Minding the Gaps for Block Frank-Wolfe Optimization of Structured SVMs

Anton Osokin*   Jean-Baptiste Alayrac*

Isabella Lukasewitz   Puneet K. Dokania

Simon Lacoste-Julien

* equal contribution

Bayesian Methods Research Group seminar
Moscow State University – June 9th 2016

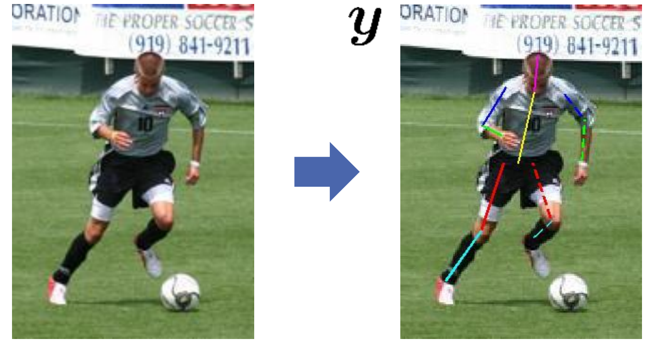
Outline

- Structured Support Vector Machine
- Frank-Wolfe optimization
- Block-Coordinate Frank-Wolfe
- Improving BC-FW:
  - Gap sampling
  - Caching
  - Pairwise and away steps
- Regularization path for SSVM

Structured SVM

- structured prediction:

$\boldsymbol{x}$ $\boldsymbol{y}$



- learn linear classifier:

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \underset{\boldsymbol{y}\in\mathcal{Y}}{\operatorname{argmax}} \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})\rangle \longleftarrow \text{decoding}$$

- structured SVM objective (primal):

structured hinge loss:

$$\min_{\boldsymbol{w}} \quad \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \boxed{\max_{\boldsymbol{y}\in\mathcal{Y}_i} \left\{ L(\boldsymbol{y}_i,\boldsymbol{y}) + \left\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i,\boldsymbol{y})\right\rangle \right\} - \left\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i,\boldsymbol{y}_i)\right\rangle}$$

vs. binary hinge loss: $\max \left\{ 0, 1 - \left\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i)\, \boldsymbol{y}_i \right\rangle \right\}$

Structured SVM

$\boldsymbol{x}$

$\boldsymbol{y}$

- structured prediction:
- learn linear classifier:

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \underset{\boldsymbol{y}\in\mathcal{Y}}{\operatorname{argmax}} \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y}) \rangle \longleftarrow \text{decoding}$$

- structured SVM objective (primal):

$$\min_{\boldsymbol{w}} \quad \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \underbrace{\max_{\boldsymbol{y}\in\mathcal{Y}_i}\left\{L(\boldsymbol{y}_i,\boldsymbol{y}) + \left\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i,\boldsymbol{y})\right\rangle\right\}}_{\text{loss-augmented decoding}} - \left\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i,\boldsymbol{y}_i)\right\rangle$$

- structured SVM dual:

$$\max_{\boldsymbol{\alpha}\in\mathcal{M}} \quad \boldsymbol{b}^T\boldsymbol{\alpha} - \frac{\lambda}{2}\|A\boldsymbol{\alpha}\|^2$$

-> exponential number of variables!

$$\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \ldots \times \Delta_{|\mathcal{Y}_n|}$$

$$A := \left[\frac{1}{\lambda n}\left(\boldsymbol{\phi}(\boldsymbol{x}_i,\boldsymbol{y}_i) - \boldsymbol{\phi}(\boldsymbol{x}_i,\boldsymbol{y})\right) \in \mathbb{R}^d\right]$$

$$\boldsymbol{b} := \left(\frac{1}{n}L_i(\boldsymbol{y})\right)_{i\in[n],\boldsymbol{y}\in\mathcal{Y}_i}$$

- primal-dual pair: $\boldsymbol{w}^* = A\boldsymbol{\alpha}^*$

Structured SVM optimization

$$\min_{\boldsymbol{w}} \quad \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \max_{\boldsymbol{y}\in\mathcal{Y}_i} \left\{ L(\boldsymbol{y}_i,\boldsymbol{y}) + \left\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i,\boldsymbol{y}) \right\rangle \right\} - \left\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i,\boldsymbol{y}_i) \right\rangle$$

suboptimality after K passes through data:

- popular approaches:
  - stochastic subgradient method [Ratliff et al. 07, Shalev-Shwartz et al. 10] — $O\left(\frac{1}{nK}\right)$
    - pros: online!
    - cons: sensitive to step-size; don't know when to stop
  - cutting plane method (SVMstruct) [Tsochantaridis et al. 05, Joachims et al. 09] — $O\left(\frac{1}{K}\right)$
    - pros: automatic step-size; duality gap
    - cons: batch! -> slow for large n

- block-coordinate Frank-Wolfe on dual [Lacoste-Julien et al. 13] — $O\left(\frac{1}{nK}\right)$

-> combines best of both worlds:

  - online
  - automatic step-size via analytic line search
  - duality gap
  - rates also hold for approximate oracles

Frank-Wolfe algorithm [Frank, Wolfe 1956]

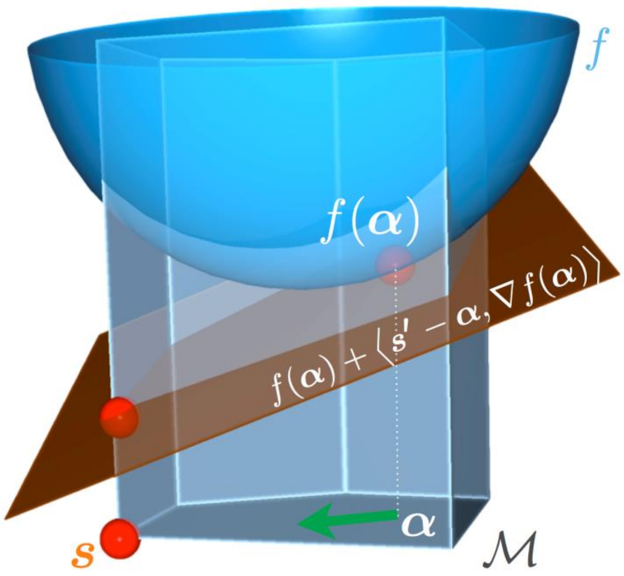
(aka conditional gradient)

- constrained optimization: $\min_{\boldsymbol{\alpha} \in \mathcal{M}} f(\boldsymbol{\alpha})$

where:

f convex & cts. differentiable

$\mathcal{M}$ convex & compact



- FW algorithm – repeat:

1) Find a good feasible direction by minimizing linearization of f:

$$\boldsymbol{s}_{t+1} \in \arg\min_{\boldsymbol{s}' \in \mathcal{M}} \langle \boldsymbol{s}', \nabla f(\boldsymbol{\alpha}_t) \rangle$$

2) Take a convex step in the direction:

