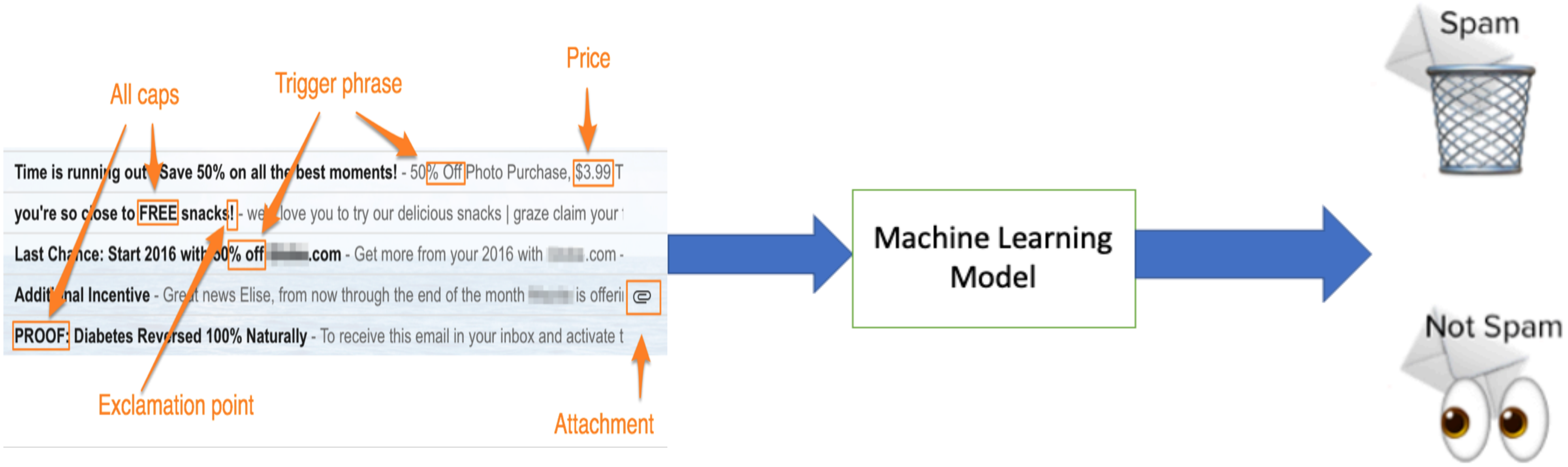
Fast Algorithms for AUC Maximization

Mingrui Liu
Boston University

June 02, 2021

An Example of Spam Email Classification



$\mathbf{x}$: feature vector

model: $f(\mathbf{x})$

$y \in \{\text{spam, not spam}\}$

Risk Minimization

Risk of model f

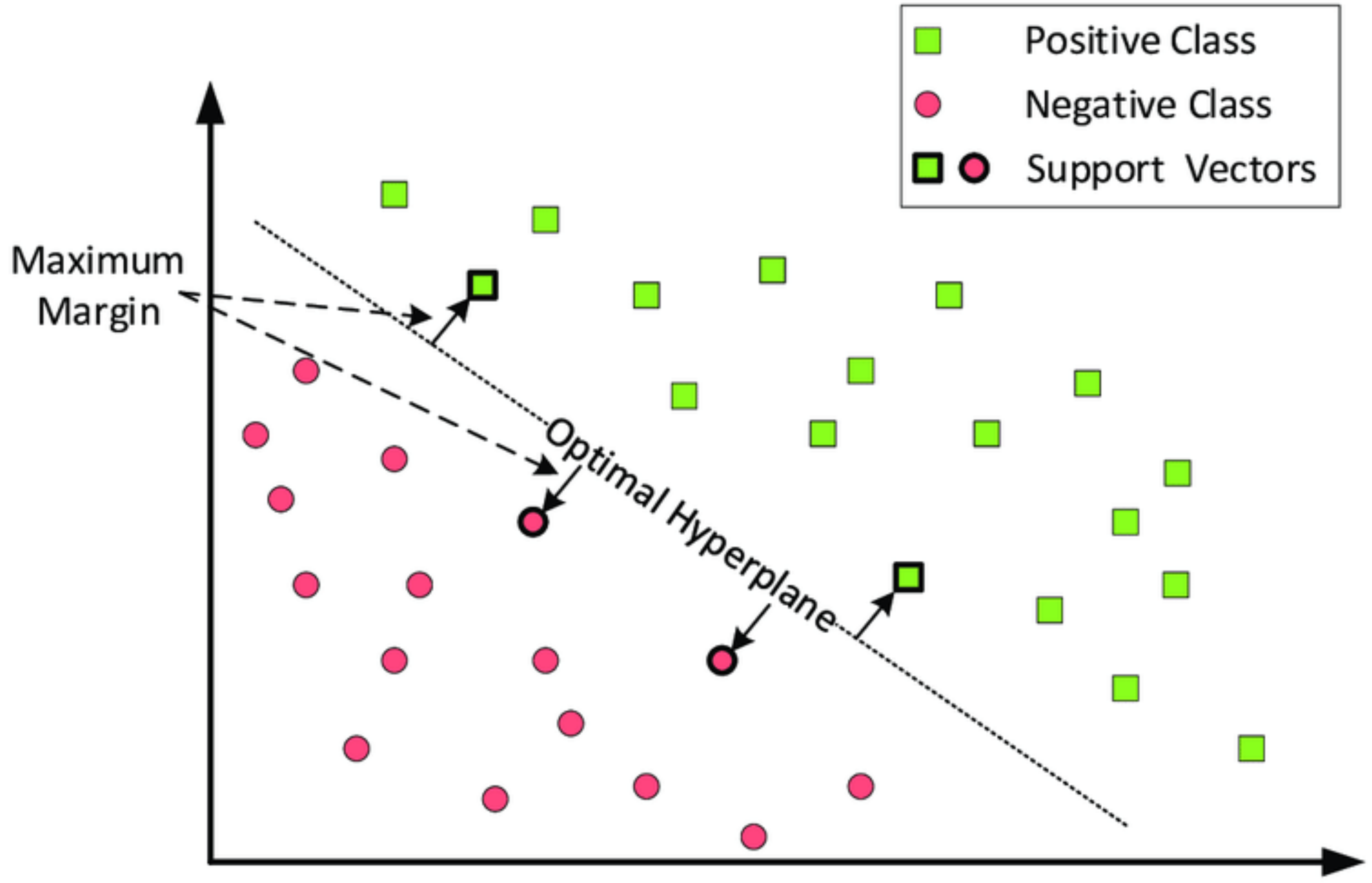
$$f^* = \arg\min_{f \in \mathcal{F}} R(f) := \mathbb{E}_{\mathbf{x},y}\left[\ell(f(\mathbf{x}), y)\right]$$

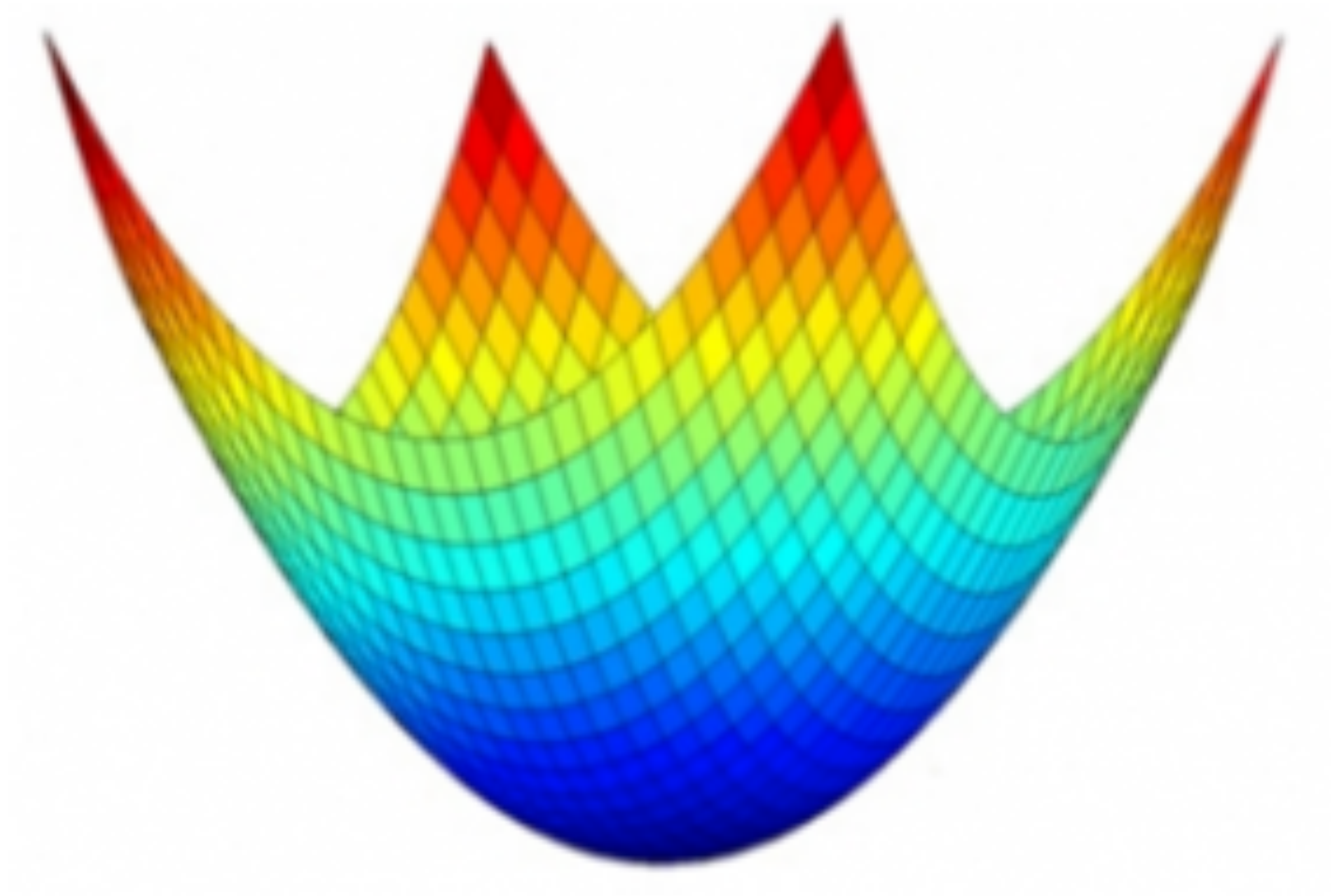
- $\mathcal{F}$: hypothesis class
- Loss function $\ell(\hat{y}, y)$ measures the prediction error

Prediction $\hat{y} = f(\mathbf{w}; \mathbf{x})$

$$\mathbf{w}_* = \arg\min_{\mathbf{w}} \mathbb{E}_{\mathbf{x},y}\left[\ell(f(\mathbf{w}; \mathbf{x}), y)\right]$$

