10-708 Probabilistic Graphical Models

Machine Learning Department
School of Computer Science
Carnegie Mellon University

MAP Inference by MILP

+

Structured SVM

Matt Gormley
Lecture 9
Mar. 1, 2021

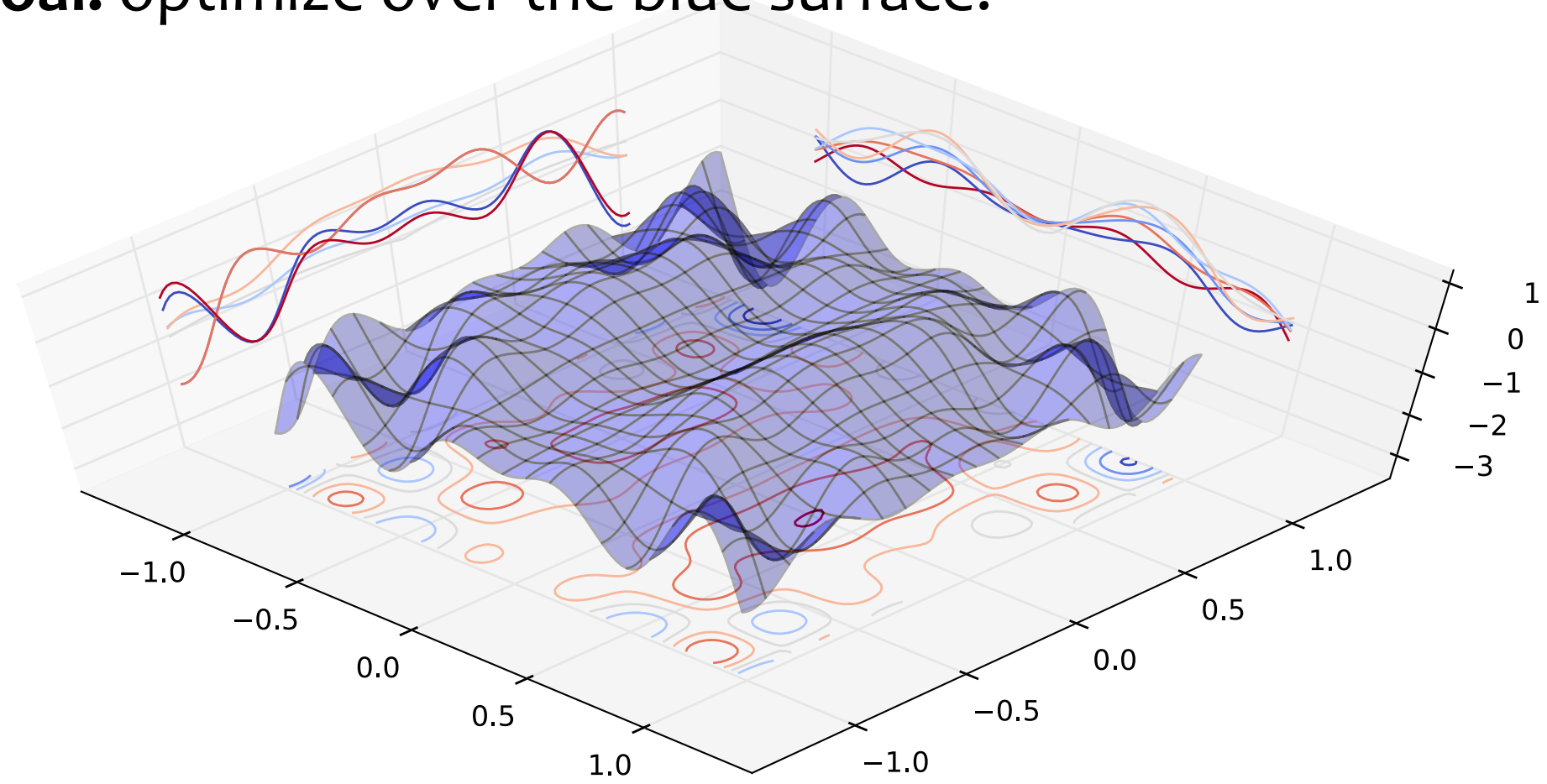
Reminders

- Quiz 1:
  - Fri, Mar. 05 during class time
  - Material: Lectures 1 – 8 only
  - Logistics: see forthcoming Piazza post
- Homework 2: Exact inference and supervised learning (CRF+RNN)
  - Out: Wed, Feb. 24
  - Due: Wed, Mar. 10 at 11:59pm

BRANCH AND BOUND

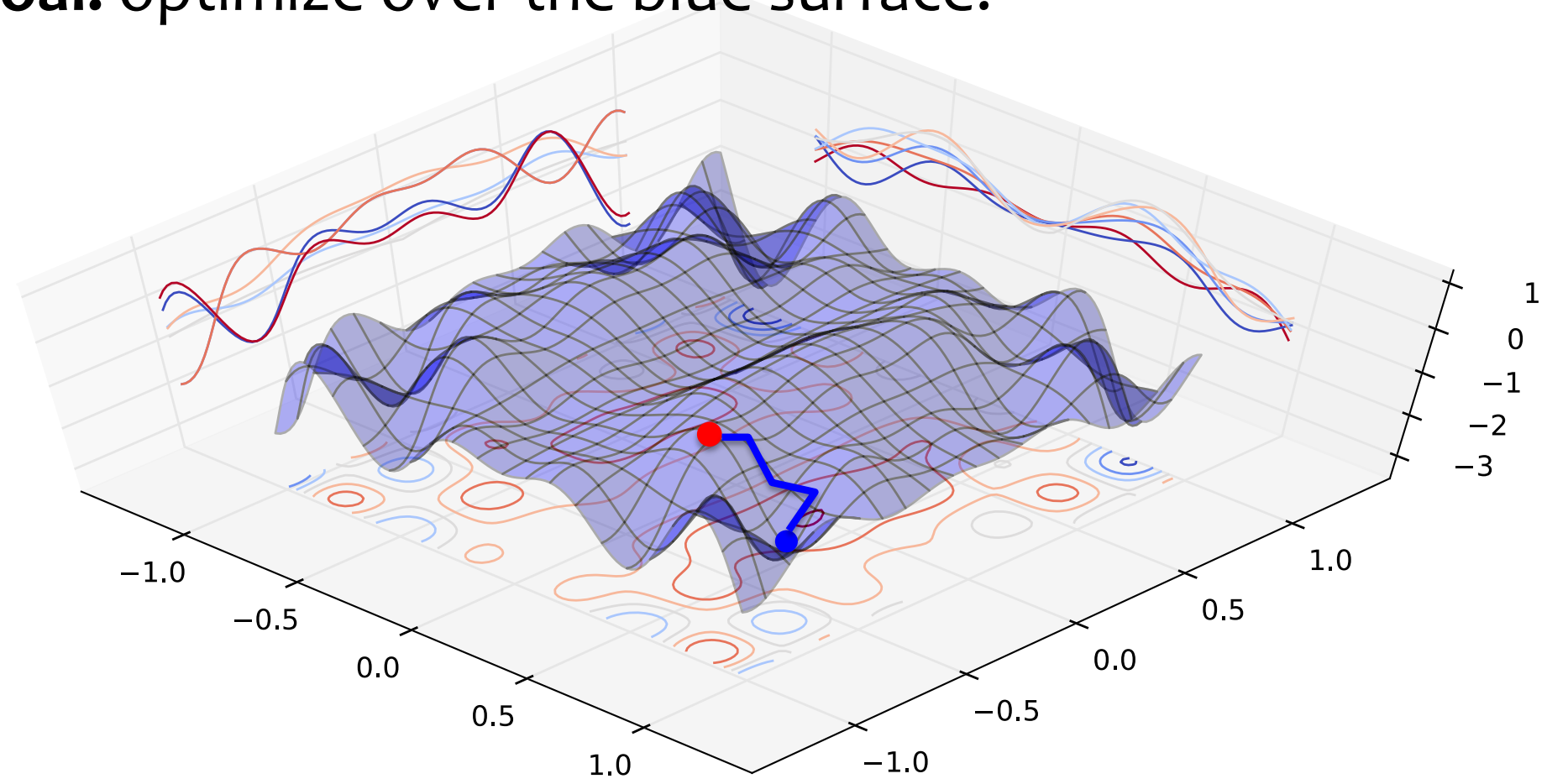
Background: Nonconvex Global Optimization

Goal: optimize over the blue surface.