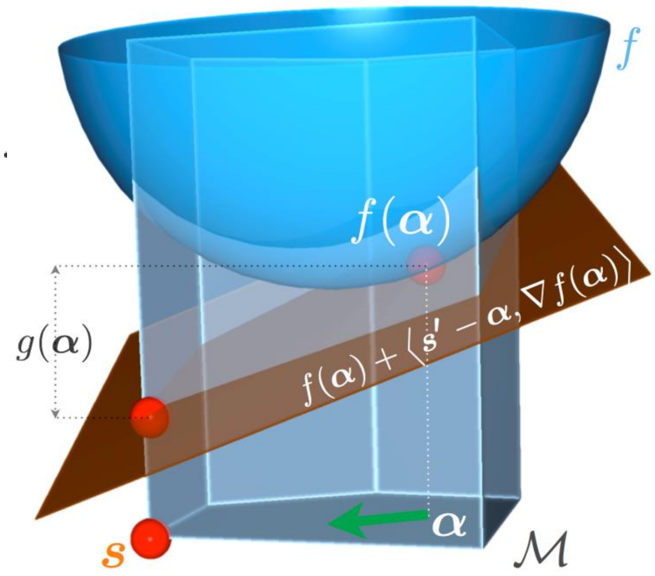
$$\boldsymbol{\alpha}_{t+1} = (1 - \gamma_t)\, \boldsymbol{\alpha}_t + \gamma_t\, \boldsymbol{s}_{t+1}$$

- Properties: O(1/T) rate
  - sparse iterates
  - get duality gap $g(\boldsymbol{\alpha})$ for free
  - affine invariant
  - rate holds even if linear subproblem solved **approximately**

Frank-Wolfe gap

- FW gap is free:

$$g(\boldsymbol{\alpha}) := \max_{\boldsymbol{s}' \in \mathcal{M}} \langle \boldsymbol{\alpha} - \boldsymbol{s}', \nabla f(\boldsymbol{\alpha}) \rangle = \langle \boldsymbol{\alpha} - \boldsymbol{s}, \nabla f(\boldsymbol{\alpha}) \rangle.$$



- FW algorithm – repeat:

1) Find a good feasible direction by minimizing linearization of f:

$$\boldsymbol{s}_{t+1} \in \arg\min_{\boldsymbol{s}' \in \mathcal{M}} \langle \boldsymbol{s}', \nabla f(\boldsymbol{\alpha}_t) \rangle$$

2) Take a convex step in the direction:

$$\boldsymbol{\alpha}_{t+1} = (1 - \gamma_t)\, \boldsymbol{\alpha}_t + \gamma_t\, \boldsymbol{s}_{t+1}$$

- Properties: O(1/T) rate
  - sparse iterates
  - get duality gap $g(\boldsymbol{\alpha})$ for free
  - affine invariant
  - rate holds even if linear subproblem solved **approximately**

Frank-Wolfe for SSVM [Lacoste-Julien et al., 2013]

- structured SVM dual: $-\min_{\boldsymbol{\alpha}\in\mathcal{M}} f(\boldsymbol{\alpha})$ $\quad f(\boldsymbol{\alpha}) := \frac{\lambda}{2}\|A\boldsymbol{\alpha}\|^2 - \boldsymbol{b}^T\boldsymbol{\alpha}$

$$\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \ldots \times \Delta_{|\mathcal{Y}_n|}$$

use primal-dual link: $\boldsymbol{w}_t = A\boldsymbol{\alpha}_t$

- FW algorithm – repeat:

key insight:

1) Find good feasible direction by minimizing linearization of f:	loss-augmented decoding on each example i
$s_{t+1} \in \arg\min_{s'\in\mathcal{M}} \langle s', \nabla f(\boldsymbol{\alpha}_t)\rangle$	$\arg\max_{\boldsymbol{y}\in\mathcal{Y}_i} \left\{ L(\boldsymbol{y}_i, \boldsymbol{y}) + \left\langle \boldsymbol{w}_t, \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}) \right\rangle \right\}$

2) Take a convex step in the direction: becomes a batch subgradient step:

$$\boldsymbol{\alpha}_{t+1} = (1-\gamma_t)\,\boldsymbol{\alpha}_t + \gamma_t\, s_{t+1} \qquad \boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \lambda\,\gamma_t\,\boldsymbol{d}_{sub}$$

choose by **analytic** line search on quadratic dual $f(\boldsymbol{\alpha})$

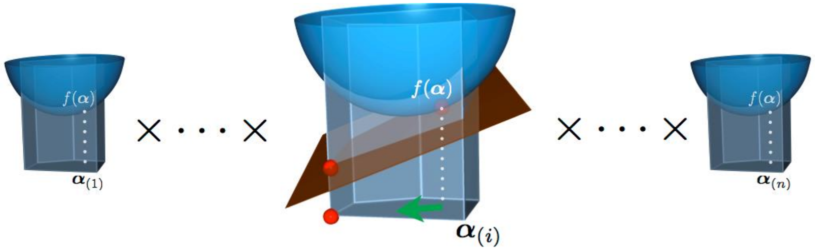
Block-Coordinate Frank-Wolfe

[Lacoste-Julien et al. 13]

- for constrained optimization over compact **product domain**:

$$\min_{\boldsymbol{\alpha} \in \mathcal{M}^{(1)} \times \ldots \times \mathcal{M}^{(n)}} f(\boldsymbol{\alpha})$$

$$\boldsymbol{\alpha} = (\boldsymbol{\alpha}_{(1)}, \ldots, \boldsymbol{\alpha}_{(n)})$$



- pick i at random; update only block i with a FW step:

$$\boldsymbol{s}_{(i)} = \operatorname*{argmin}_{\boldsymbol{s}'_{(i)} \in \mathcal{M}^{(i)}} \left\langle \boldsymbol{s}'_{(i)}, \nabla_{(i)} f(\boldsymbol{\alpha}^{(t)}) \right\rangle$$

$$\boldsymbol{\alpha}_{(i)}^{(t+1)} = (1-\gamma)\boldsymbol{\alpha}_{(i)}^{(t)} + \gamma \boldsymbol{s}_{(i)}$$

- same O(1/T) rate as batch FW

  -> each step **n times cheaper** though

  -> constant can be the same (SVM e.g.)