Linear Model: Convex Methods

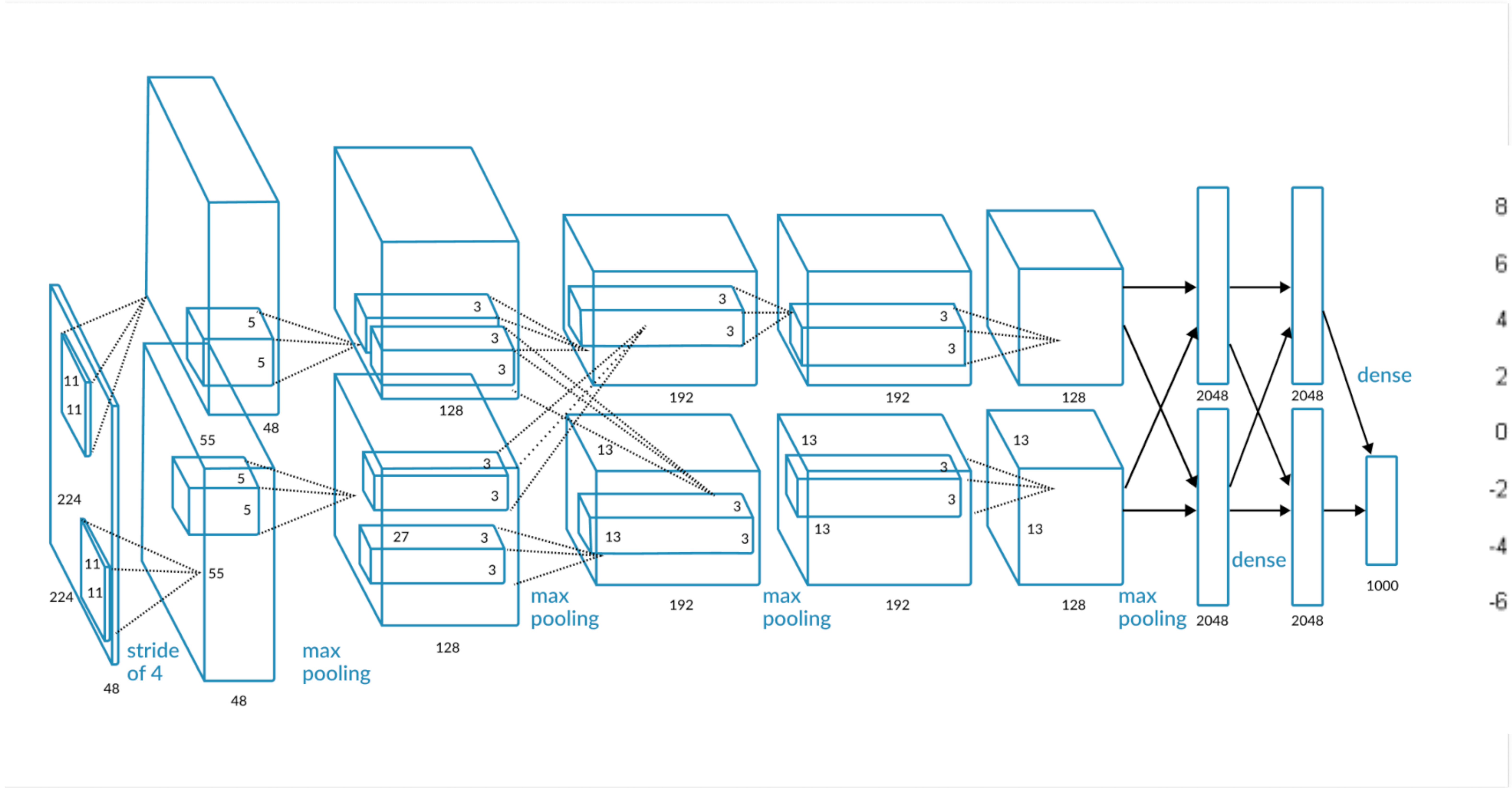


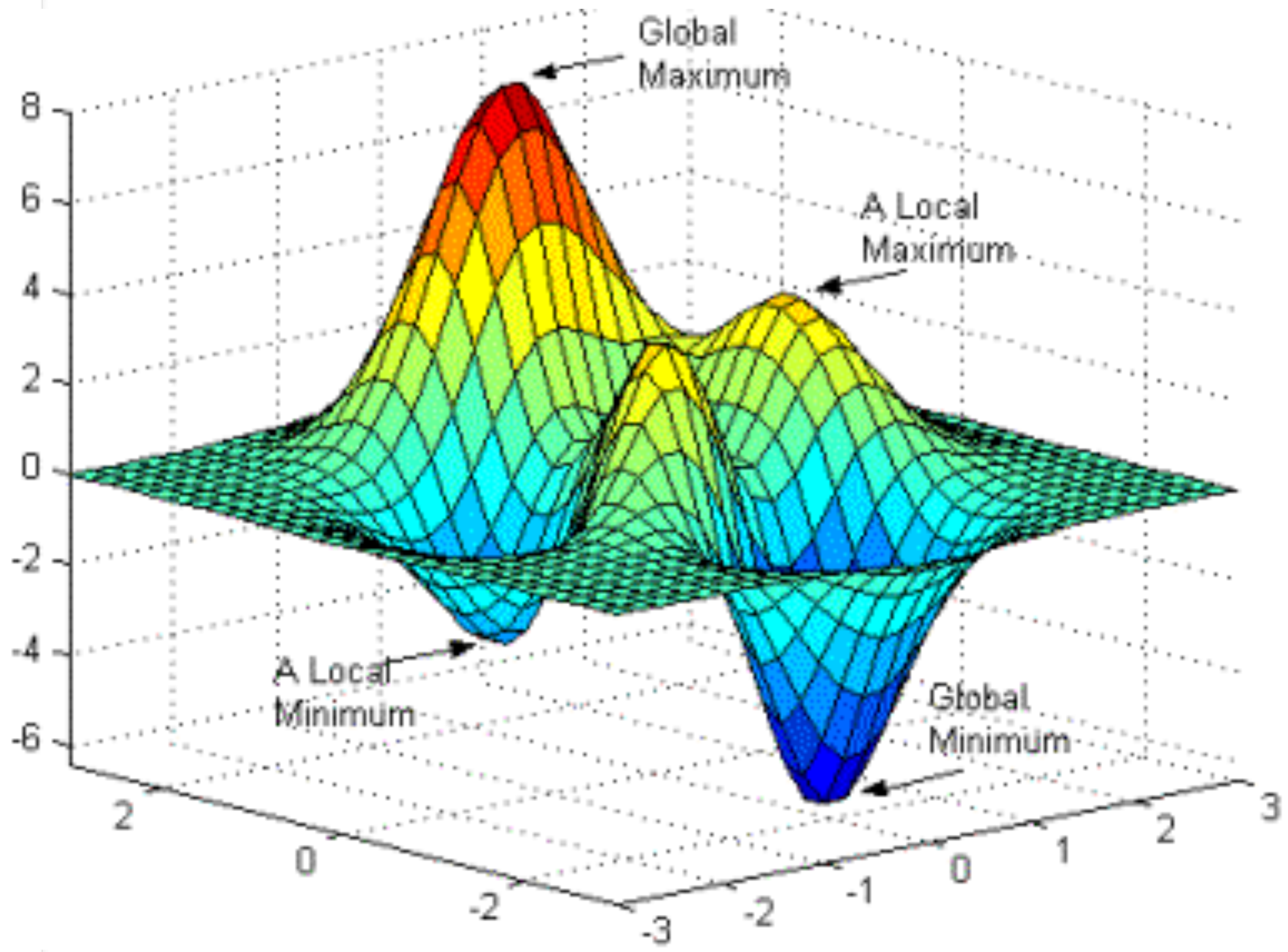


$$\mathbf{x} \to f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^{\top}\mathbf{x}$$

$$\min_{\mathbf{w}} \mathbb{E}_{\mathbf{x},y}\left[\ell(f_{\mathbf{w}}(\mathbf{x}), y)\right]$$

Deep Neural Networks: Nonconvex Methods





Alexnet: $\mathbf{x} \rightarrow f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}_L \circ \sigma\left(\ldots\sigma\left(\mathbf{w}_2 \circ \sigma(\mathbf{w}_1 \circ \mathbf{x})\right)\right)$

$$\min_{\mathbf{w}} \mathbb{E}_{\mathbf{x},y}\left[\ell(f_{\mathbf{w}}(\mathbf{x}), y)\right]$$

Classical Learning Paradigm

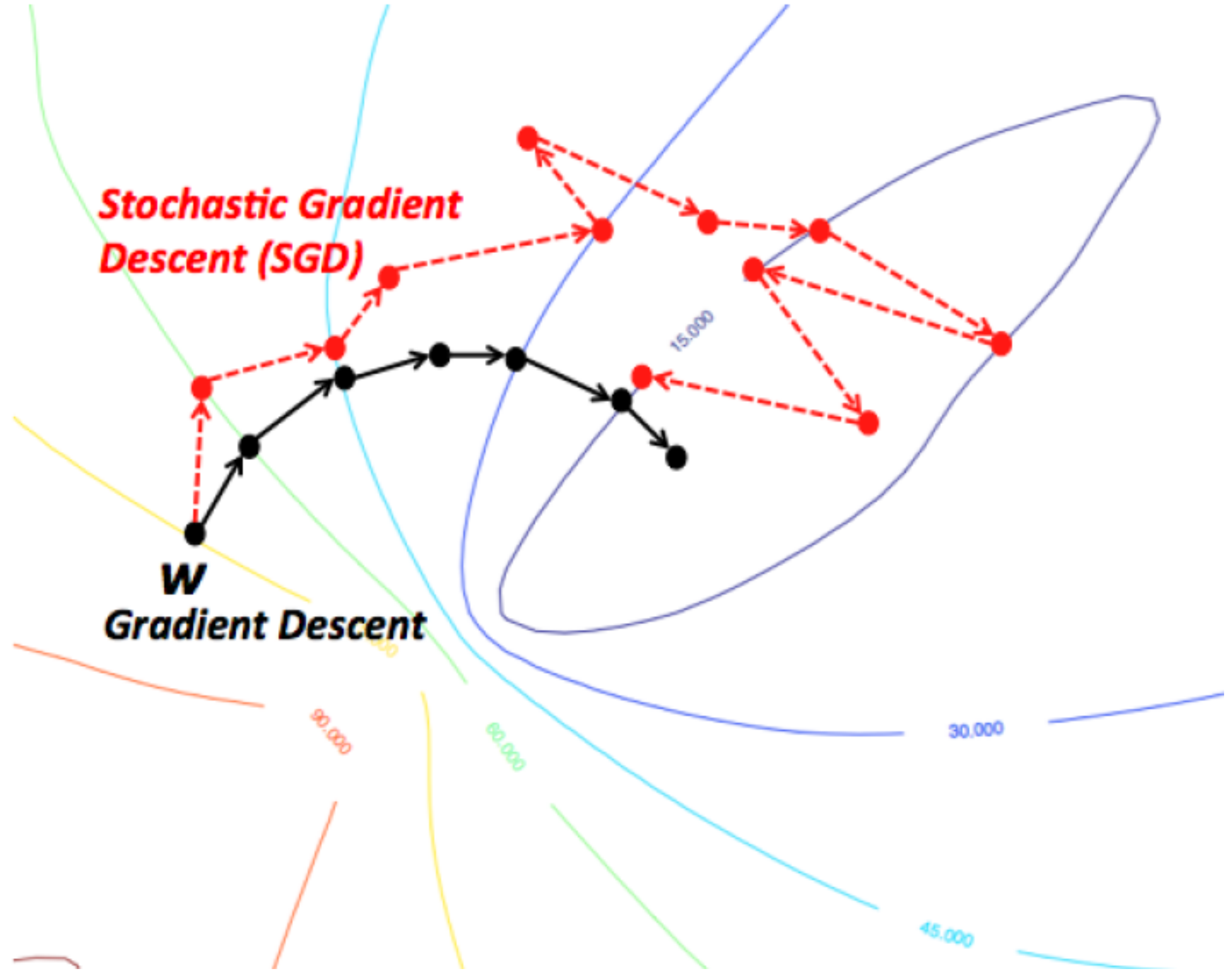
Solve risk minimization by stochastic gradient descent

$$\min_{\mathbf{w}} \mathbb{E}_{\mathbf{x},y}\left[\ell(f_{\mathbf{w}}(\mathbf{x}), y)\right]$$

- Stochastic Gradient Descent (SGD) [Robbins-Monro'51]
  - Sample $(\mathbf{x}_t, y_t)$ uniformly
  - $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \partial\ell(\mathbf{w}_t, \mathbf{x}_t, y_t)$