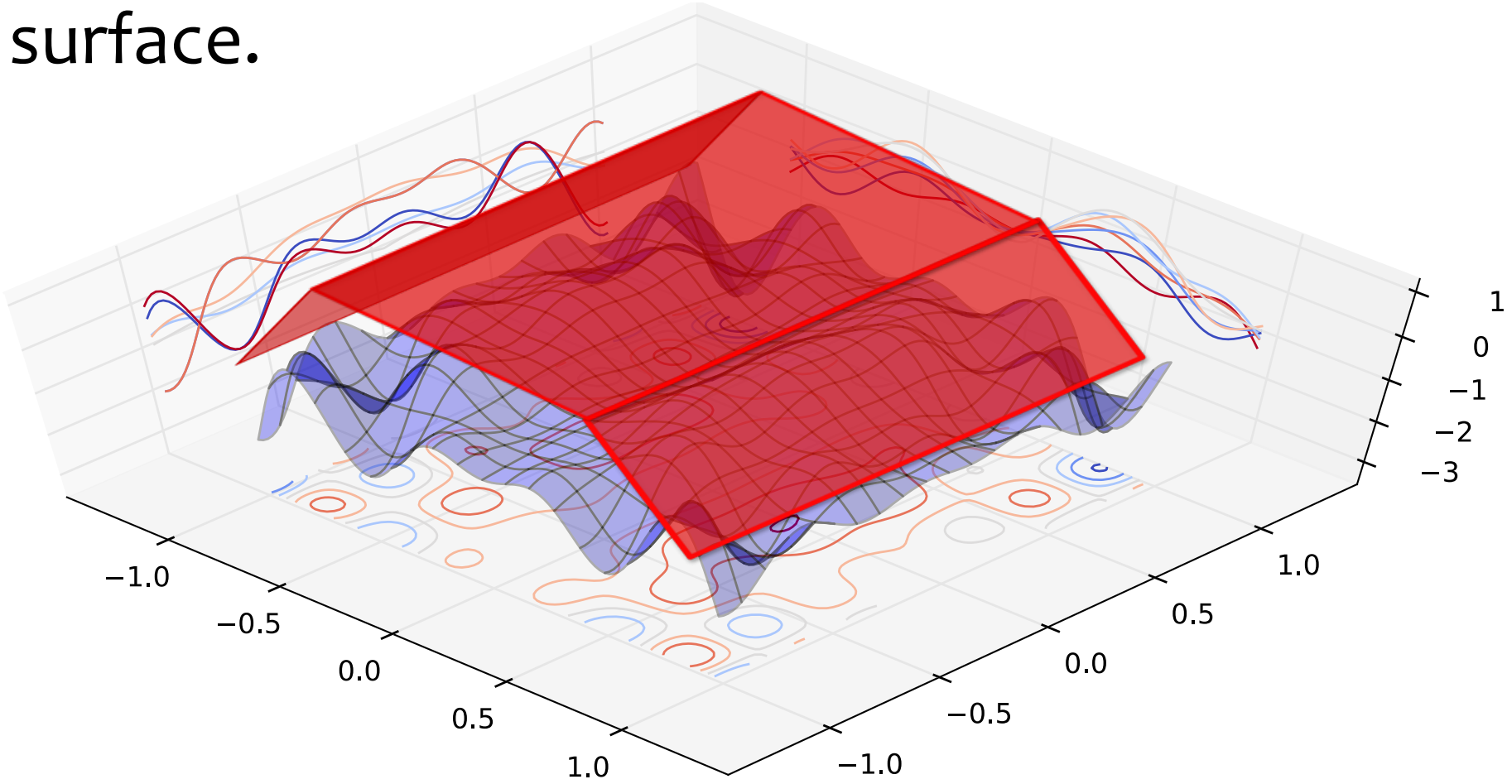
Background: Nonconvex Global Optimization

Goal: optimize over the blue surface.



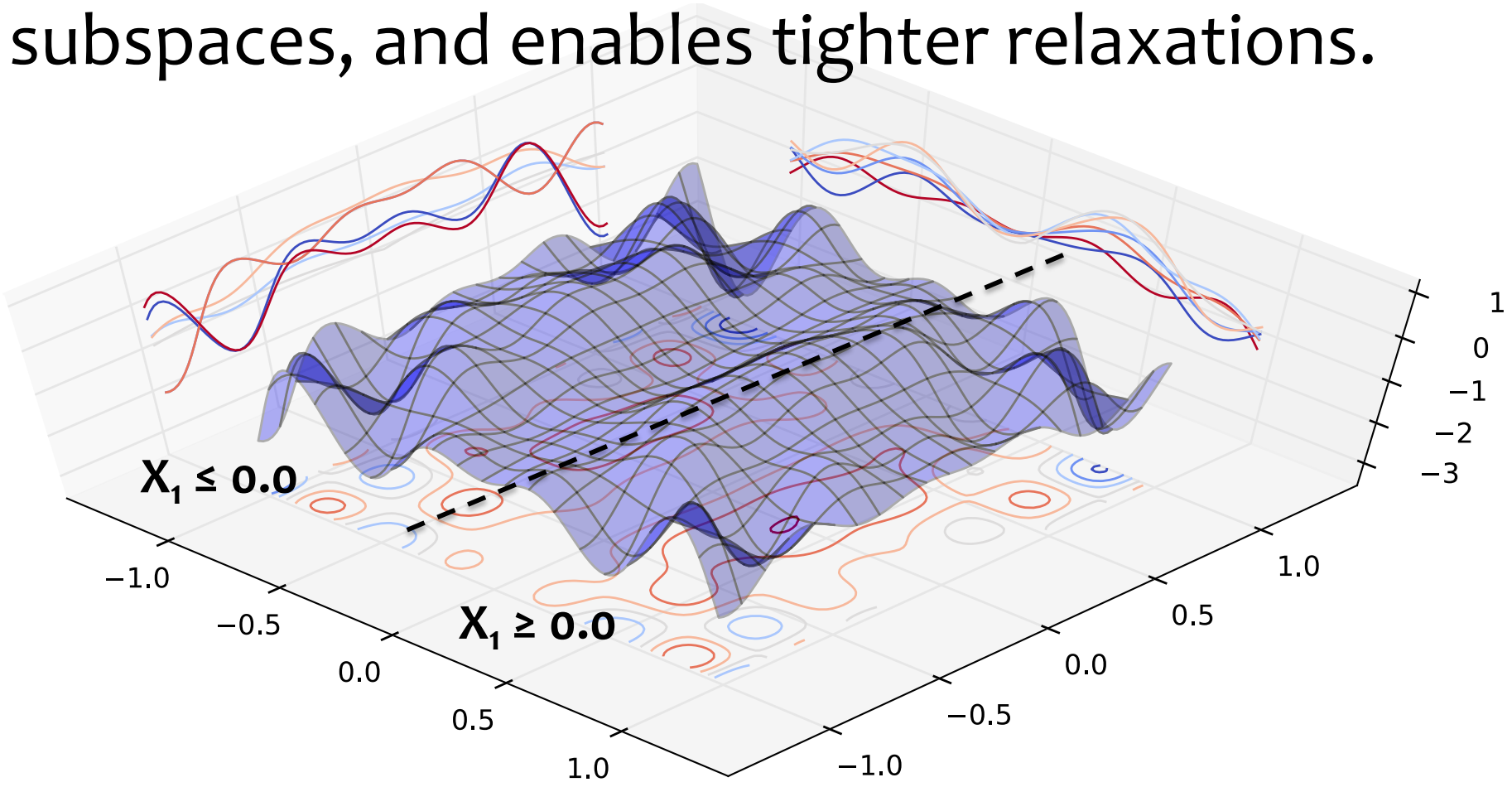
Background: Nonconvex Global Optimization

Relaxation: provides an upper bound on the surface.