- Properties: O(1/T) rate
  - sparse iterates
  - get duality gap guarantees
  - affine invariant
  - rate holds even if linear subproblem solved **approximately**

Block-Coordinate Frank-Wolfe

[Lacoste-Julien et al. 13]

- for constrained optimization over compact **product domain**:

$$\min_{\boldsymbol{\alpha} \in \mathcal{M}^{(1)} \times \ldots \times \mathcal{M}^{(n)}} f(\boldsymbol{\alpha})$$

$$\boldsymbol{\alpha} = (\boldsymbol{\alpha}_{(1)}, \ldots, \boldsymbol{\alpha}_{(n)})$$

structured SVM:

$$f(\boldsymbol{\alpha}) := \tfrac{\lambda}{2} \|A\boldsymbol{\alpha}\|^2 - \boldsymbol{b}^T \boldsymbol{\alpha}$$

$$\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \ldots \times \Delta_{|\mathcal{Y}_n|}$$

- pick i at random; update only block i with a FW step:

$$\boldsymbol{s}_{(i)} = \operatorname*{argmin}_{\boldsymbol{s}'_{(i)} \in \mathcal{M}^{(i)}} \left\langle \boldsymbol{s}'_{(i)}, \nabla_{(i)} f(\boldsymbol{\alpha}^{(t)}) \right\rangle$$

$$\operatorname*{arg\,max}_{\boldsymbol{y} \in \mathcal{Y}_i} \left\{ L(\boldsymbol{y}_i, \boldsymbol{y}) + \left\langle \boldsymbol{w}_t, \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}) \right\rangle \right\}$$

loss-augmented decoding

$$\boldsymbol{\alpha}_{(i)}^{(t+1)} = (1 - \gamma)\boldsymbol{\alpha}_{(i)}^{(t)} + \gamma \boldsymbol{s}_{(i)}$$

- same O(1/T) rate as batch FW

  -> each step **n times cheaper** though

  -> constant can be the same (SVM e.g.)

Key insight: separable FW gap

- Frank-Wolfe gap

$$g(\boldsymbol{\alpha}) := \max_{\boldsymbol{s}\in\mathcal{M}} \langle \boldsymbol{\alpha} - \boldsymbol{s}, \nabla f(\boldsymbol{\alpha})\rangle$$

can be written as a sum of block gaps

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^{n} g_i(\boldsymbol{\alpha})$$

where

$$g_i(\boldsymbol{\alpha}) := \max_{\boldsymbol{s}_{(i)}\in\mathcal{M}^{(i)}} \left\langle \boldsymbol{\alpha}_{(i)} - \boldsymbol{s}_{(i)}, \nabla_{(i)} f(\boldsymbol{\alpha})\right\rangle$$

- block gap represents suboptimality at one block
- can use block gaps to **adaptively** adjust the algorithm

Contributions

- Improving BC-FW:
  - Gap sampling
  - Caching
  - Pairwise and away steps

- Regularization path for SSVM

Gap sampling (new!)

- We can use block gaps to adaptively pick an object for the next iteration
- Multiple schemes possible:
  - pick the object with largest gap (deterministic)
  - sampling with probabilities proportional to block gaps (or squares?)

- More adaptive than sampling proportional to Lipschitz constants

  [Nesterov, 2012; Needell et al., 2014; Zhao & Zhang, 2015]

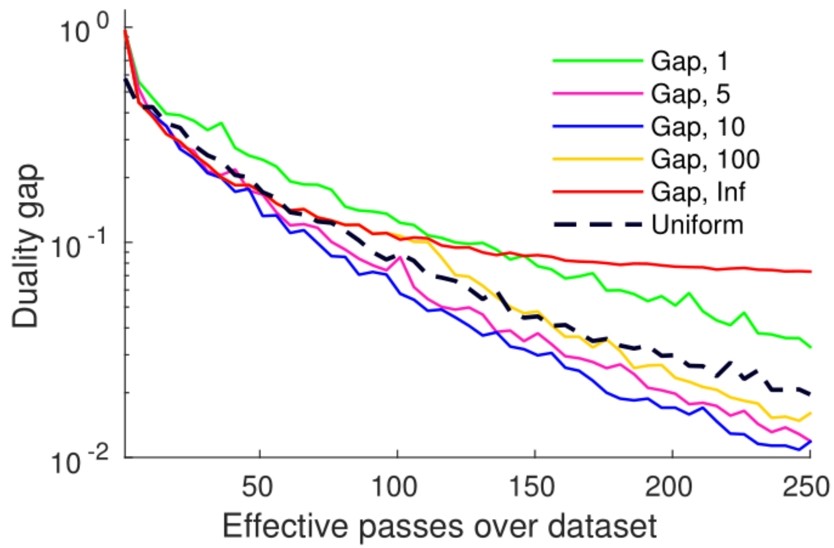
- We are aware of only one adaptive sampling method:

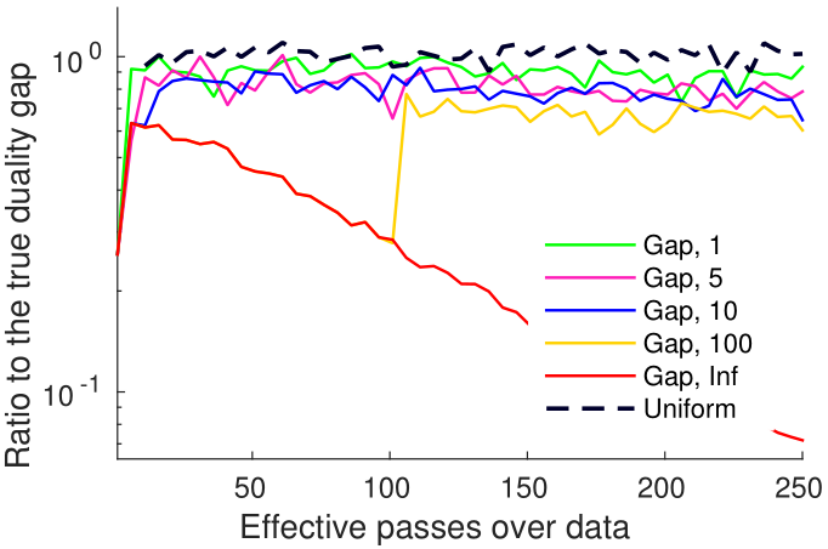
  [Csiba et al. (2015)] in the context of SDCA

Exploitation vs. staleness trade-off

- When selecting objects all the other gaps become outdated (stale)
- If using very stale gaps, the gap estimates become bad
- To compensate, we can recomputed the true gap after every X passes over the dataset

Illustrative experiment on OCR dataset:





Gap sampling: theoretical result

- If we sample objects proportional to the **exact** block gaps then convergence rate $O(1/k)$ is multiplied by a constant depending on the non-uniformity of the gaps and the non-uniformity of the (unknown) curvature constants.

- In the best case (curvature constants are uniform, gaps are non-uniform), gap sampling is $n\sqrt{n}$ times faster
- In the worst case (curvature constants are non-uniform, gaps are uniform), gap sampling is $\sqrt{n}$ times slower
- If gaps are moderately non-uniform gap sampling is always faster

- Open problem: how to analyze the staleness effect?

Caching oracle calls (new!)