Stochastic gradient

Learning rate



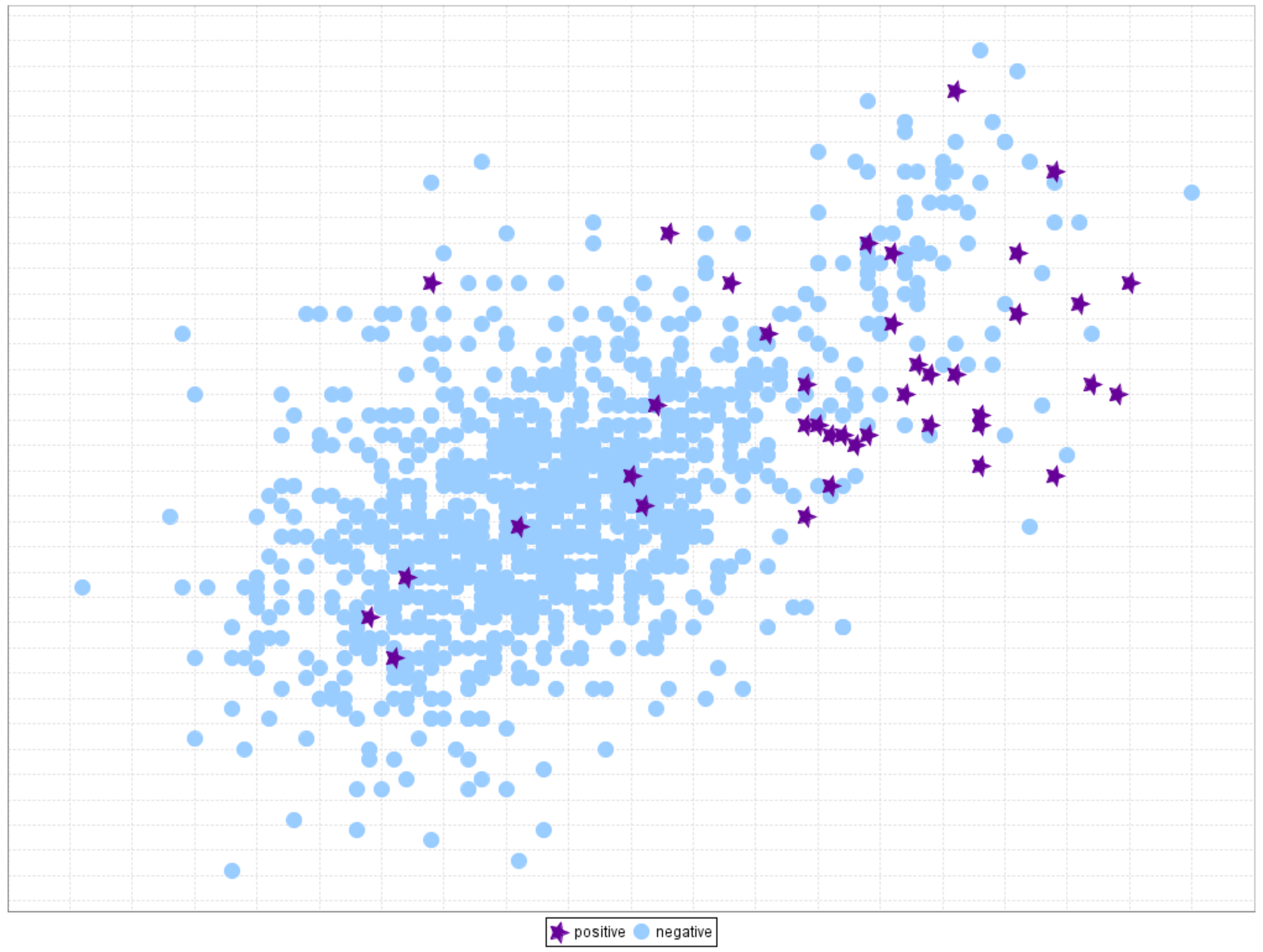
Goal: For an small $\epsilon > 0$, find a solution $\hat{\mathbf{w}}$ such that

$$F\left(\hat{\mathbf{w}}\right) - \min_{\mathbf{w}} F\left(\mathbf{w}\right) \leq \epsilon$$

- Iteration complexity of SGD for convex function: $O(1/\epsilon^2)$ $(\eta_t = 1/\sqrt{t})$

Limitations of Classical Learning Paradigm

- Slow convergence: $O(1/\epsilon^2)$ (e.g., 10^{12} iterations if $\epsilon = 10^{-6}$)
- Not sufficient for learning imbalanced data

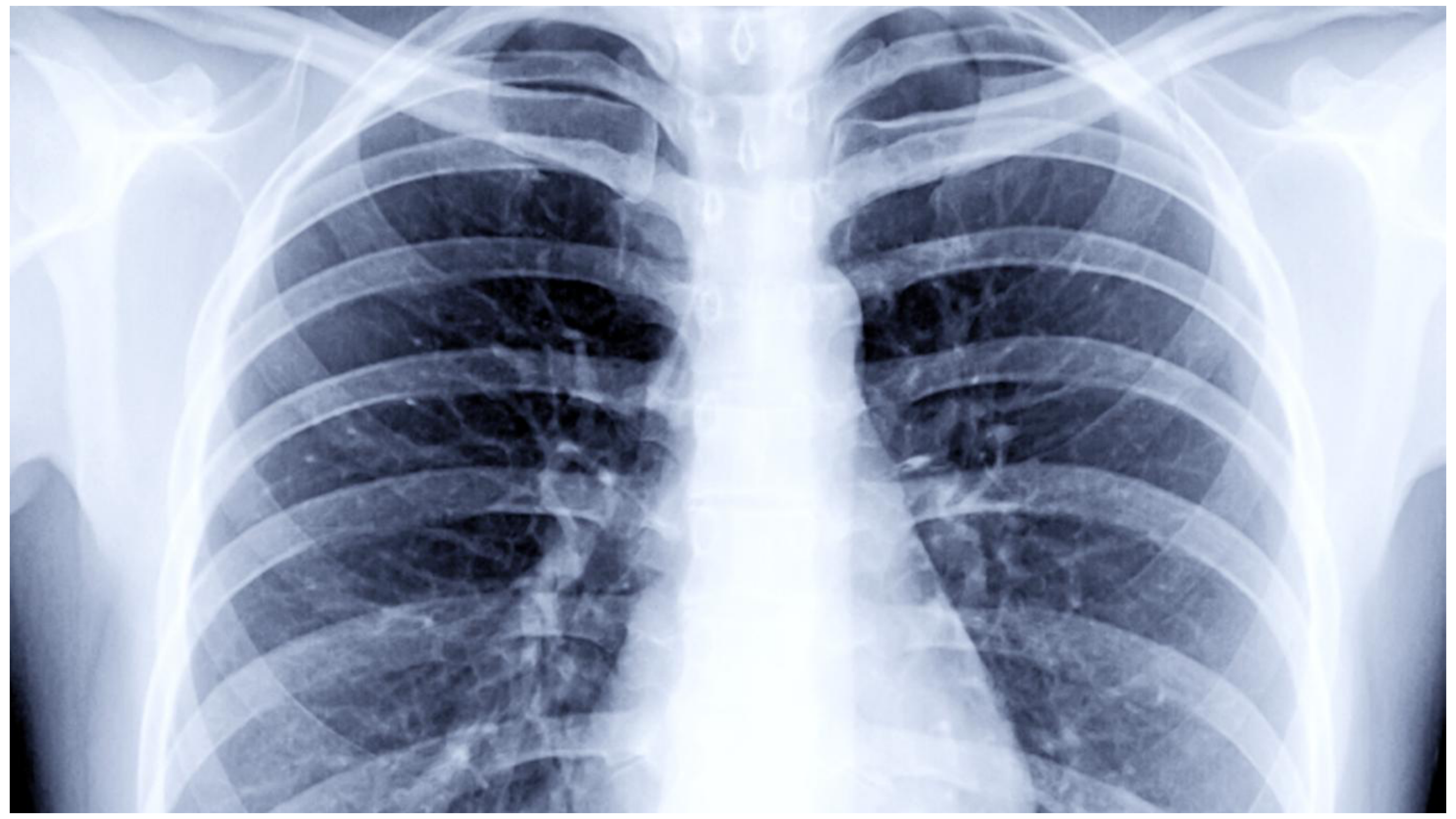


Imbalanced data

Imbalanced Data is Common

Credit Card Fraud Detection

Software Bug Detection



Medical Image Classification

Minimizing Classification Error Rate is not a good idea

Q: How to efficiently learn from imbalanced data?

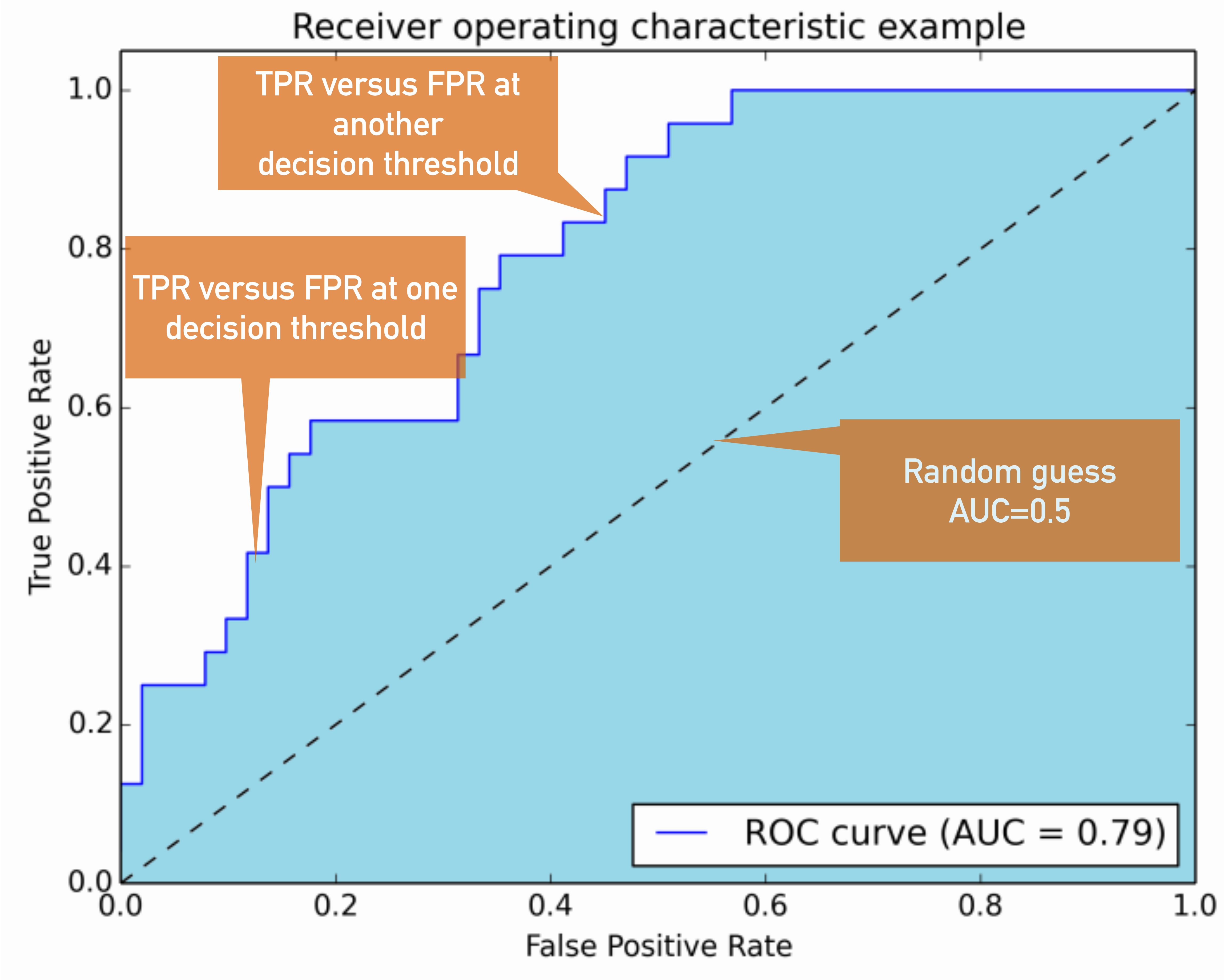
A: Fast Algorithms for AUC Maximization

Area Under the ROC Curve (AUC)