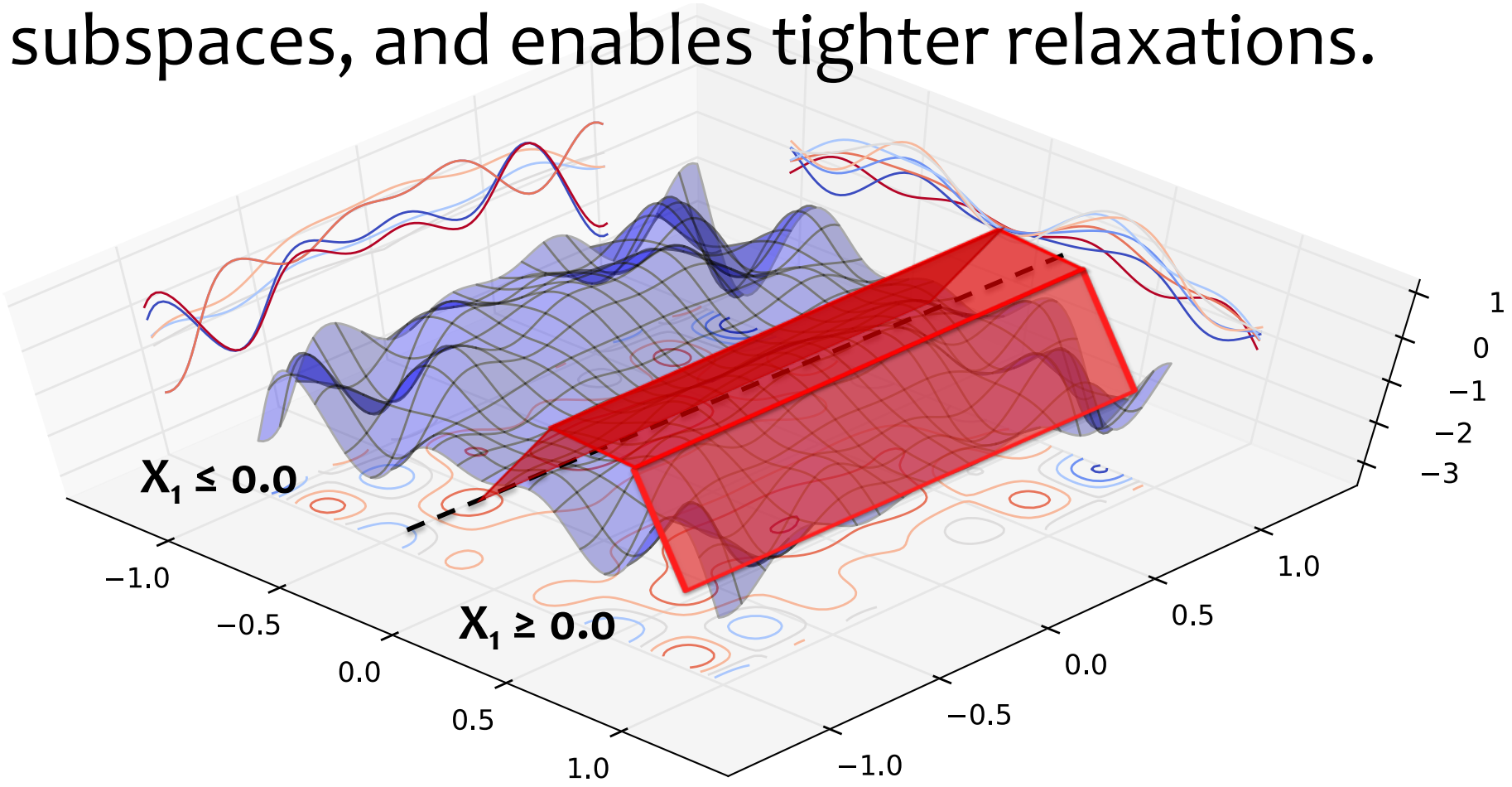
Background: Nonconvex Global Optimization

Branching: partitions the search space into subspaces, and enables tighter relaxations.



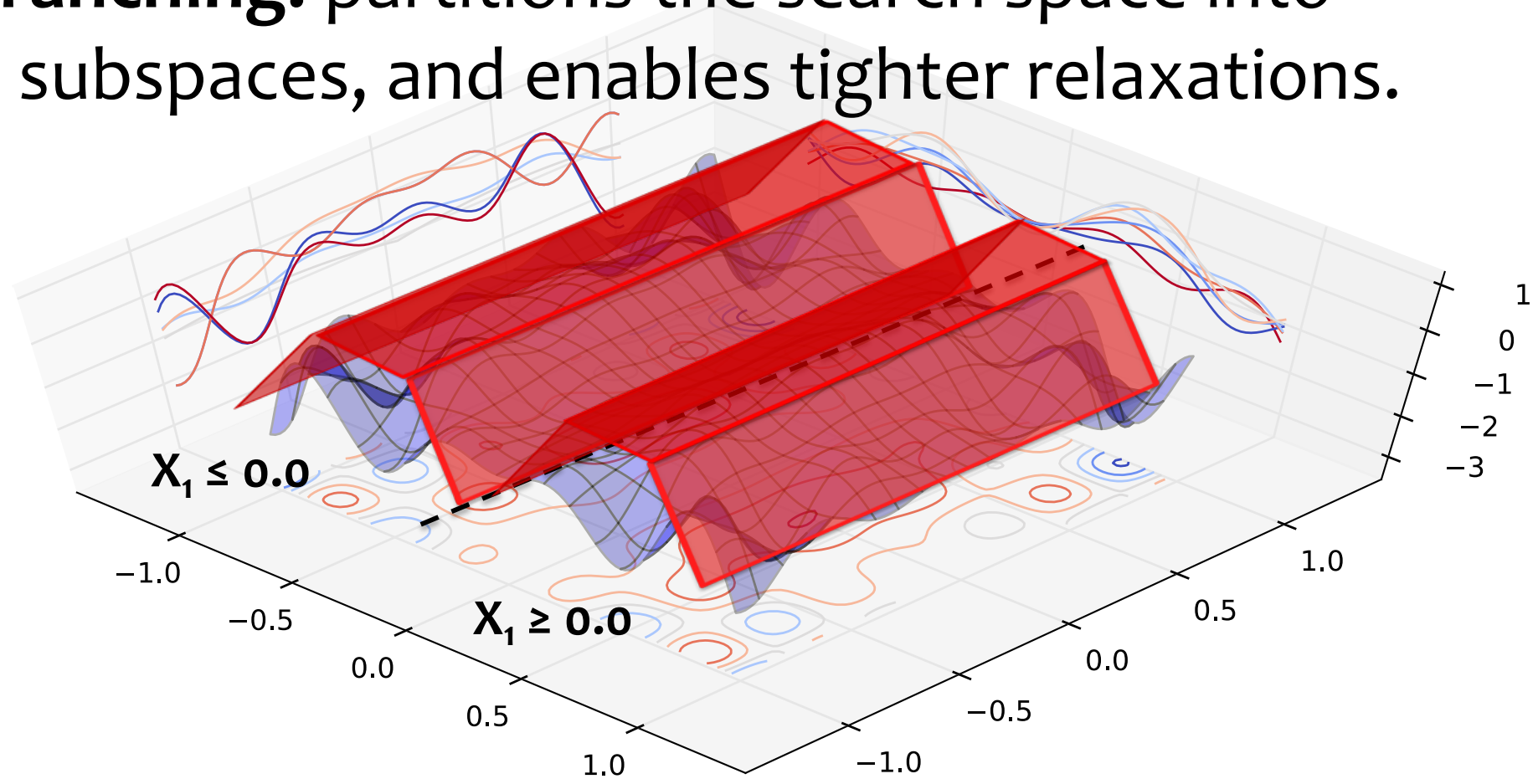
Background: Nonconvex Global Optimization

Branching: partitions the search space into subspaces, and enables tighter relaxations.



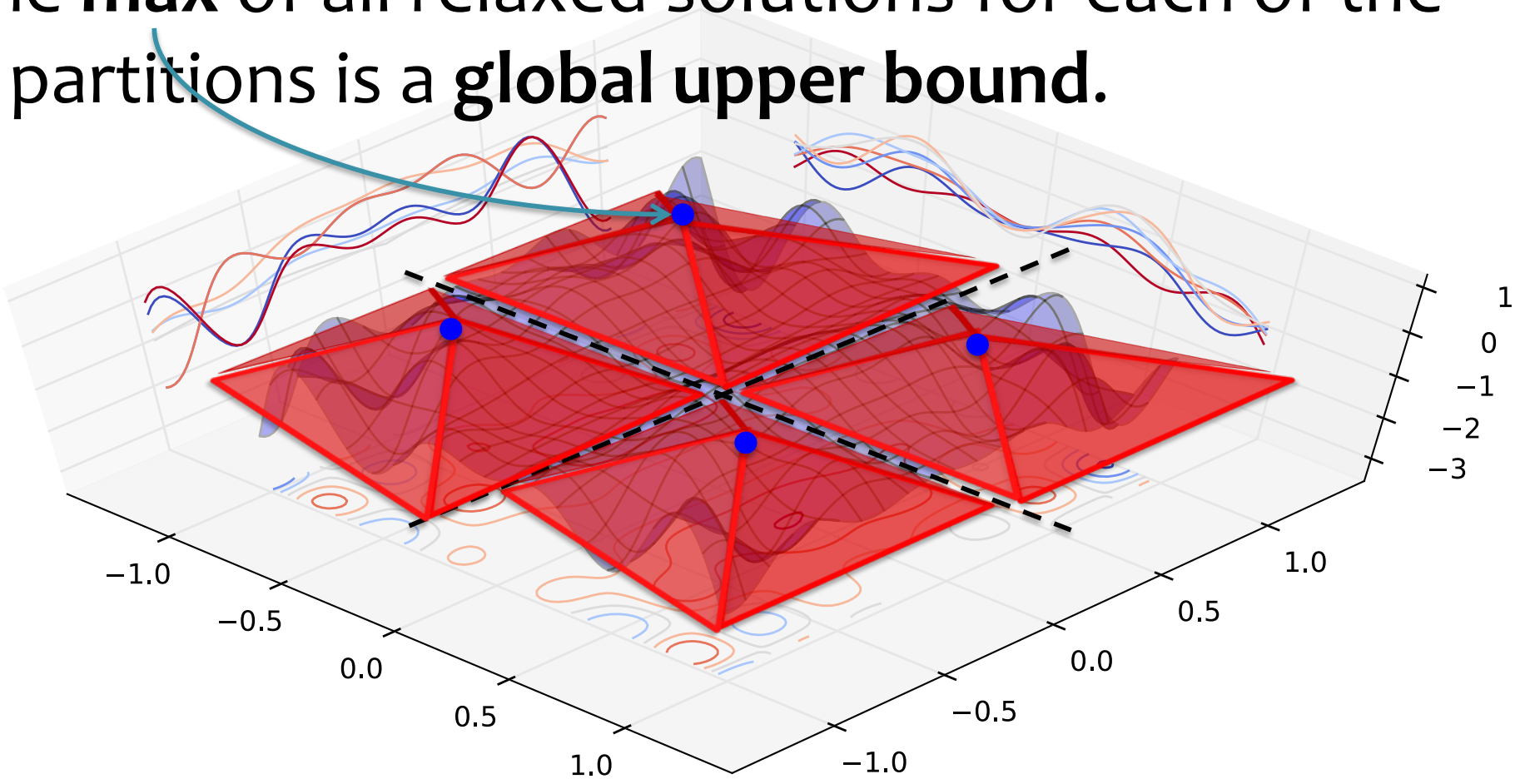
Background: Nonconvex Global Optimization

