathbf{P}$.

Exam Tip

To show NP-hardness: (1) choose a known NP-hard problem, (2) give a poly-time *many-one reduction* from it to your problem, (3) show correctness in both directions. Common pitfall: reducing the wrong way (from your problem, not *to* it).

Key References

1. Shalev-Shwartz & Ben-David. *Understanding Machine Learning*. Cambridge, 2014.
2. Nesterov, Y. *Introductory Lectures on Stochastic Optimization*. Springer, 2004.
3. Koller & Friedman. *Probabilistic Graphical Models*. MIT Press, 2009.
4. Sipser, M. *Introduction to the Theory of Computation*, 3rd ed. Cengage, 2012.
5. Bartlett & Mendelson. "Rademacher and Gaussian Complexities." *JMLR*, 3:463–482, 2002.