	Ground truth: Positive	Ground truth: Negative
Predict: Positive	TP	FP
Predict: Negative	FN	TN

TPR = TP/(TP+FN), FPR = FP/(FP+TN)

- ROC curve considers different classification thresholds
- Better measure than classification error rate



Probabilistic Interpretation of AUC

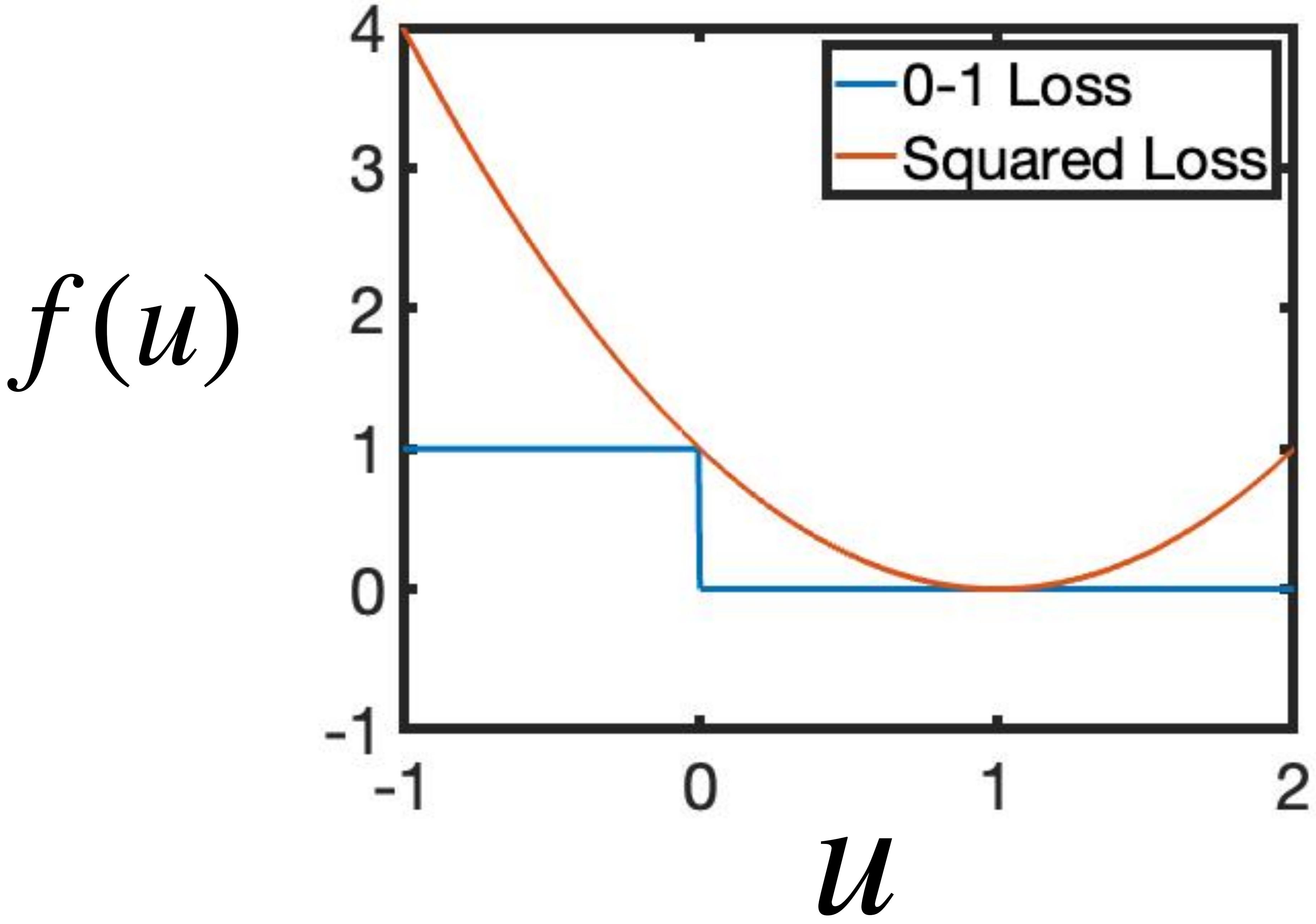
$$\text{AUC}(h) = \Pr\left(h(\mathbf{x}) \geq h(\mathbf{x}') \mid y = 1, y' = -1\right)$$

- Equivalent to Wilcoxon Statistics [Hanley and McNeil'82]
- h : prediction model (e.g., linear model, deep neural network)
- $(\mathbf{x}, y), (\mathbf{x}', y')$: feature-label pair

Surrogate Loss

$$\max_h \text{AUC}(h) = \Pr\left(h(\mathbf{x}) \geq h(\mathbf{x}') \mid y = 1, y' = -1\right) = \mathbb{E}\left[\overbrace{\mathbf{I}}^{\text{0-1 Loss}}\left(h(\mathbf{x}) \geq h(\mathbf{x}')\right) \mid y = 1, y' = -1\right]$$

$$\min_h \text{Surrogate-AUC}(h) = \mathbb{E}\left[\overbrace{f}^{\text{Surrogate Loss}}\left(h(\mathbf{x}) - h(\mathbf{x}')\right) \mid y = 1, y' = -1\right]$$



0-1 Loss: $f(u) = \mathbf{I}(u \leq 0)$

Squared Loss: $f(u) = (u - 1)^2$

Non-decomposable over individual training data, not suitable for online learning

Min-max Reformulation with Squared Loss

Model parameter

$$\min_{\mathbf{w}\in\mathbf{R}^d} \mathbb{E}_{\mathbf{z},\mathbf{z}'}\left[\left(h_{\mathbf{w}}(\mathbf{x}) - h_{\mathbf{w}}(\mathbf{x}') - 1\right)^2 \mid y = 1, y' = -1\right]$$

$\mathbf{z} = (\mathbf{x}, y), \mathbf{z}' = (\mathbf{x}', y')$

- Min-max Reformulation [Ying-Wen-Lyu'16]

Auxiliary variables

Classification threshold

Model parameter

$\mathbf{z} = (\mathbf{x}, y)$

$$\min_{\mathbf{w}\in\mathbb{R}^d, (a,b)\in\mathbb{R}^2} \max_{\alpha\in\mathbb{R}} f(\mathbf{w}, a, b, \alpha) = \mathbb{E}_{\mathbf{z}}\left[F(\mathbf{w}, a, b, \alpha; \mathbf{z})\right]$$

- [Ying-Wen-Lyu'16]
  - focuses on linear model: $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$
  - Convex-concave min-max problem
  - Solve the problem by Primal-Dual Stochastic Gradient (PDSG)

Primal-Dual Stochastic Gradient (PDSG)

- Consider $\min_{\mathbf{v}\in\Omega_1} \max_{\alpha\in\Omega_2} f(\mathbf{v}, \alpha) = \mathbb{E}_{\mathbf{z}}\left[F(\mathbf{v}, \alpha; \mathbf{z})\right]$, where $\mathbf{v} = (\mathbf{w}, a, b)$
- PDSG: $\begin{cases} \mathbf{v}_{t+1} = \Pi_{\Omega_1}\left[\mathbf{v}_t - \eta_t \nabla_{\mathbf{v}} F(\mathbf{v}_t, \alpha_t; \mathbf{z}_t)\right] \\ \alpha_{t+1} = \Pi_{\Omega_2}\left[\alpha_t + \eta_t \nabla_{\alpha} F(\mathbf{w}_t, \alpha_t; \mathbf{z}_t)\right] \end{cases}$

Primal: Stochastic Gradient Descent

Dual: Stochastic Gradient Ascent

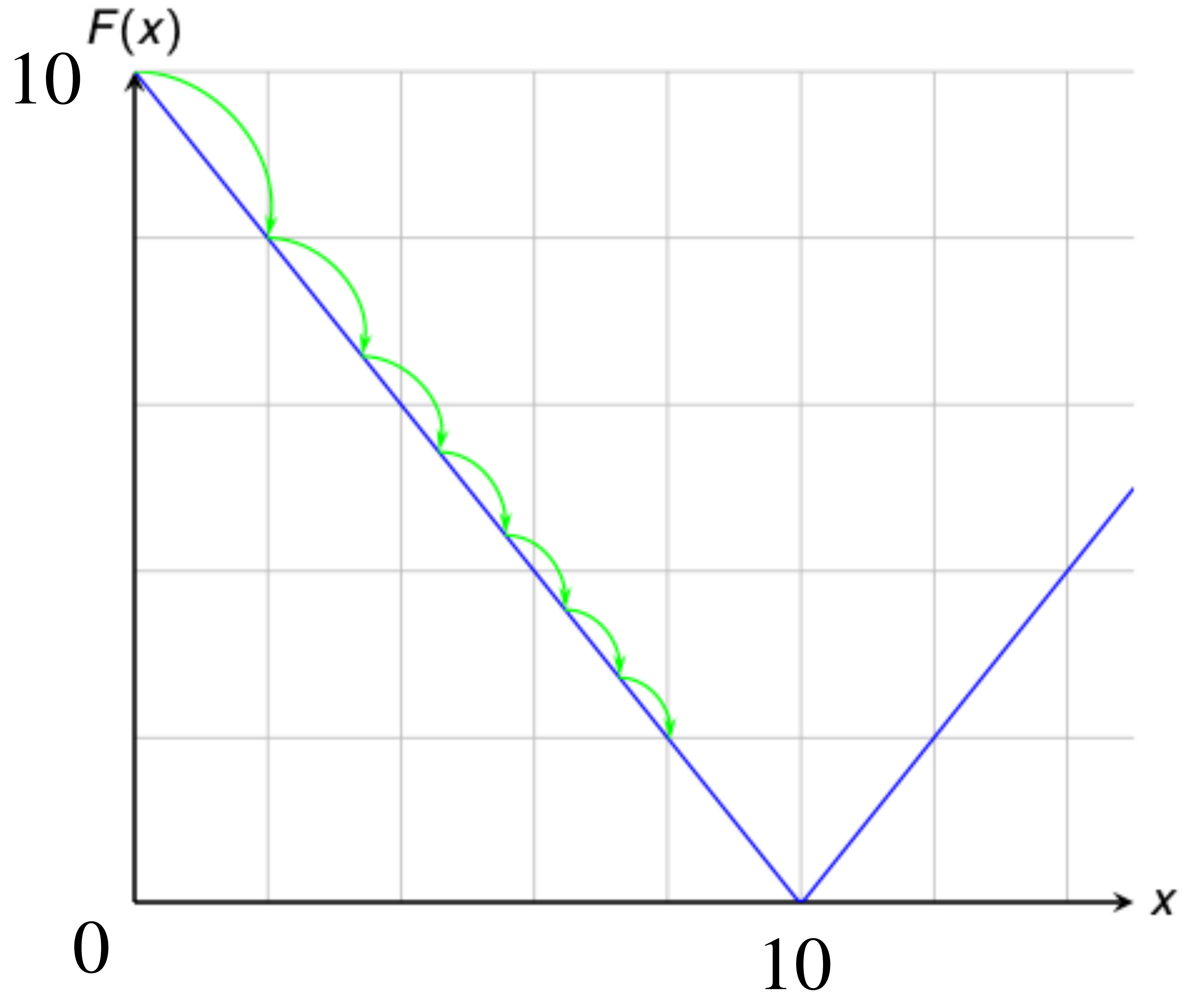
- Return: $\hat{\mathbf{v}} = \sum_{t=1}^{T} \mathbf{v}_t / T, \quad \hat{\alpha} = \sum_{t=1}^{T} \alpha_t / T$
- PDSG is a generalization of SGD for min-max problems (default method)
- $O(1/\epsilon^2)$ iteration complexity for convex-concave min-max problems [Nemirovski-Juditsky-Lan-Shapiro'09]

Question: How to design algorithms for optimizing AUC with lower complexity?