Branching: partitions the search space into subspaces, and enables tighter relaxations.



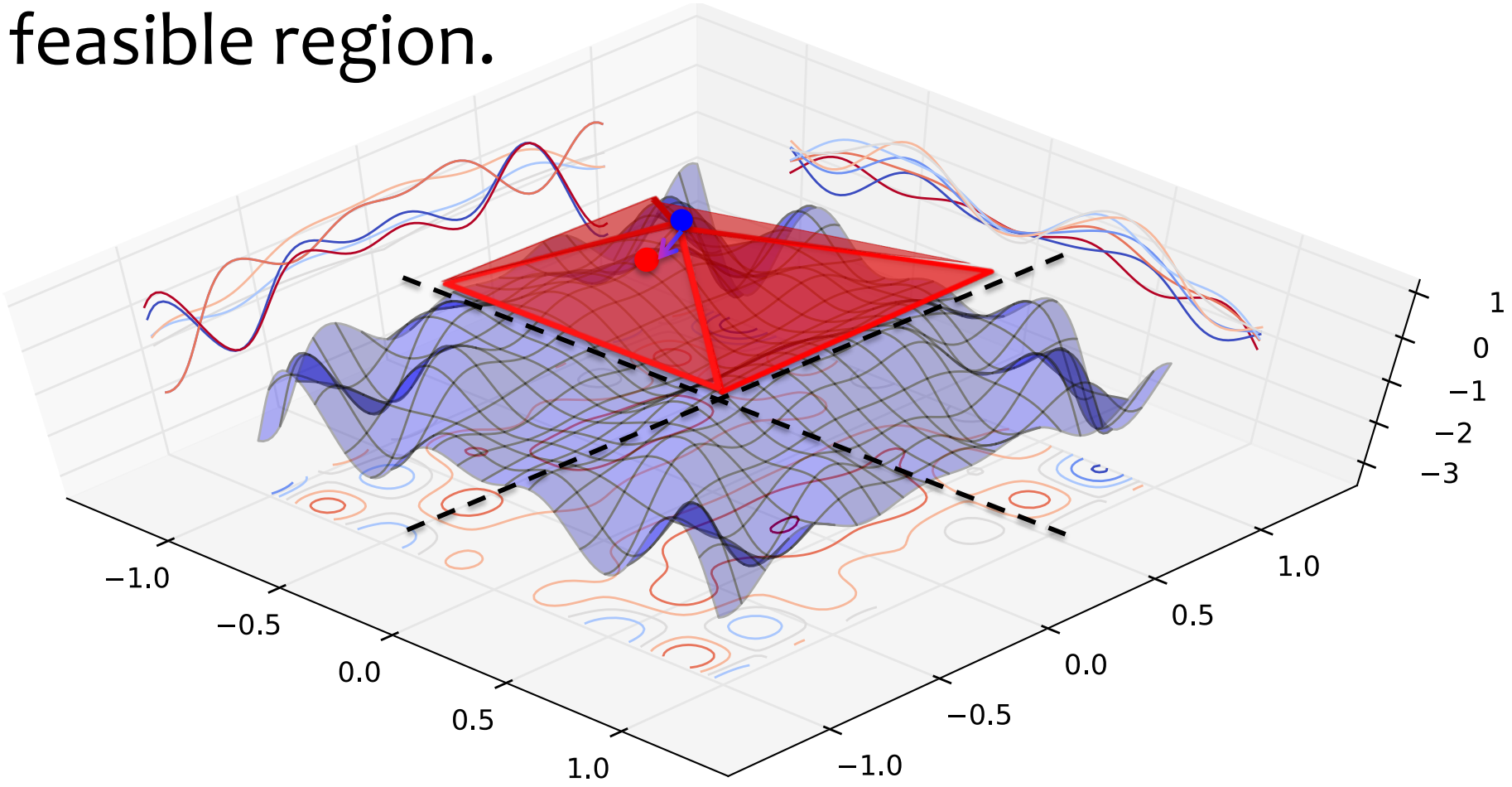
Background: Nonconvex Global Optimization

The **max** of all relaxed solutions for each of the partitions is a **global upper bound.**



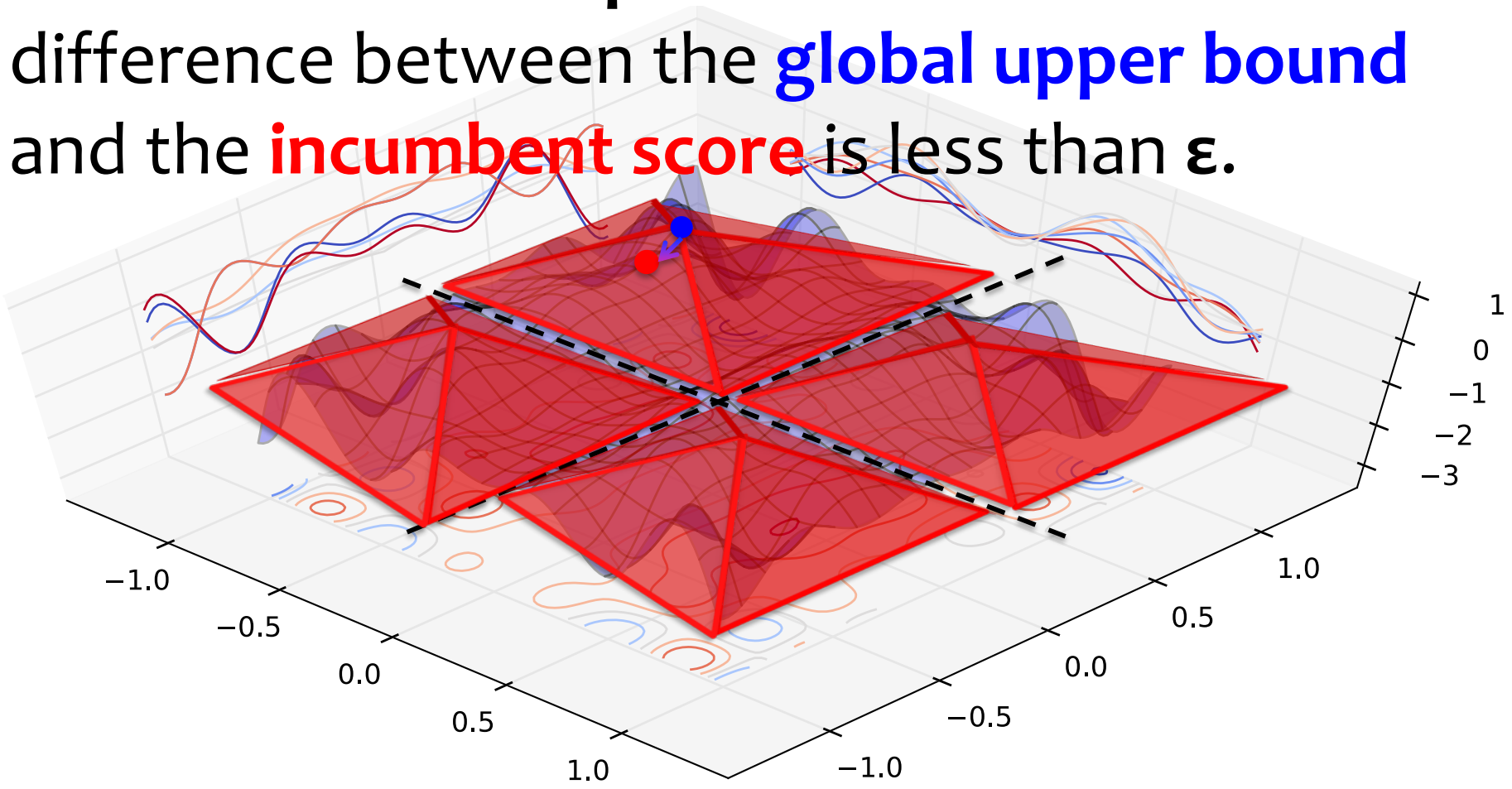
Background: Nonconvex Global Optimization

We can **project** a relaxed solution onto the feasible region.