- The oracle might be the bottleneck of the algorithm
- We can cache the output of the oracle and reuse them

  same idea was used in 1-slack cutting plane [Joachims et al. 09]

- Instead of the oracle we call a **cache oracle**

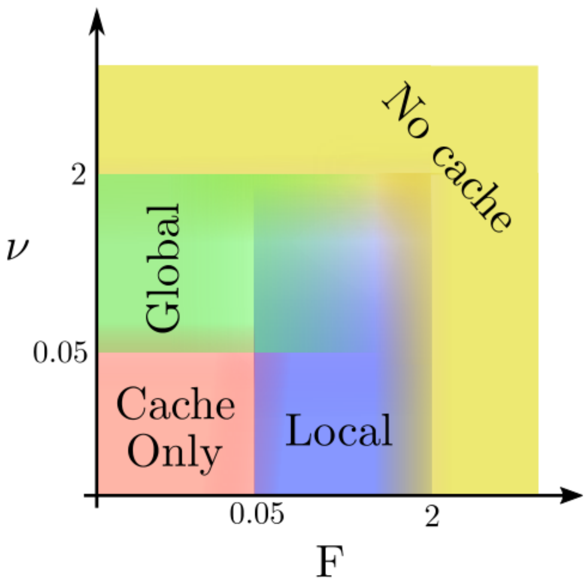
$$\boldsymbol{y}_i^c := \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{C}_i} \left\{ L_i(\boldsymbol{y}) + \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}) \rangle \right\}$$

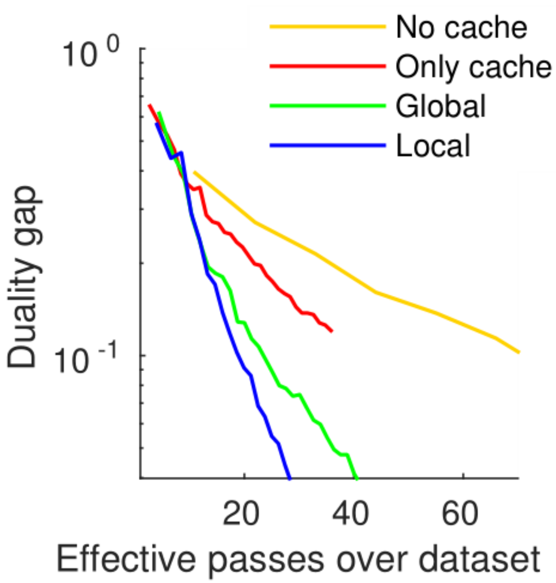
- If the cache corner can give enough improvement use it
- Adaptive criterion for cache hit:

$$\widehat{g}_{\text{cache}} \geq \max\left(F g_i^{(k_i)}, \frac{\nu}{n} g^{(k_0)}\right)$$

Caching oracle calls (new!)

- Cache regimes:



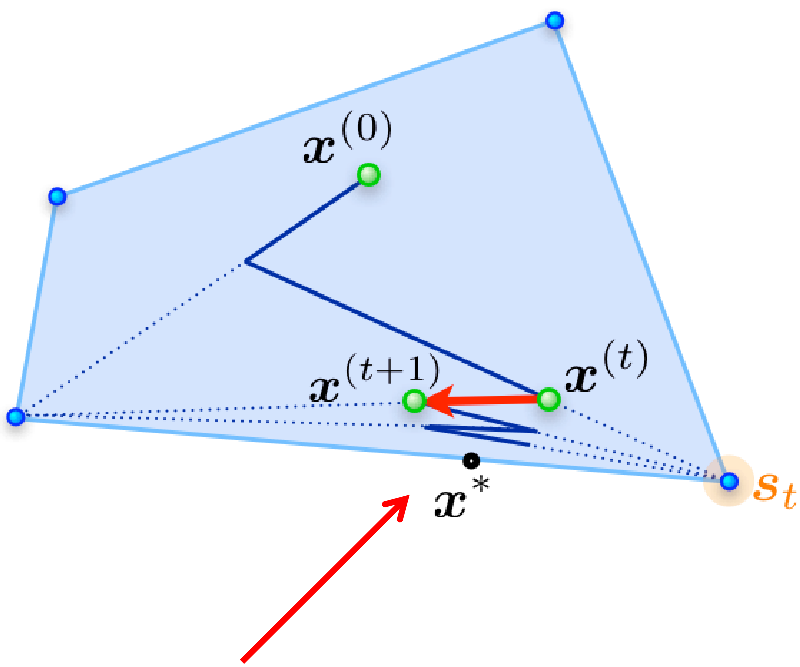


- With global cache criterion we can prove convergence
- Open problem: convergence rate based on the local criterion

Pairwise and away steps (new!)