Why PDSG is slow?

$$\min_x |x - 10|$$

Decreasing learning rate ($\eta_t = 1/\sqrt{t}$)



Constant Learning rate doesn't work

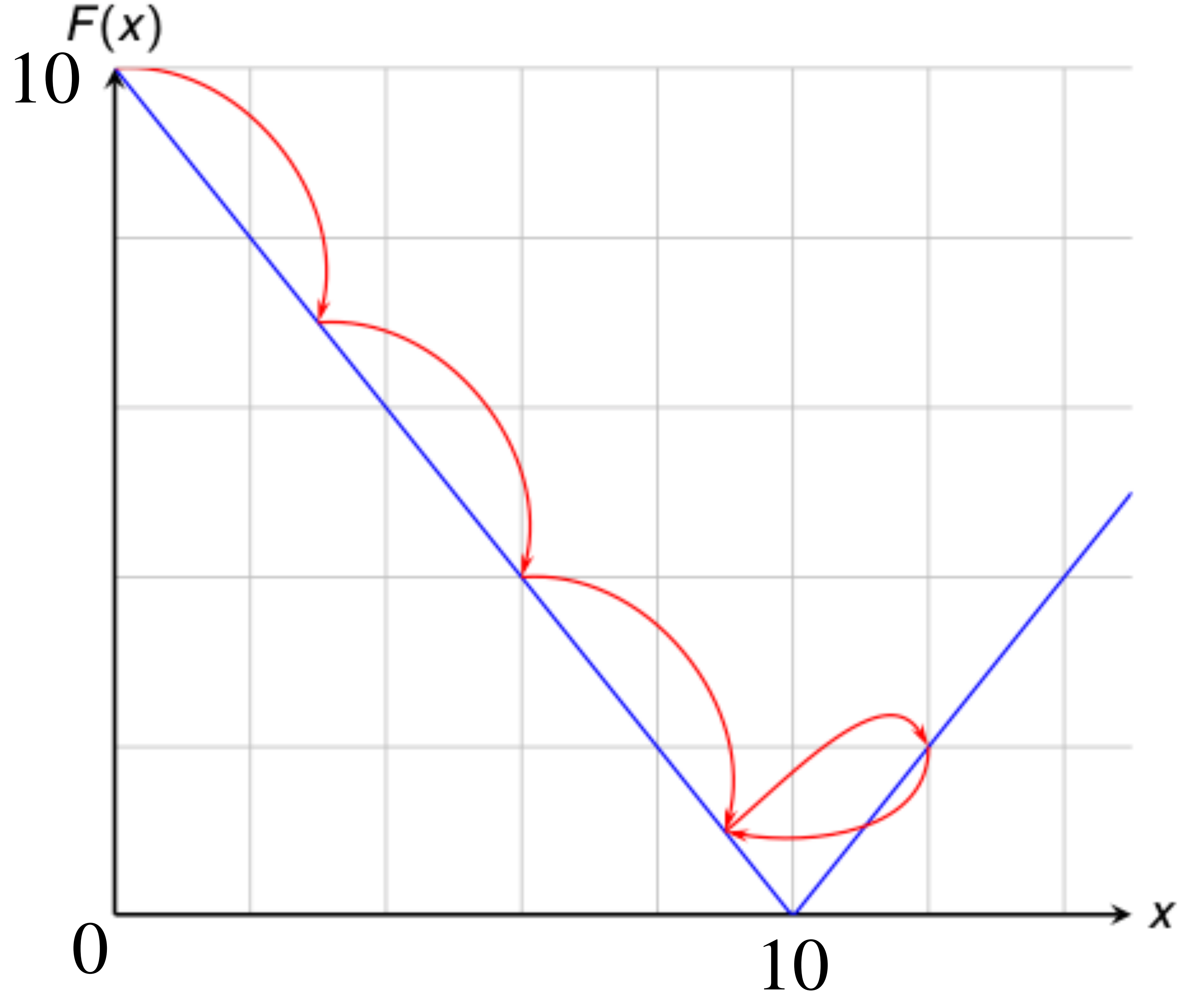
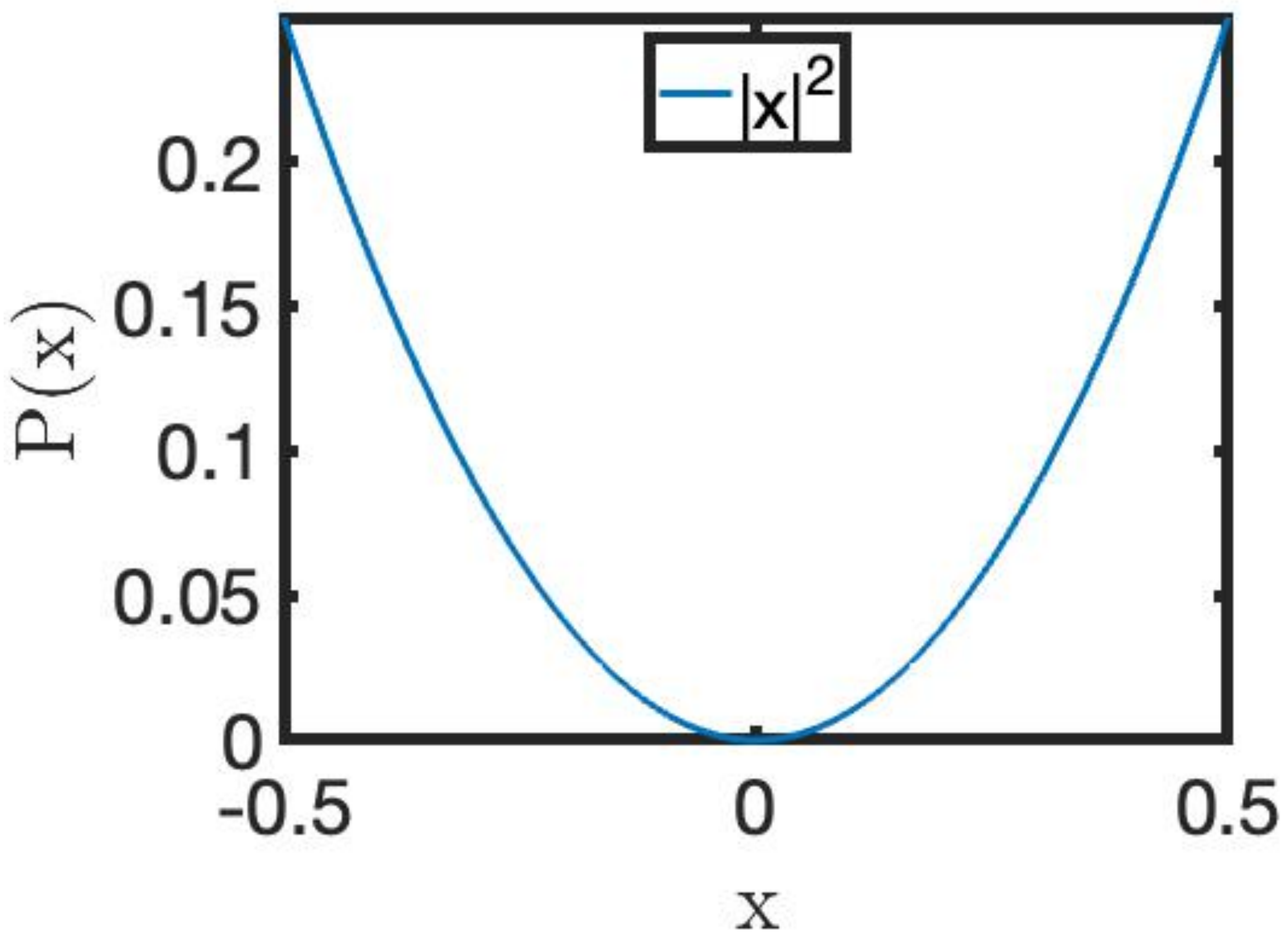


Image Credit: Orabona ICML 2020 tutorial

Key Observation: Quadratic Growth Condition



We prove that Quadratic Growth Condition (QGC) holds for $P(\mathbf{v}) = \max_{\alpha \in \Omega_2} f(\mathbf{v}, \alpha)$:

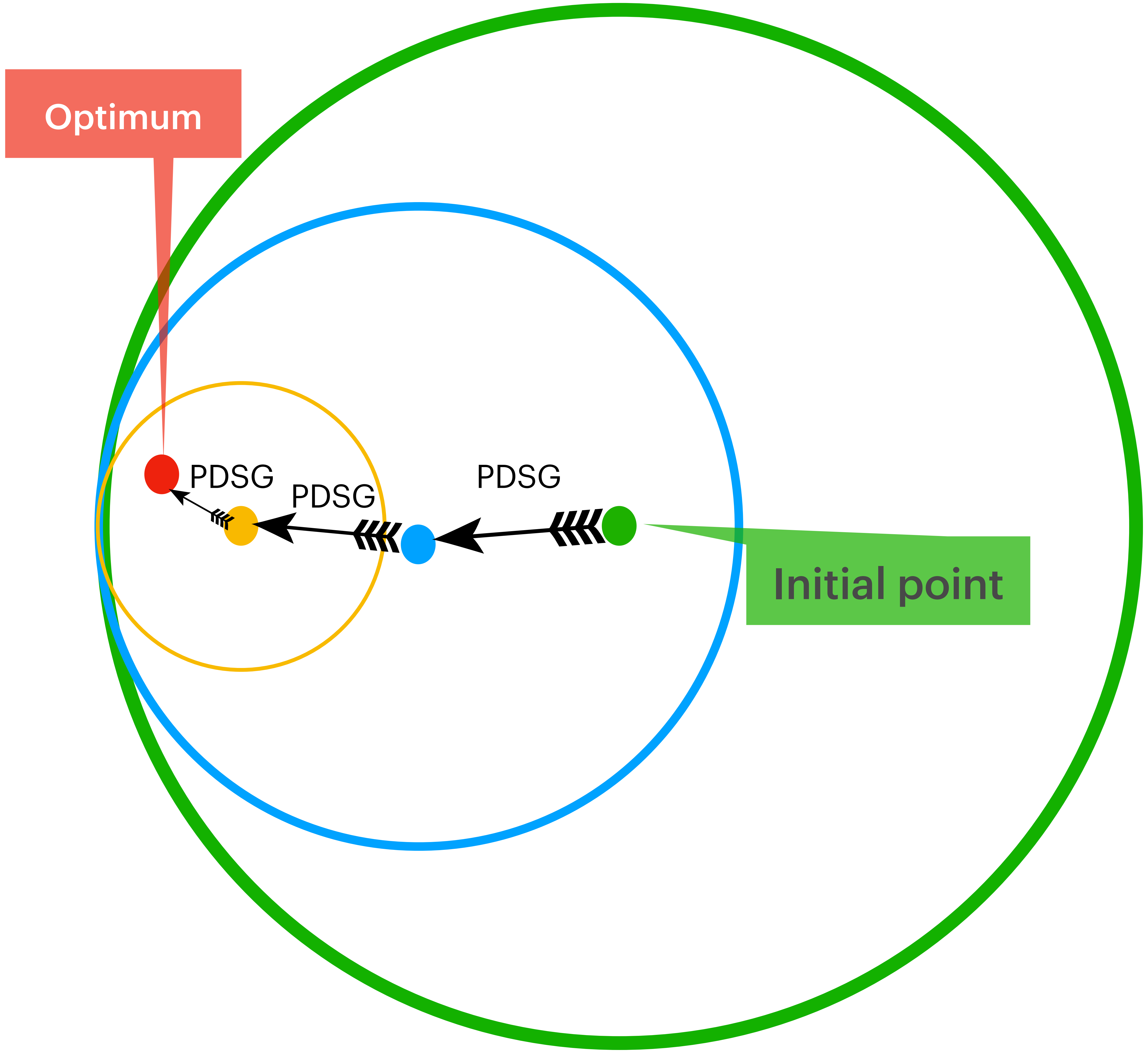
$$\|\mathbf{v} - \mathbf{v}_*\| \le c\left(P(\mathbf{v}) - \min_{\mathbf{v} \in \Omega_1} P(\mathbf{v})\right)^{1/2}$$