Background: Nonconvex Global Optimization

The **incumbent** is **ε-optimal** if the relative difference between the **global upper bound** and the **incumbent score** is less than **ε**.



How much should we subdivide?

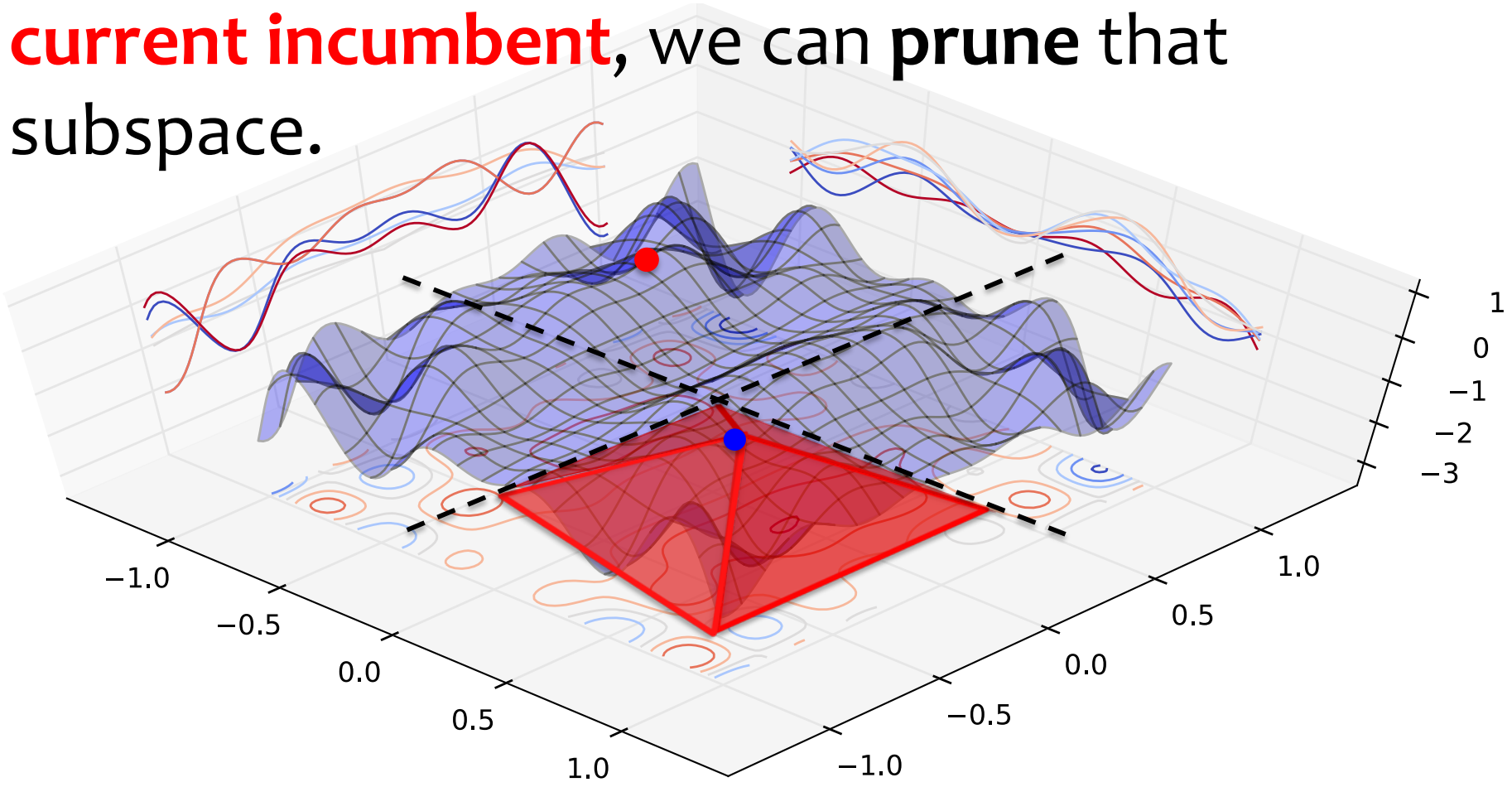
How much should we subdivide?

BRANCH-AND-BOUND

- Method for **recursively subdividing** the search space
- **Subspace order** can be determined heuristically (e.g. best-first search with depth-first plunging)
- **Prunes** subspaces that can't yield better solutions

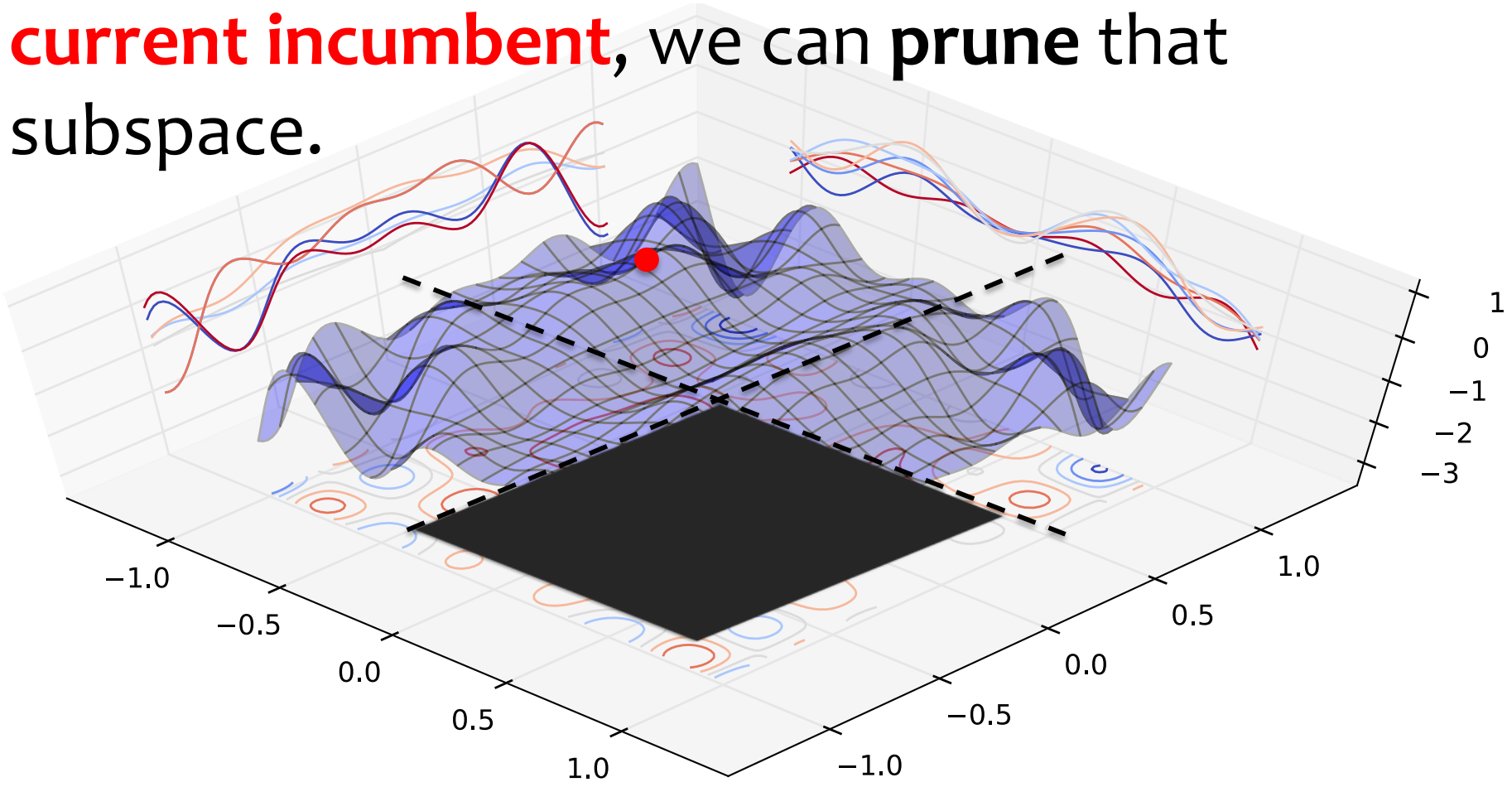
Background: Nonconvex Global Optimization

If the **subspace upper bound** is worse than the **current incumbent**, we can **prune** that subspace.



Background: Nonconvex Global Optimization

If the **subspace upper bound** is worse than the **current incumbent**, we can **prune** that subspace.