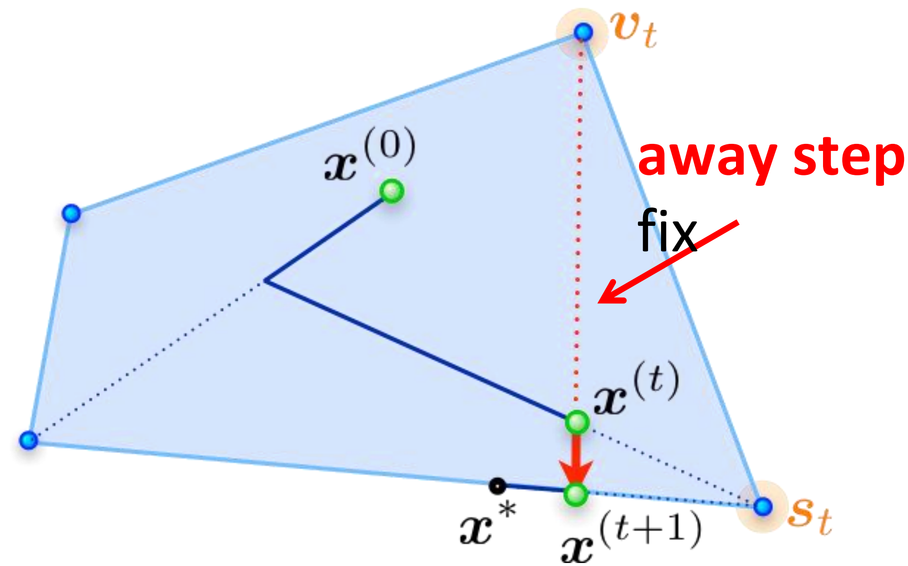
Slow convergence of Frank-Wolfe…

standard FW



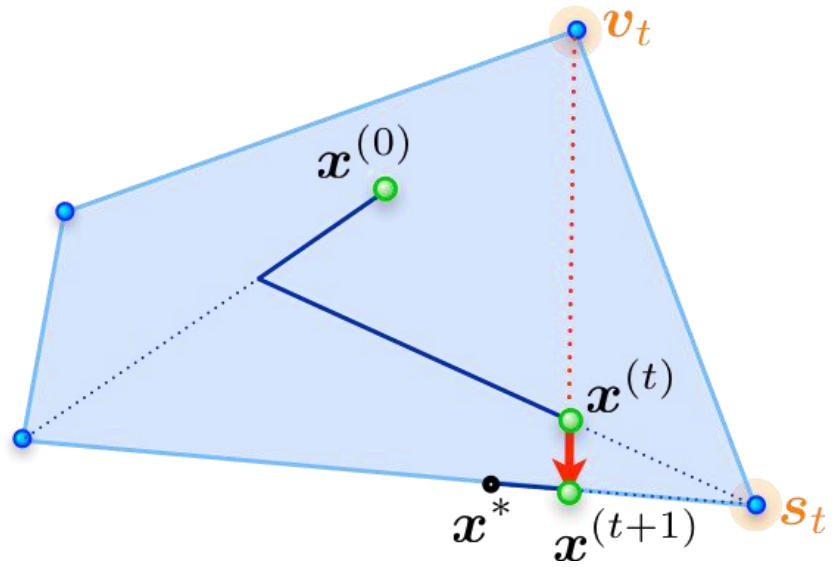
zig-zagging problem for FW

away-step FW

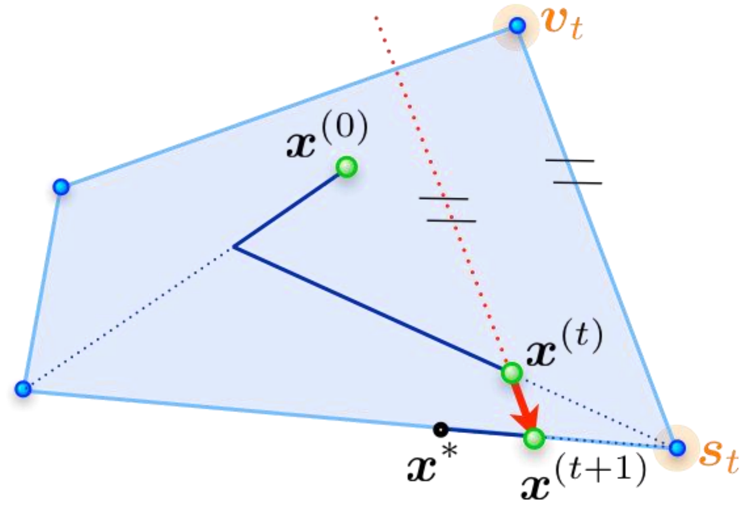


see [Lacoste-Julien & Jaggi 15]

Variants with linear convergence

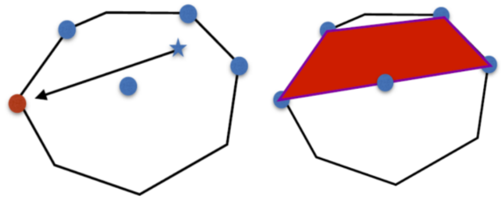


away-step FW



pairwise FW

- fully-corrective FW (FC-FW): re-optimize over convex hull of previously found vertices (correction polytope)



see [Lacoste-Julien & Jaggi 15]

Block-Coordinate versions (new!)

- We propose Pairwise and Away variants for BC-FW
- Algorithm BC-PFW (pairwise steps)
  - Pick FW corner
    $$\arg\max_{\boldsymbol{y} \in \mathcal{Y}_i} \left\{ L(\boldsymbol{y}_i, \boldsymbol{y}) + \left\langle \boldsymbol{w}_t, \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}) \right\rangle \right\}$$
  - Pick Away corner
    $$\arg\min_{\boldsymbol{y} \in \mathcal{S}_i} \left\{ L(\boldsymbol{y}_i, \boldsymbol{y}) + \left\langle \boldsymbol{w}_t, \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}) \right\rangle \right\}$$
  - Analytic line search
  - Move mass from Away corner to FW corner

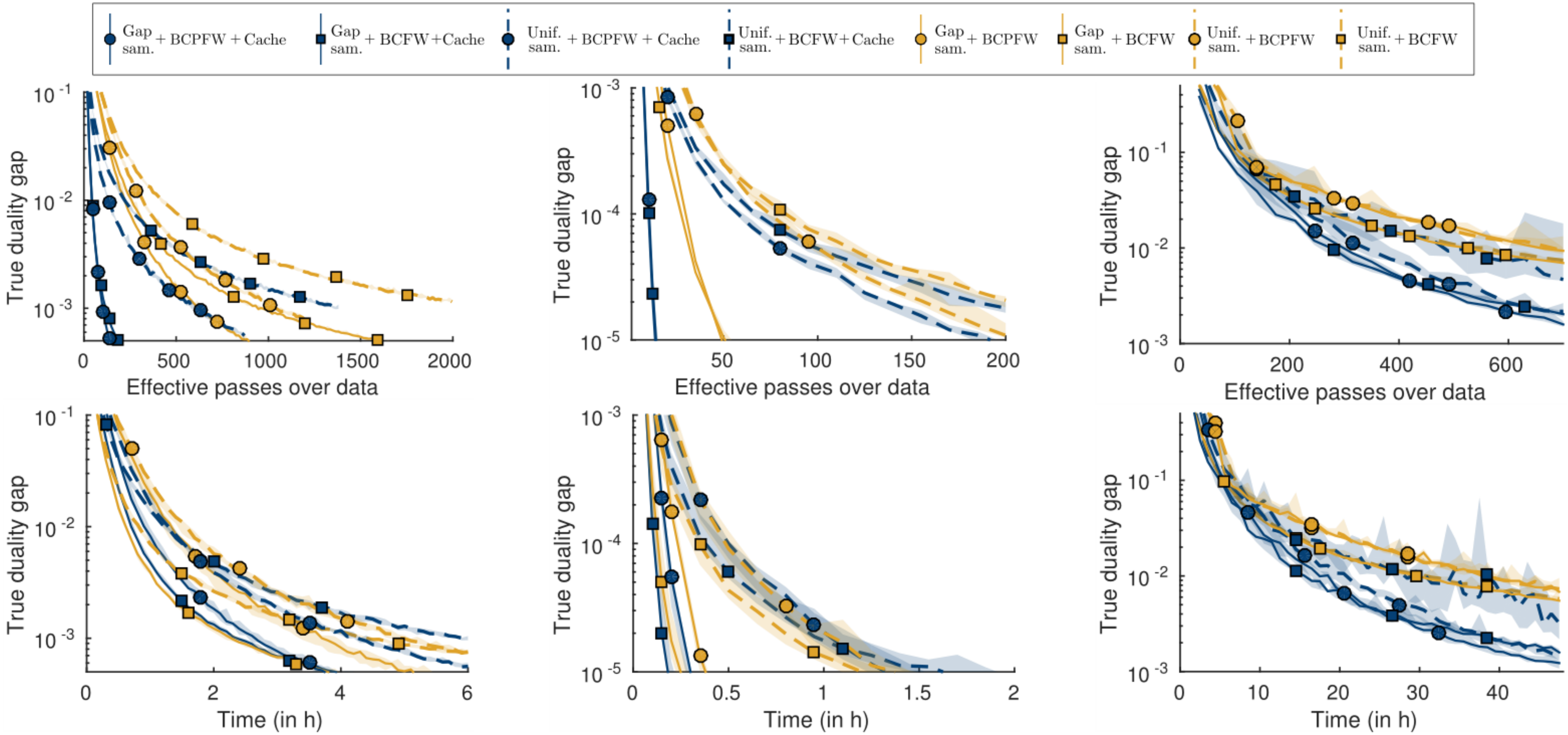
- Catch: need to maintain dual variables, but similar to cache
- Bad news: do not have satisfying theoretical results
- Good news: observe linear convergence in some cases