Optimal solution

Decreasing Learning Rate → Stagewise Decreasing Learning Rate

Fast Stochastic AUC Maximization

[L.-Zhang-Chen-Wang-Yang, ICML 18]



Algorithm 1 FSAUC

Set $m = \lfloor \frac{1}{2} \log_2 \frac{2n}{\log_2 n} \rfloor - 1$, $n_0 = \lfloor n/m \rfloor$

for $k = 0, \ldots, m-1$ **do**

$(\widehat{\mathbf{v}}_{k+1}, \widehat{\alpha}_{k+1}) = \text{PDSG}(\widehat{\mathbf{v}}_k, \widehat{\alpha}_k, R_k, D_k, n_0, \eta_k)$

$\eta_{k+1} = \eta_k/2,\ R_{k+1} = R_k/2,\ D_{k+1} = D_k/2$

end for

Constrained version of PDSG

Shrinking learning rate and domain

Algorithm 2 PDSG$(\mathbf{v}_1, \alpha_1, R, D, T, \eta)$

for $t = 1, \ldots, T$ **do**

$\mathbf{v}_{t+1} = \underbrace{\Pi_{\|\mathbf{v}-\mathbf{v}_1\| \le R}}_{\text{projection onto a ball}} (\mathbf{v}_t - \eta \nabla_{\mathbf{v}} F(\mathbf{v}_t, \alpha_t; \mathbf{z}_t))$

$\alpha_{t+1} = \underbrace{\Pi_{\|\alpha-\alpha_1\| \le D}}_{\text{projection onto a ball}} (\alpha_t + \eta \nabla_{\alpha} F(\mathbf{v}_t, \alpha_t; \mathbf{z}_t))$

end for

return $\bar{\mathbf{v}}_T = \frac{1}{T} \sum_{i=1}^{T} \mathbf{v}_i$

$\bar{\alpha}_T = \bar{\mathbf{w}}_T^\top [\mathbb{E}(\mathbf{x}|y=-1) - \mathbb{E}(\mathbf{x}|y=1)]$

(closed form of α given $\bar{\mathbf{v}}_T$)

Faster Convergence Exploiting Function Structure

Recall that we prove Quadratic Growth Condition for $P(\mathbf{v}) = \max_{\alpha \in \Omega_2} f(\mathbf{v}, \alpha)$:

$$\|\mathbf{v} - \mathbf{v}_*\| \leq c(P(\mathbf{v}) - \min_{\mathbf{v} \in \Omega_1} P(\mathbf{v}))^{1/2}$$

Optimal solution

Theorem [L.-Zhang-Chen-Wang-Yang, ICML'18]

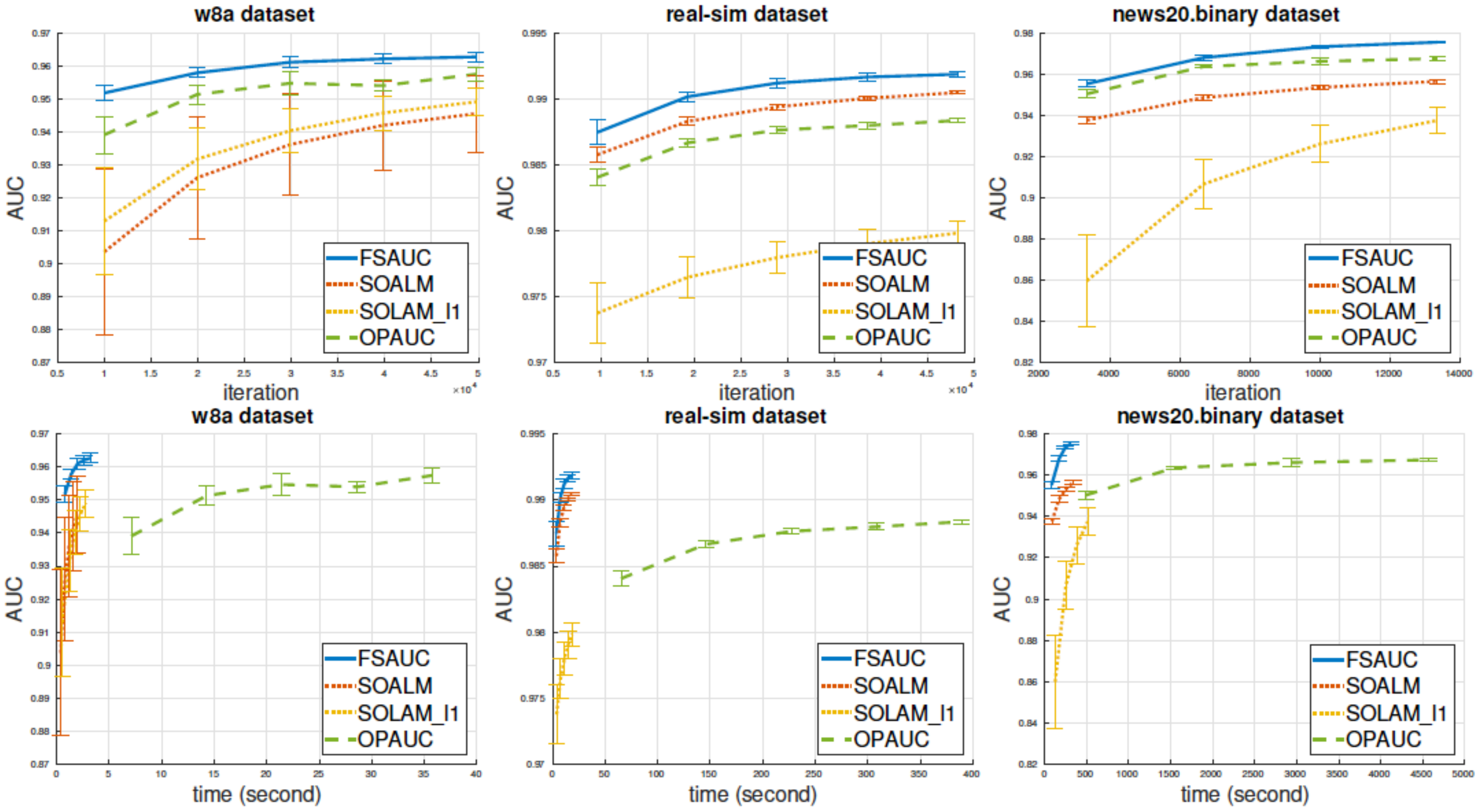
With high probability, $P(\hat{\mathbf{v}}_m) - \min_{\mathbf{v}} P(\mathbf{v}) \leq \widetilde{O}(1/n)$

$$\max_{\mathbf{v}} AUC(\mathbf{v}) - AUC(\hat{\mathbf{v}}_m) \leq \widetilde{O}(1/n)$$

Improved Complexity: $O(1/\epsilon^2) \Rightarrow \widetilde{O}(1/\epsilon)$

$\widetilde{O}(\cdot)$ compresses logarithmic factors

Experiments



$p = 2.97\%$, Very Imbalanced

$p = 30.68\%$, Mildly Imbalanced

$p = 50.29\%$ Balanced

Blue: our algorithm

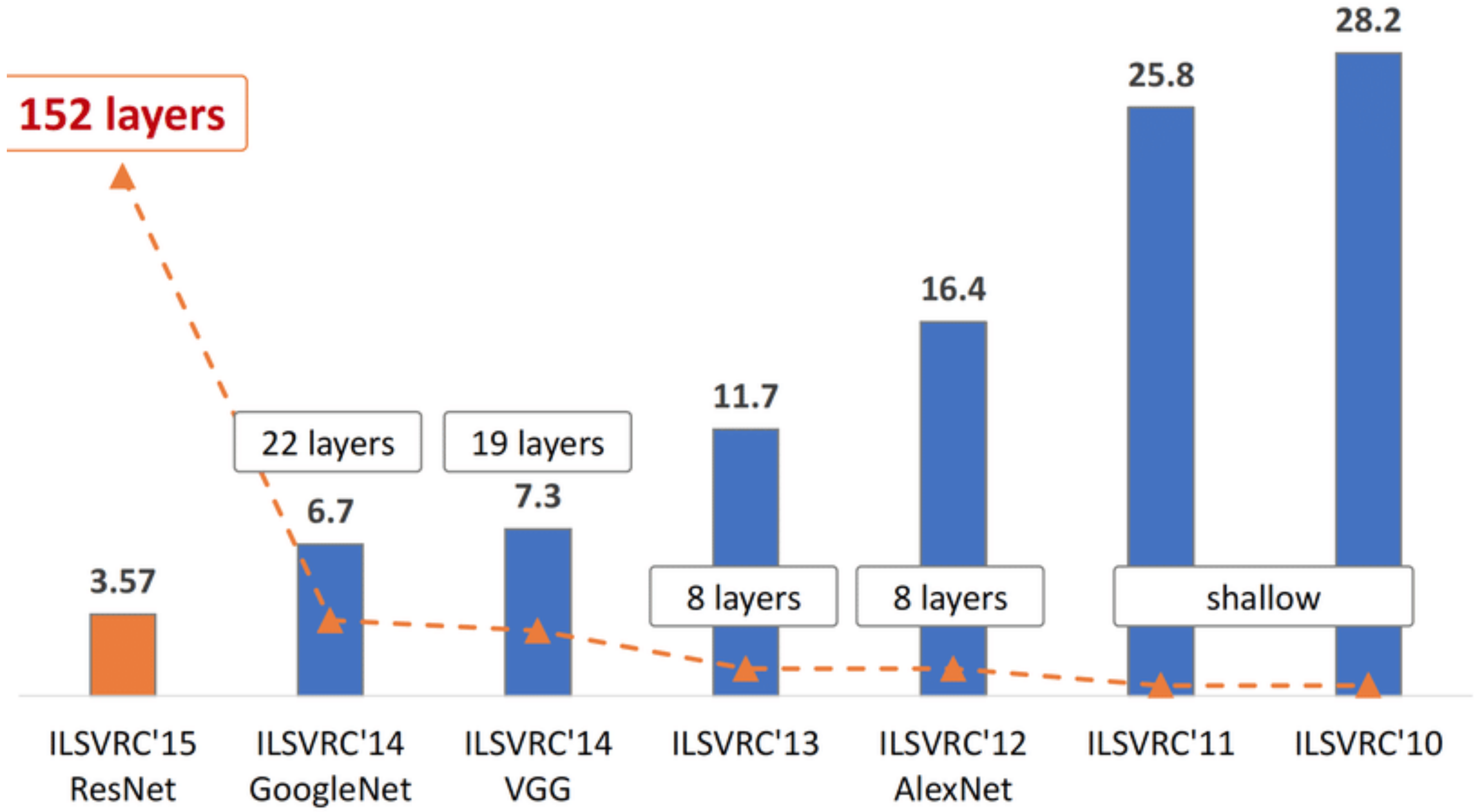
Red: PDSG

Yellow: PDSG with l1-ball constraint

Green: One-pass AUC (each iteration computes covariance matrix)