Limitations:
Branch-and-Bound

Branch-and-bound

- Kind of tricky to get it right…
  - Lots of hyperparameters to tune
- Curse of dimensionality kicks in quickly
- Applications to problems other than ILP (e.g. QP) rather unsuccessful
  - Nonconvex quadratic optimization by LP-based branch-and-bound usually fails with more than 80 variables (Burer and Vandenbussche, 2009)

BRANCH-AND-BOUND INGREDIENTS

Mathematical Program

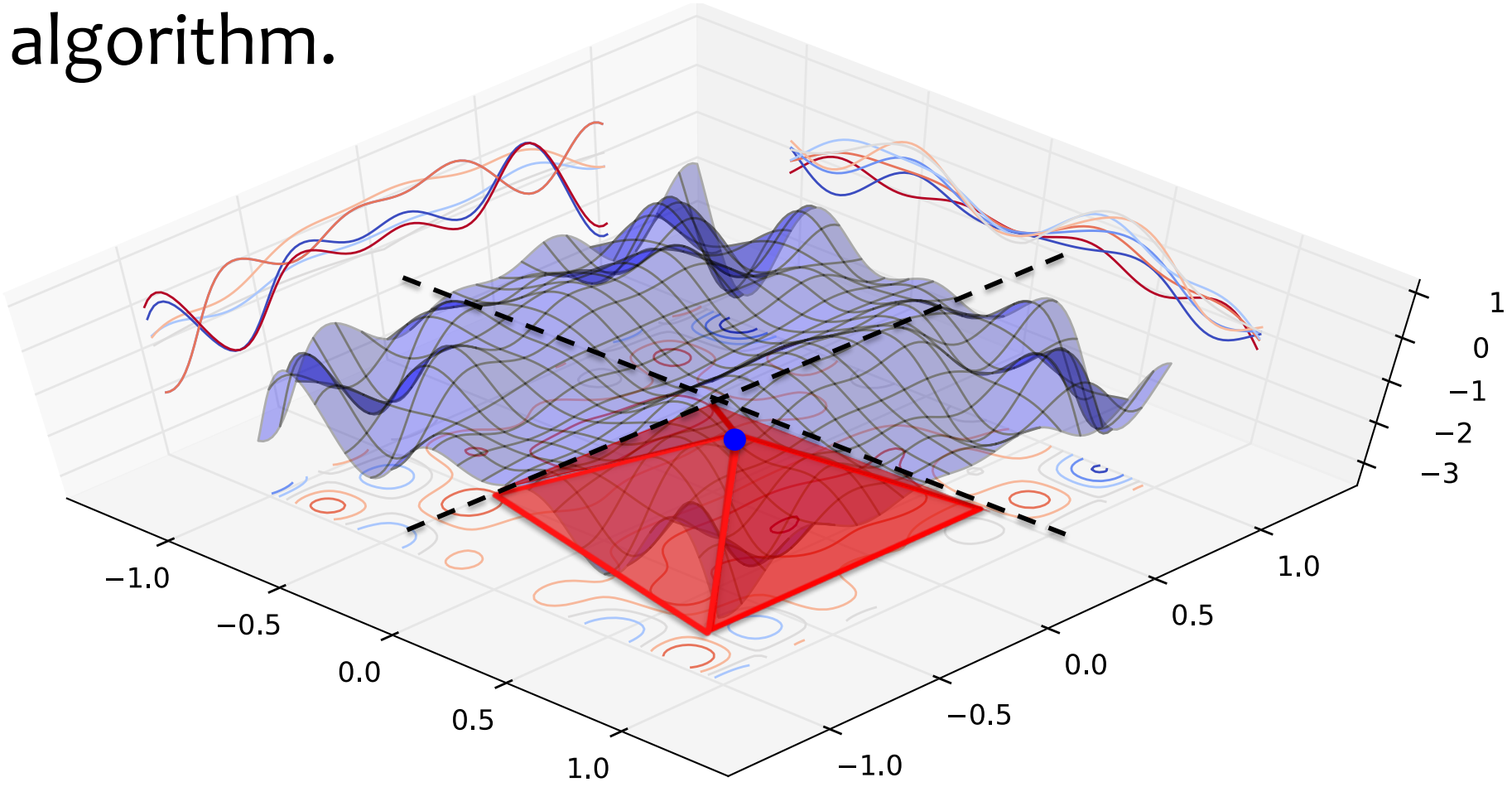
Relaxation

Projection

(Branch-and-Bound Search Heuristics)

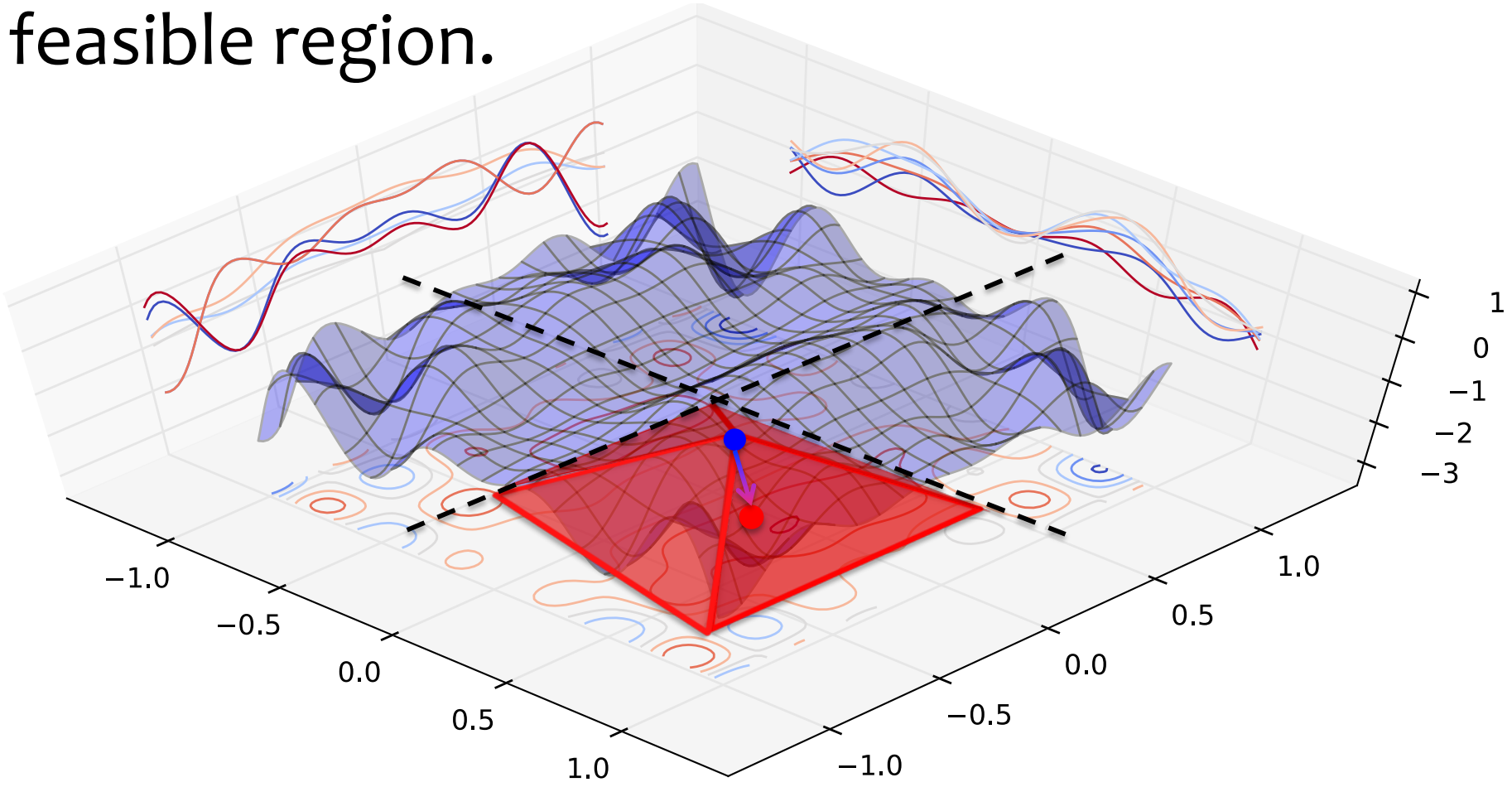
Background: Nonconvex Global Optimization

We solve the **relaxation** using the Simplex algorithm.



Background: Nonconvex Global Optimization

We can **project** a relaxed solution onto the feasible region.