Comparing different variants

- 8 methods:
  - gap sampling / uniform sampling
  - caching / no caching
  - BC-FW / BC-PFW (pairwise steps)
- 4 structured prediction datasets:
  - OCR – character recognition
  - CoNLL – text chunking
  - HorseSeg (3 sizes) – binary image segmentation
  - LSP – human pose estimation
- 3 values of regularization parameter: good, too big, too small

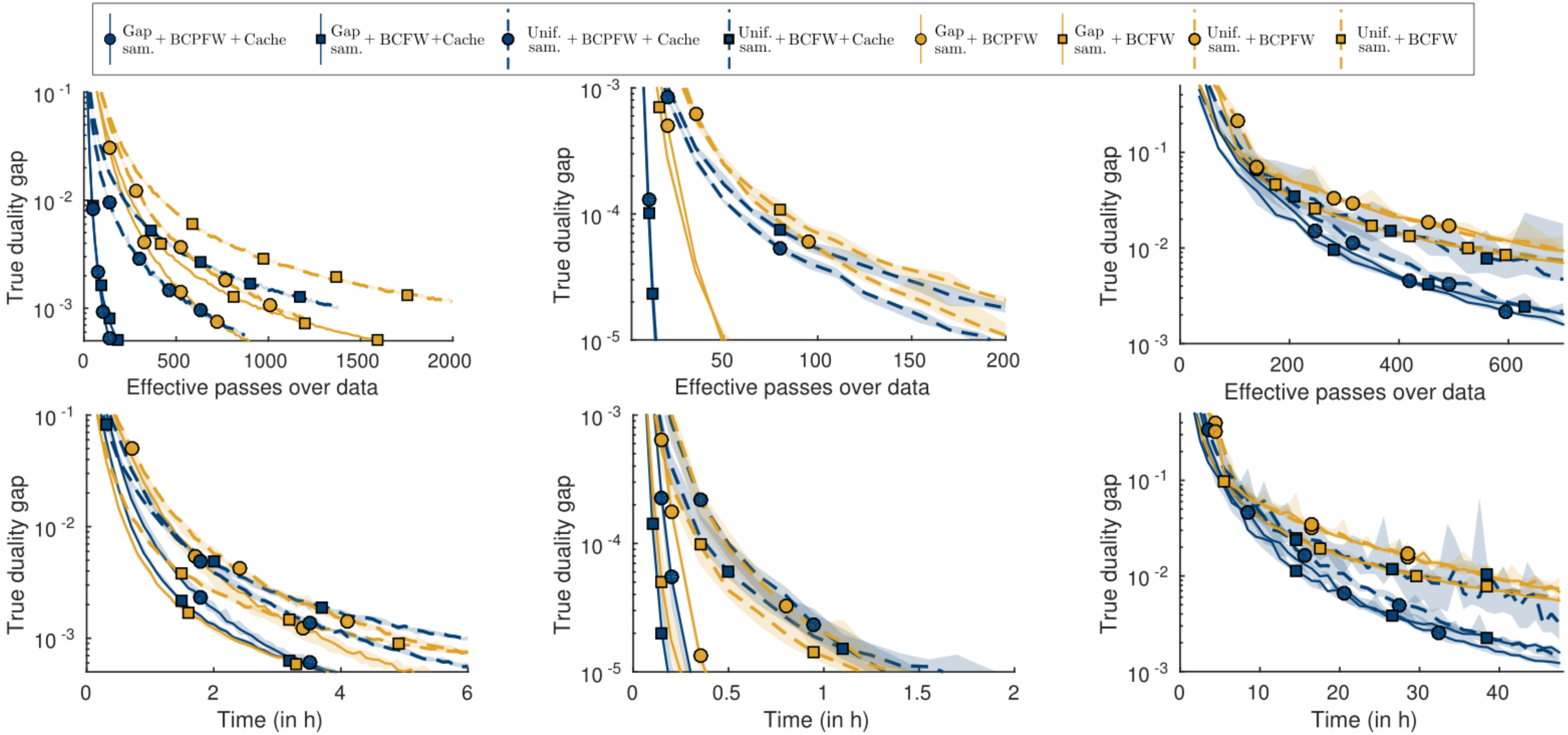
- 2 pages of plots 🙂

Comparing different variants



(a) OCR-large, $\lambda = 0.001$ (b) HorseSeg-medium, $\lambda = 10$ (c) LSP-small, $\lambda = 100$