From Linear Model to Deep Neural Networks

ImageNet: 1000 object classes, 1.2M training, 100k test



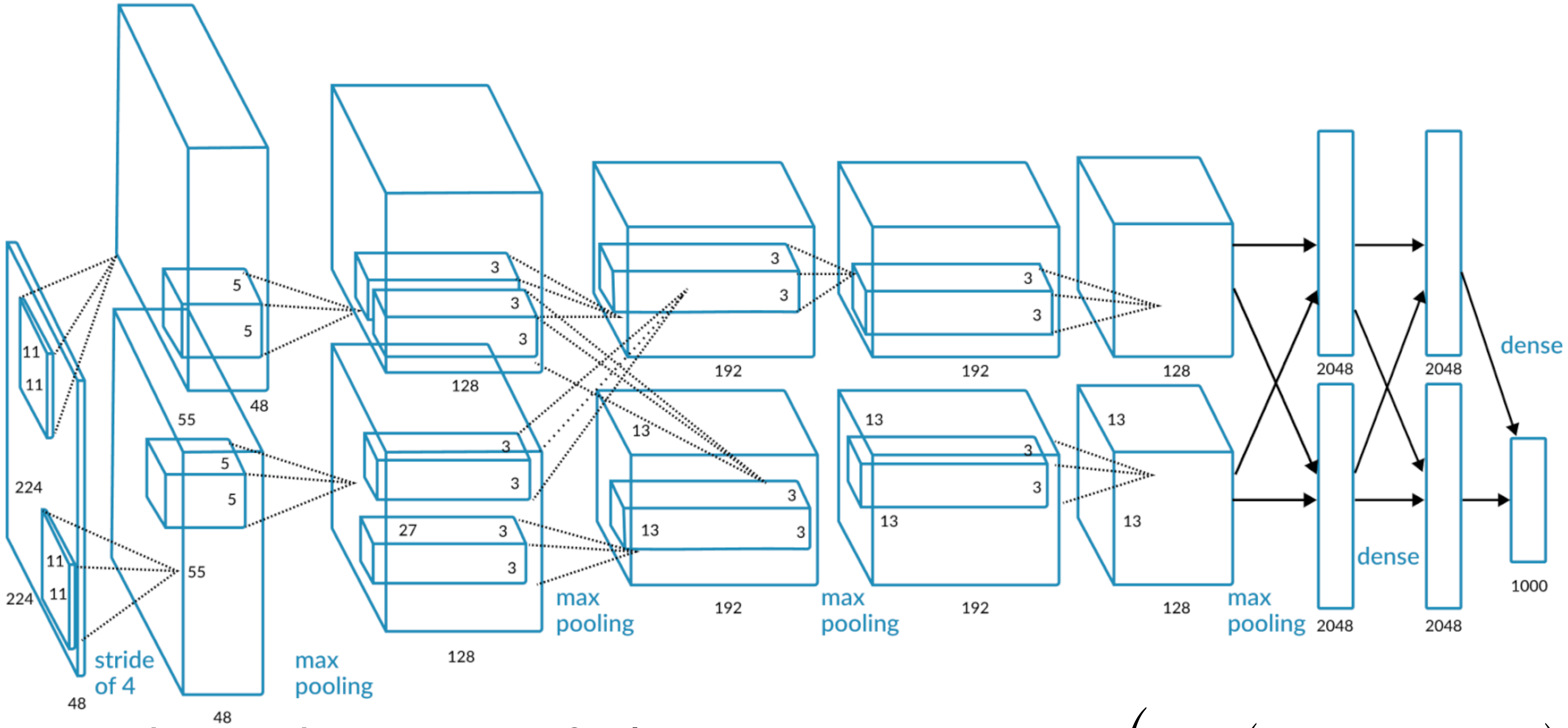
Classification Error Rates for ImageNet Competition

Question: How to maximize AUC when the predictive model is a deep neural network?

Difficulty of Optimizing AUC with DNN

- Min-max Reformulation [L.-Yuan-Ying-Yang, ICLR 20]

$$\min_{\mathbf{w}\in\mathbb{R}^d,(a,b)\in\mathbb{R}^2}\ \max_{\alpha\in\mathbb{R}} f(\mathbf{w},a,b,\alpha) = \mathbb{E}_{\mathbf{z}}\left[F(\mathbf{w},a,b,\alpha;\mathbf{z})\right]$$



The architecture of AlexNet: $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}_L \circ \sigma\left(\ldots\sigma\left(\mathbf{w}_2 \circ \sigma(\mathbf{w}_1 \circ \mathbf{x})\right)\right)$

- Difficulty: Nonconvex-Concave Min-max Problem

Nonconvex-Concave Min-max Optimization

$$\min_{\mathbf{v}\in\Omega_1}\max_{\alpha\in\Omega_2} f(\mathbf{v},\alpha) = \mathbb{E}_{\mathbf{z}}\left[F(\mathbf{v},\alpha;\mathbf{z})\right]$$

- Nonconvex in $\mathbf{v}$ and concave in α
- PDSG does not work, we solve this problem by proximal-point framework
- Approximately solve $\left(\mathbf{v}_{k+1},\alpha_{k+1}\right) \approx \arg\min_{\mathbf{v}}\max_{\alpha} f(\mathbf{v},\alpha)+\frac{\rho}{2}\|\mathbf{v}-\mathbf{v}_k\|^2$

subproblem solver

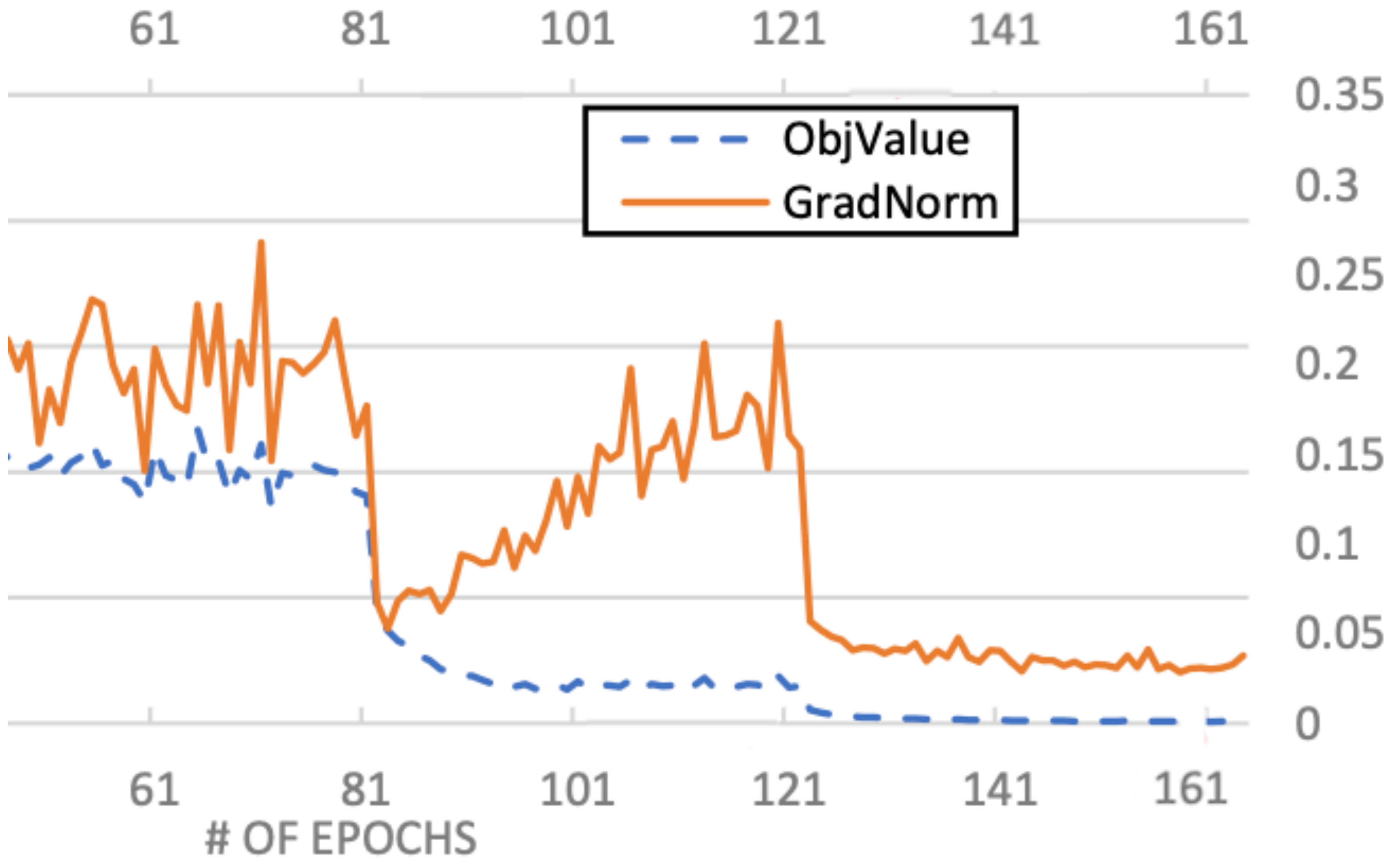
convex-concave

- Intuition (Fixed-point Iterate): $(\mathbf{v}_*,\alpha_*) = \arg\min_{\mathbf{v}}\max_{\alpha} f(\mathbf{v},\alpha)+\frac{\rho}{2}\|\mathbf{v}-\mathbf{v}_*\|^2$

Optimal solution

Property of an Overparameterized NN

- NN has favorable property: Polyak-Lojasiewicz (PL) condition



[Allenzhu-Li-Song'19]

- $\phi(\mathbf{v}) = \max_{\alpha \in \Omega_2} f(\mathbf{v}, \alpha)$, we prove the PL condition holds [L.-Yuan-Ying-Yang, ICLR 20]:

$$\phi(\mathbf{w}) - \min_{\mathbf{w}} \phi(\mathbf{w}) \leq \frac{1}{2\mu} \|\nabla \phi(\mathbf{w})\|^2$$

- PL condition is stronger than QGC in [L.-Zhang-Chen-Wang-Yang, ICML 18], but we do not need convexity

Fast Rate under PL Condition

- [L.-Yuan-Ying-Yang, ICLR 20]

Algorithm 2 Stagewise-PDSG-PL

for $s = 1, \ldots, S$ **do**
  Define $f^{(s)}(\mathbf{v}, \alpha) = f(\mathbf{v}, \alpha) + \rho \|\mathbf{v} - \mathbf{v}^{(s)}\|^2/2$
  $\eta_s \propto \exp(-s), T_s \propto \exp(s)$
  $(\mathbf{v}^{(s+1)}, \alpha^{(s+1)}) = \text{PDSG}(f^{(s)}, \mathbf{v}^{(s)}, \alpha^{(s)}, \eta_s, T_s)$
end for
return $(\mathbf{v}^{(S+1)}, \alpha^{(S+1)})$