Integer Linear Programming

Whiteboard

– Branch and bound for an ILP in 2D

Branch and Bound

Algorithm 2.1 Branch-and-bound

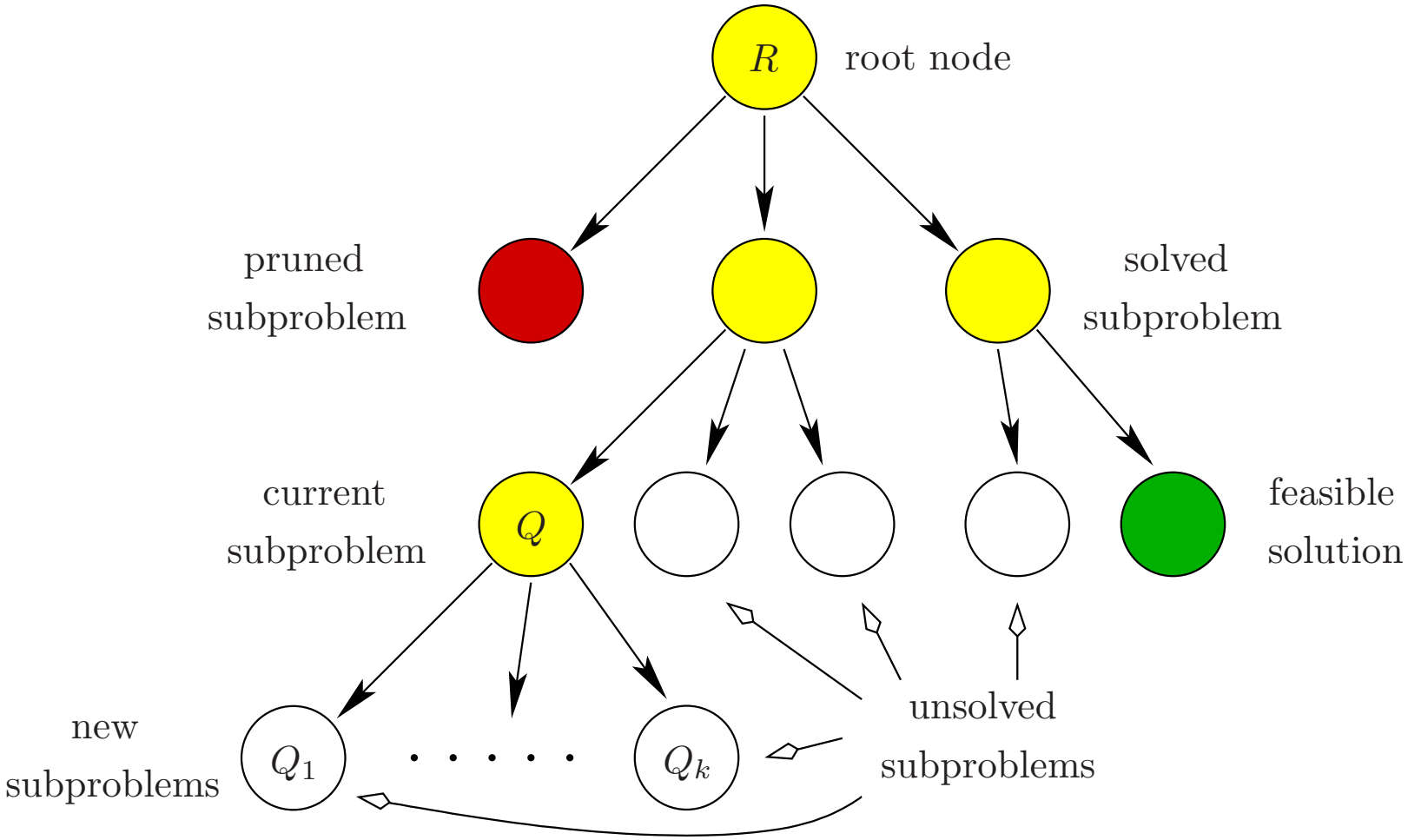
Input: Minimization problem instance R.

Output: Optimal solution $x^\star$ with value $c^\star$, or conclusion that R has no solution, indicated by $c^\star = \infty$.

1. Initialize $\mathcal{L} := \{R\}$, $\hat{c} := \infty$. [*init*]
2. If $\mathcal{L} = \emptyset$, stop and return $x^\star = \hat{x}$ and $c^\star = \hat{c}$. [*abort*]
3. Choose $Q \in \mathcal{L}$, and set $\mathcal{L} := \mathcal{L} \setminus \{Q\}$. [*select*]
4. Solve a relaxation Q_{relax} of Q. If Q_{relax} is empty, set $\check{c} := \infty$. Otherwise, let $\check{x}$ be an optimal solution of Q_{relax} and $\check{c}$ its objective value. [*solve*]
5. If $\check{c} \geq \hat{c}$, goto Step 2. [*bound*]
6. If $\check{x}$ is feasible for R, set $\hat{x} := \check{x}$, $\hat{c} := \check{c}$, and goto Step 2. [*check*]
7. Split Q into subproblems $Q = Q_1 \cup \ldots \cup Q_k$, set $\mathcal{L} := \mathcal{L} \cup \{Q_1, \ldots, Q_k\}$, and goto Step 2. [*branch*]

Slide from Achterberg (thesis, 2007)

Branch and Bound



Slide from Achterberg (thesis, 2007)

Branch and Bound

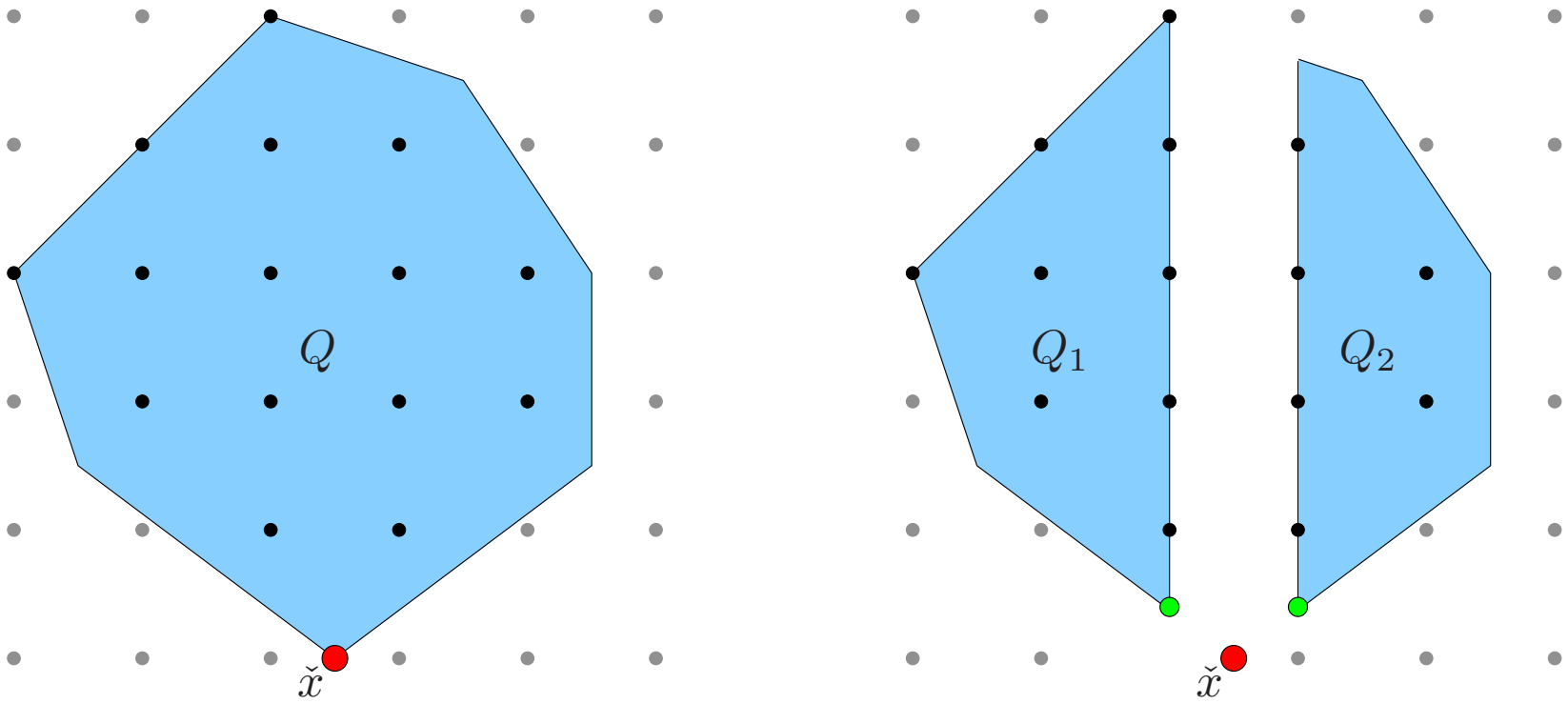


Figure 2.2. LP based branching on a single fractional variable.

Slide from Achterberg (thesis, 2007)

Case #1: Binary Variables

MAP INFERENCE AS MATHEMATICAL PROGRAMMING

Exact Inference

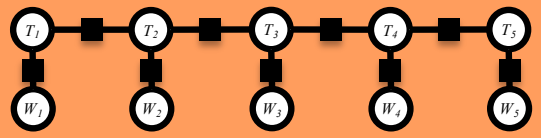
Recall...