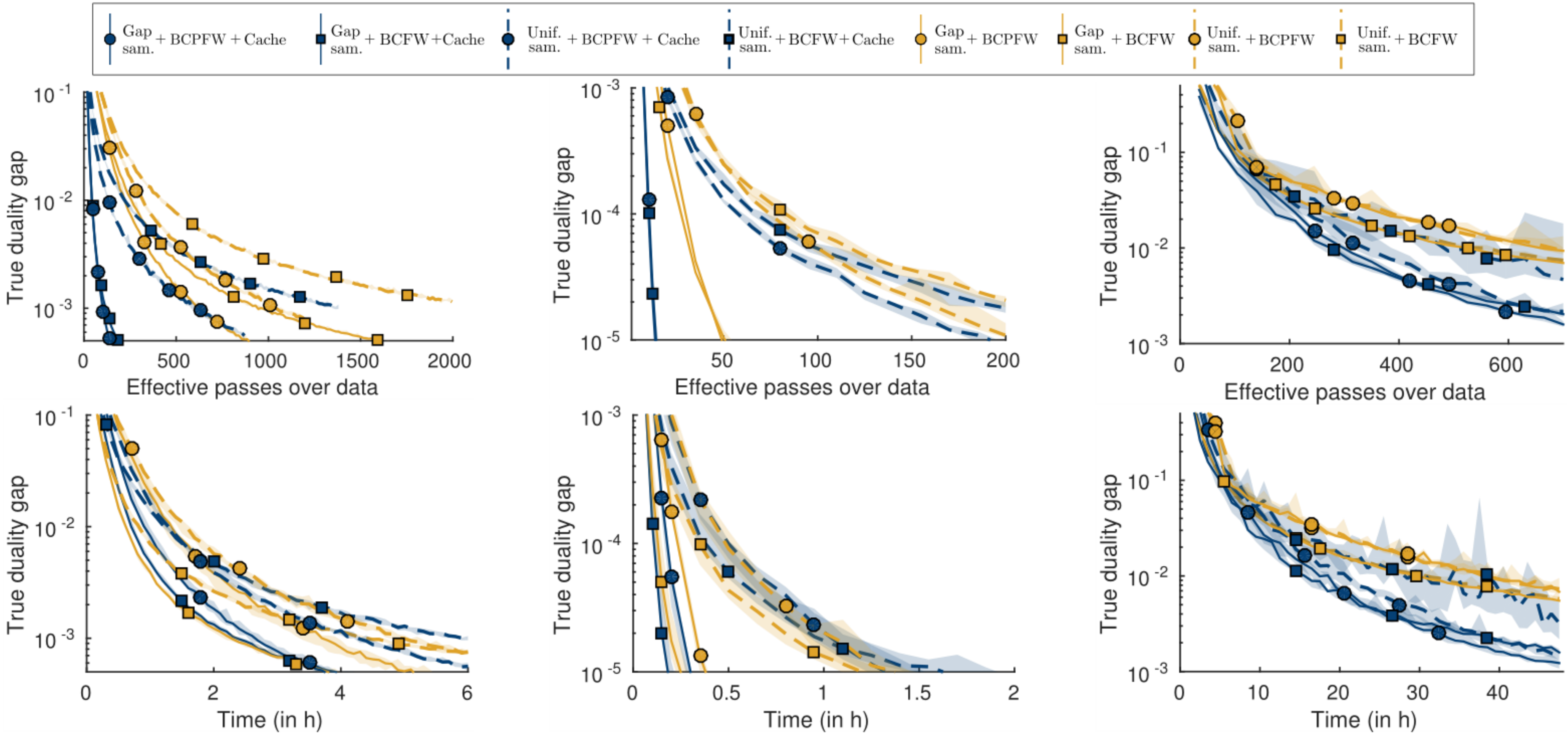
Comparing different variants



(a) OCR-large, $\lambda = 0.001$ (b) HorseSeg-medium, $\lambda = 10$ (c) LSP-small, $\lambda = 100$

Conclusion 1: gap sampling always helps! (solid vs. dashed)

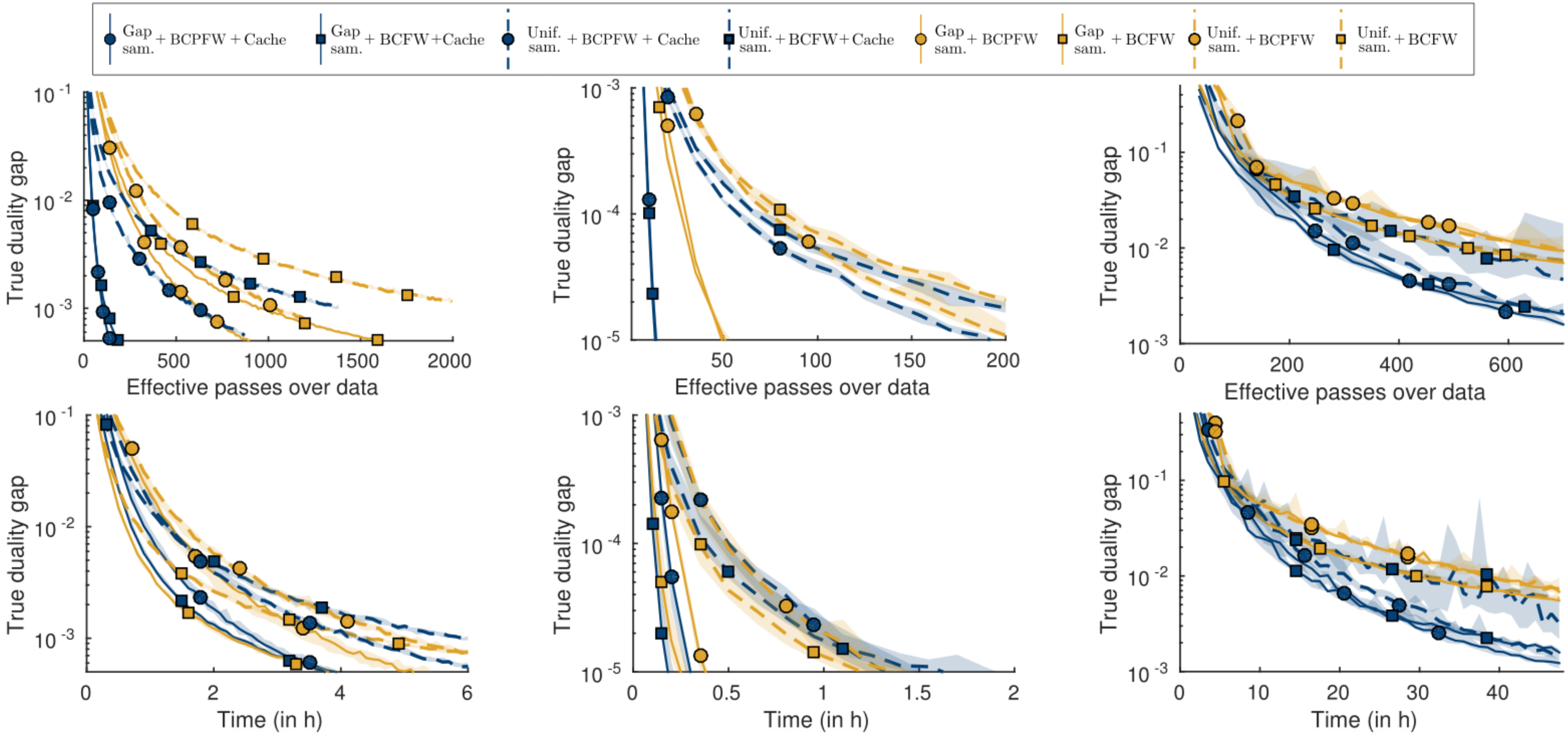
Comparing different variants



(a) OCR-large, $\lambda = 0.001$ (b) HorseSeg-medium, $\lambda = 10$ (c) LSP-small, $\lambda = 100$

Conclusion 2: caching always helps in the number of oracle calls (blue vs. yellow). If oracle is fast, caching can even hurt because of overheads. If oracle is slow, caching is a must!