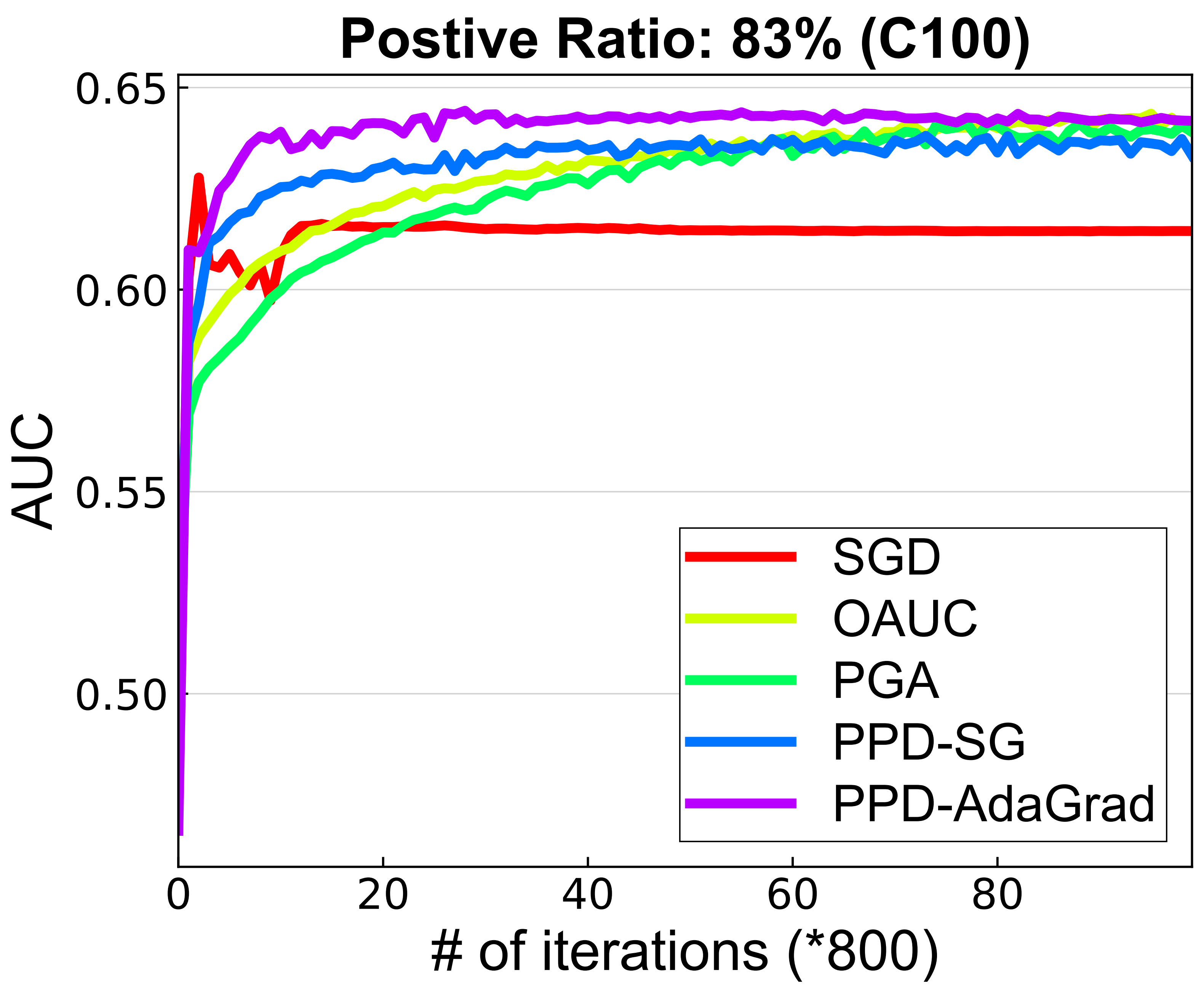
Learning rate

Number of Iterations

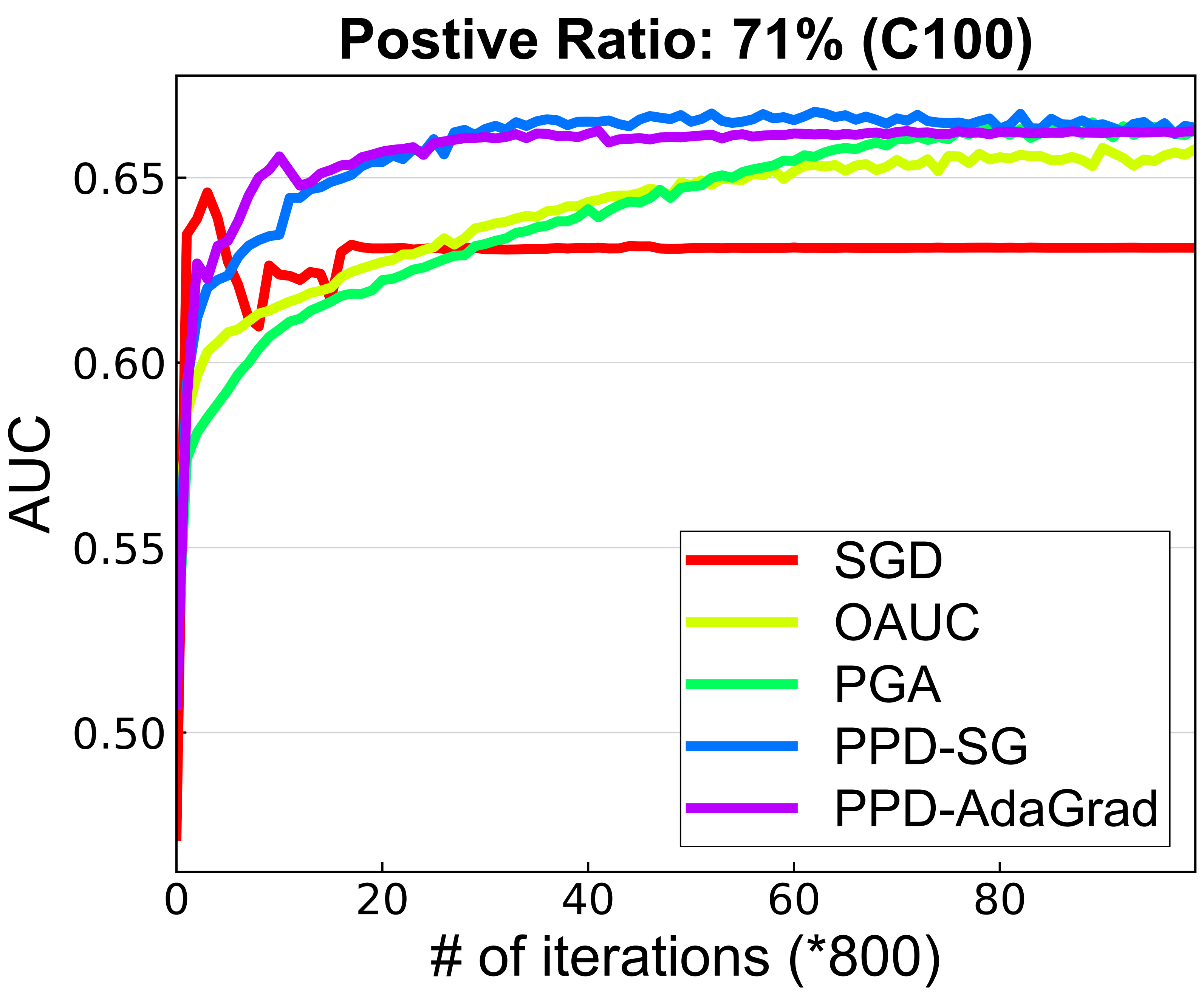
AUC value is ϵ-close to maximal AUC

- $O(1/\mu^2\epsilon)$ complexity for finding ϵ-optimal solution [L.-Yuan-Ying-Yang, ICLR 20]
- The complexity in terms of ϵ is optimal, matching lower bound [Hazan-Kale'11]
- Replace PDSG with other algorithms (e.g., AdaGrad)

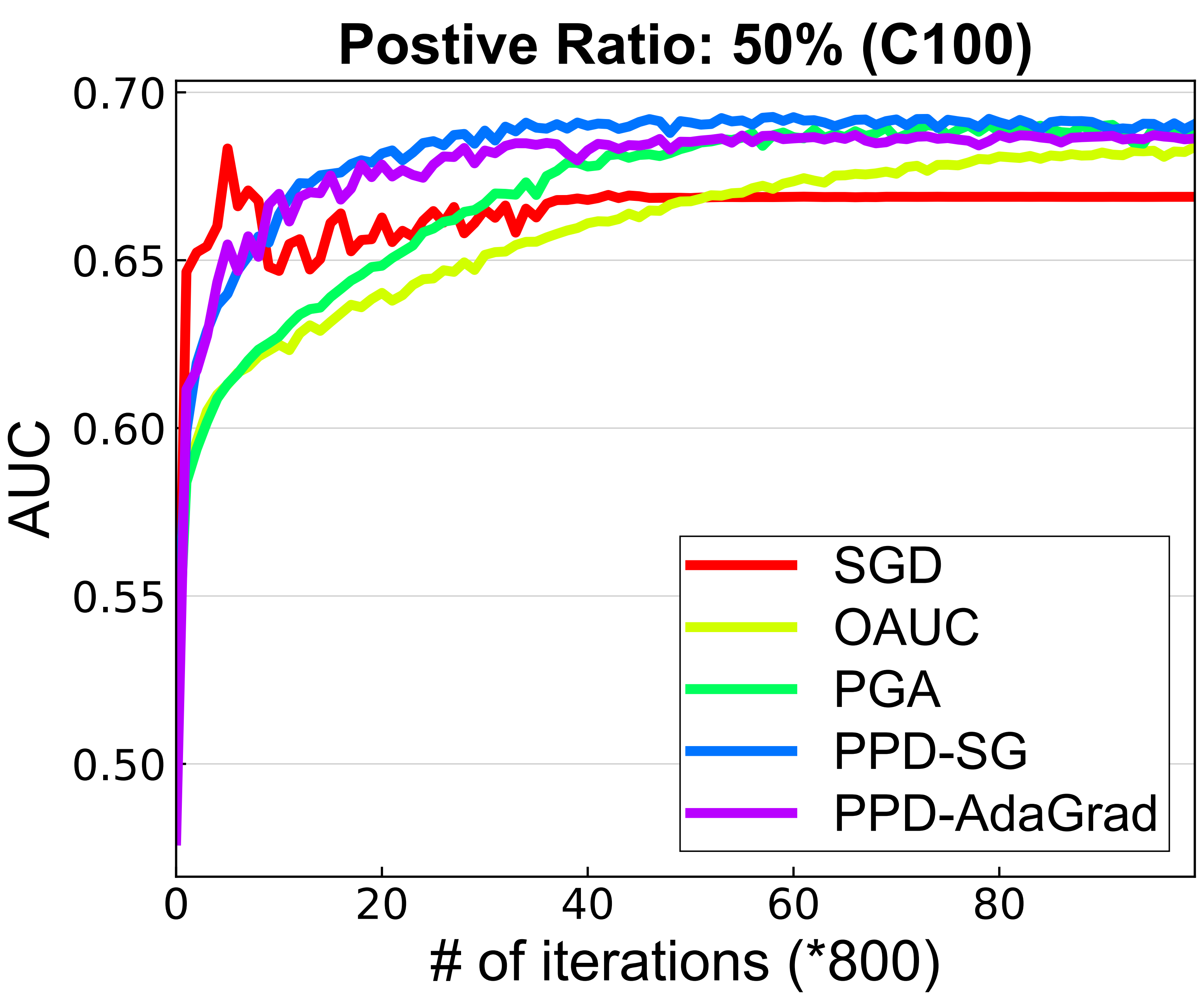
Experiments



Very Imbalanced



Mildly Imbalanced



Balanced

Blue, Purple: our algorithm exploring PL condition [L.-Yuan-Ying-Yang, ICLR 20]

Green: our algorithm without exploring PL condition [Rafique-L.-Lin-Yang, OSM 18]

Red: standard SGD for optimizing cross entropy loss

Acknowledgment

- Collaborators:
  - Zaiyi Chen, Xiaoxuan Zhang, Zhuoning Yuan, Xiaoyu Wang, Yiming Ying, Tianbao Yang

Thank you!

Questions?