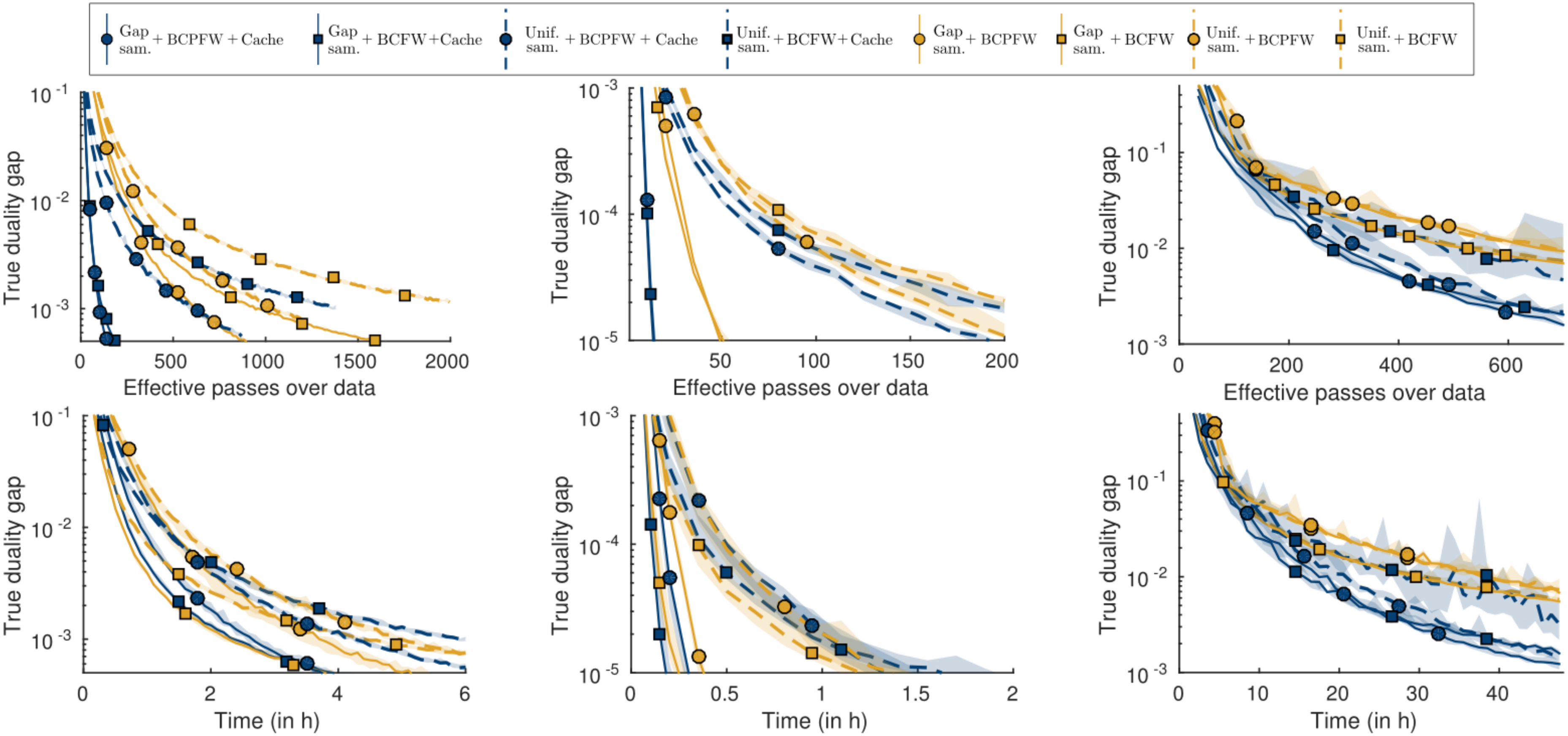
Comparing different variants



(a) OCR-large, $\lambda = 0.001$ (b) HorseSeg-medium, $\lambda = 10$ (c) LSP-small, $\lambda = 100$

Conclusion 3: pairwise steps help reaching high accuracy.
The effect is stronger if the problem is more strongly convex.

Comparing different variants



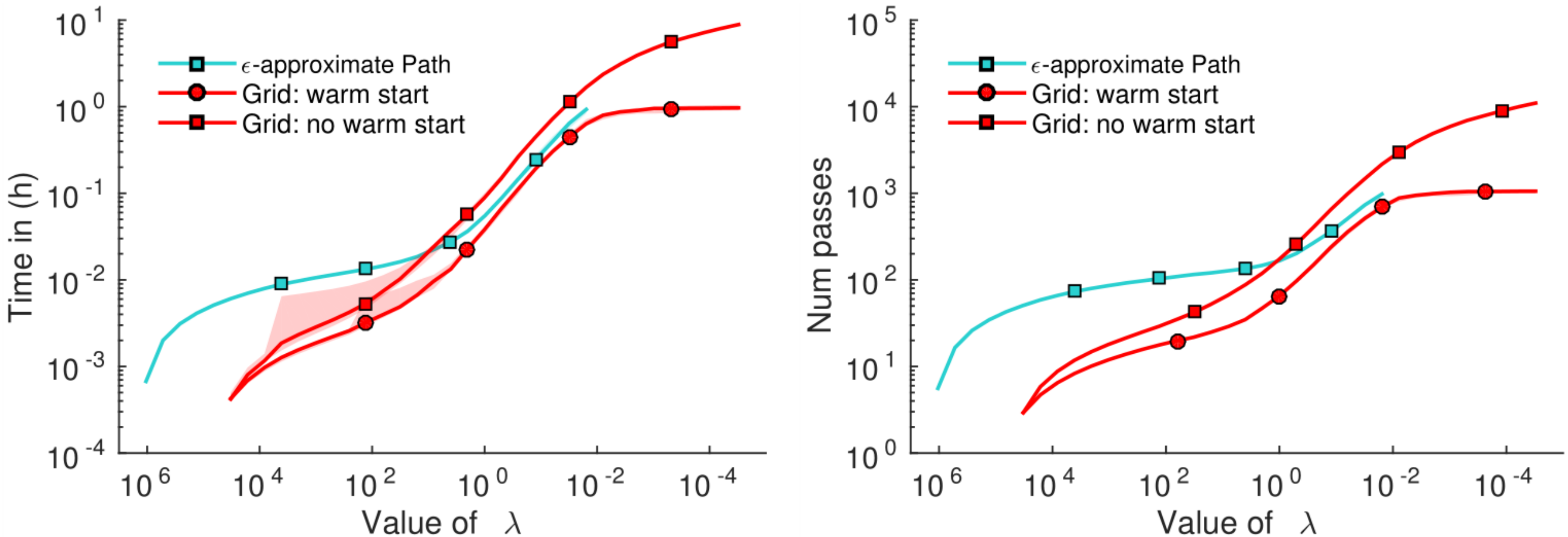
(a) OCR-large, $\lambda = 0.001$ (b) HorseSeg-medium, $\lambda = 10$ (c) LSP-small, $\lambda = 100$

Recommendation: use (a) BC-PFW + gap sampling + caching or (b) BC-FW + gap sampling

Regularization path (new!)

- Regularization path = solving the problem for all possible values of regularization parameter
- Better than the grid search, but usually expensive
- Exact paths are unstable and often intractable
- We construct an ε-approximate regularization path
- We use piecewise constant approximation except the first piece
- Algorithm:
  1. Initialization: construct the largest breakpoint
  2. At a breakpoint, construct the next one such that the gap is smaller than ε
  3. Optimize with any solver to get gap of κε, κ < 1 (to make a step)
  4. Repeat steps 2 and 3 until convergence

Regularization path: results



- We can compute the full path for smaller datasets:

HorseSeg-small and OCR-small

- For larger datasets both grid search and paths exceed time limits

Contributions

- Improvements over BCFW:
  - adaptive non-uniform sampling of the training objects
  - gap-based criterion for caching the oracle calls
  - pairwise and away steps in the block-coordinate setting

- Regularization path for SSVM.

Key insight: adaptivity via using the gaps