1. Data

$$\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$$

2. Model

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

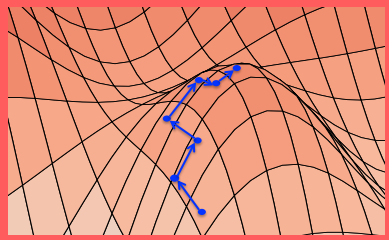


3. Objective

$$\ell(\theta; \mathcal{D}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta})$$

4. Learning

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \ell(\boldsymbol{\theta}; \mathcal{D})$$



5. Inference

1. Marginal Inference

$$p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmax}}\; p(\boldsymbol{x} \mid \boldsymbol{\theta})$$

Recall...

5. Inference

Three Tasks:

1. Marginal Inference

Compute marginals of variables and cliques

$$p(x_i) = \sum_{\boldsymbol{x}' : x'_i = x_i} p(\boldsymbol{x}' \mid \boldsymbol{\theta}) \qquad p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

Compute the normalization constant

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference (NP-Hard in the general case)

Compute variable assignment with highest probability

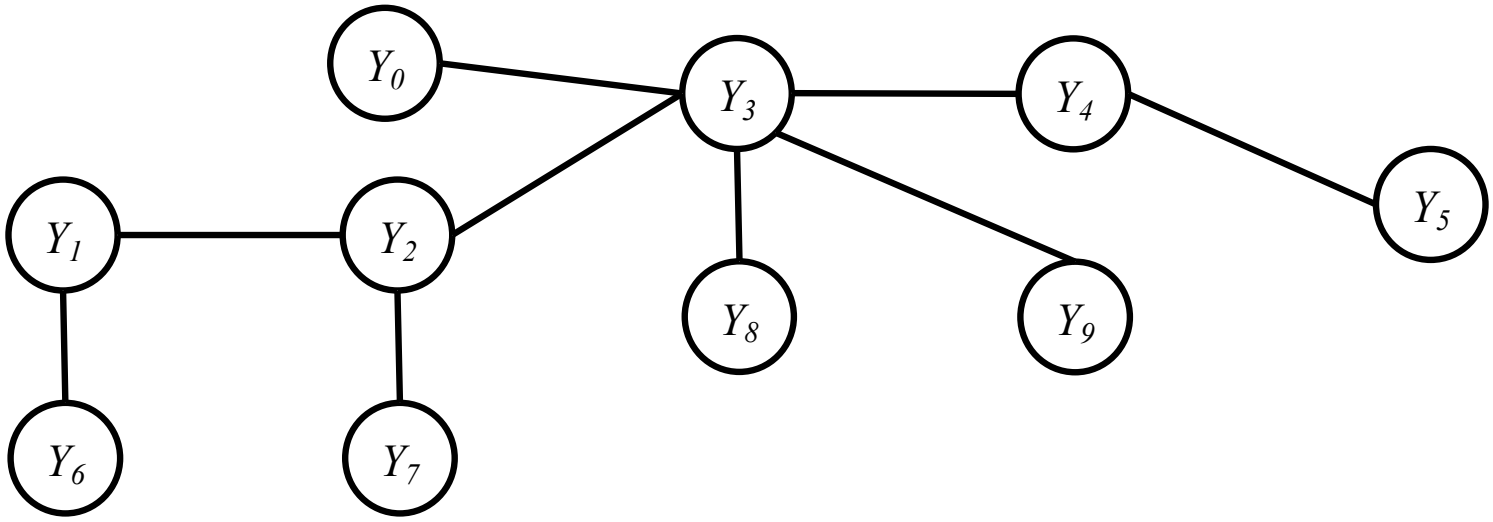
$$\hat{\boldsymbol{x}} = \operatorname*{argmax}_{\boldsymbol{x}} \; p(\boldsymbol{x} \mid \boldsymbol{\theta})$$

MAP Inference

Suppose we want to predict the highest likelihood structure y, given observations x and parameters w.

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \log p_w(y|x)$$

$$= \underset{\mathbf{y}}{\operatorname{argmax}} \sum_j \mathbf{w}^T f_{\text{node}}(x_j, y_j) + \sum_{j,k} \mathbf{w}^T f_{\text{edge}}(\mathbf{x}_{jk}, y_j, y_k)$$



MAP Inference

Suppose we want to predict the highest likelihood structure y, given observations x and parameters w.

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \log p_w(y|x)$$

$$= \underset{\mathbf{y}}{\operatorname{argmax}} \sum_j \mathbf{w}^T f_{\text{node}}(x_j, y_j) + \sum_{j,k} \mathbf{w}^T f_{\text{edge}}(\mathbf{x}_{jk}, y_j, y_k)$$

Idea:

1. Reformulate the problem as an integer linear program (ILP) – note that this is just going to be a new way of writing down the problem: y → z
2. Then remove the integer constraints (i.e. solve the linear program (LP) relaxation)

Lemma: (Wainwright et al., 2002) If there is a unique MAP assignment, the LP relaxation of the ILP above is guaranteed to have an integer solution, which is exactly the MAP solution!

Integer Linear Programming

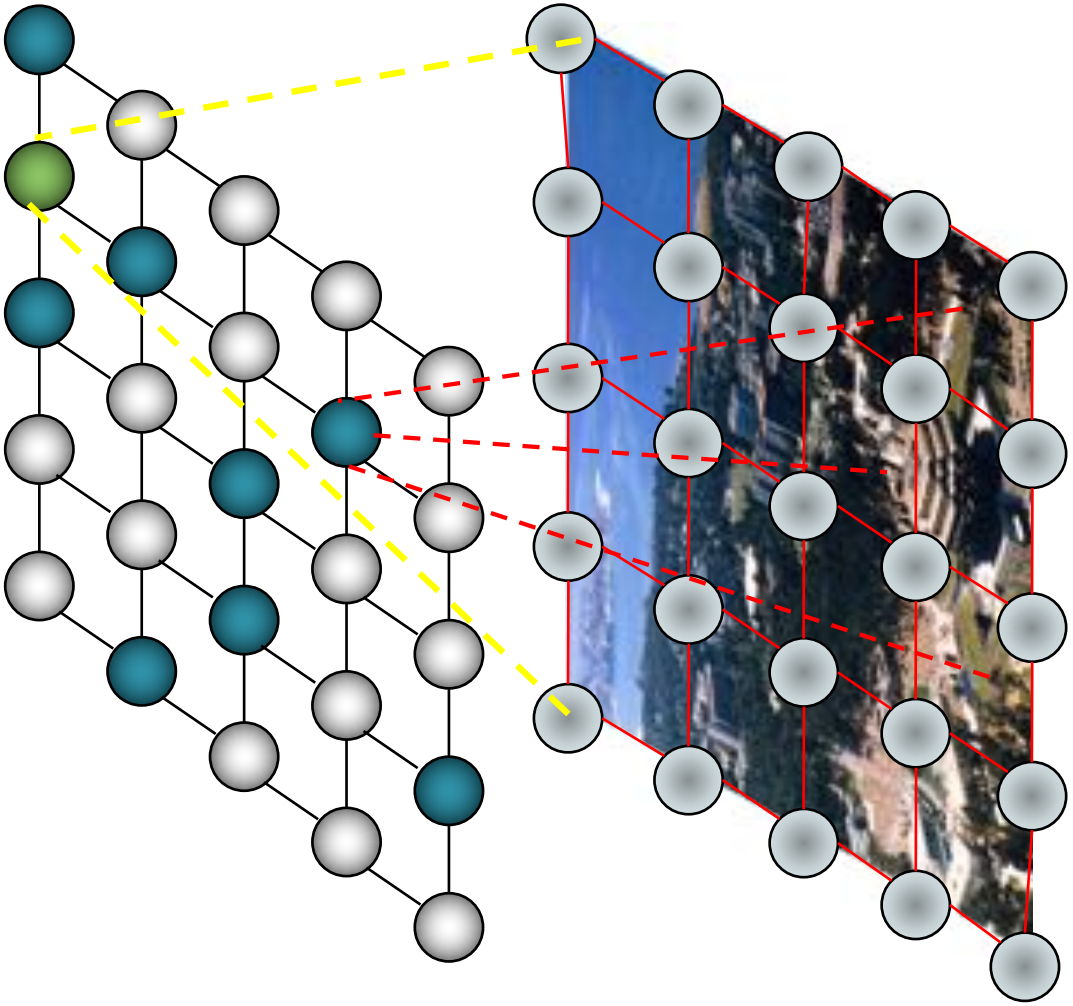
Whiteboard

- MAP Inference for a Binary Pairwise MRF as an ILP

Case #2: Multiclass Variables

MAP INFERENCE AS MATHEMATICAL PROGRAMMING

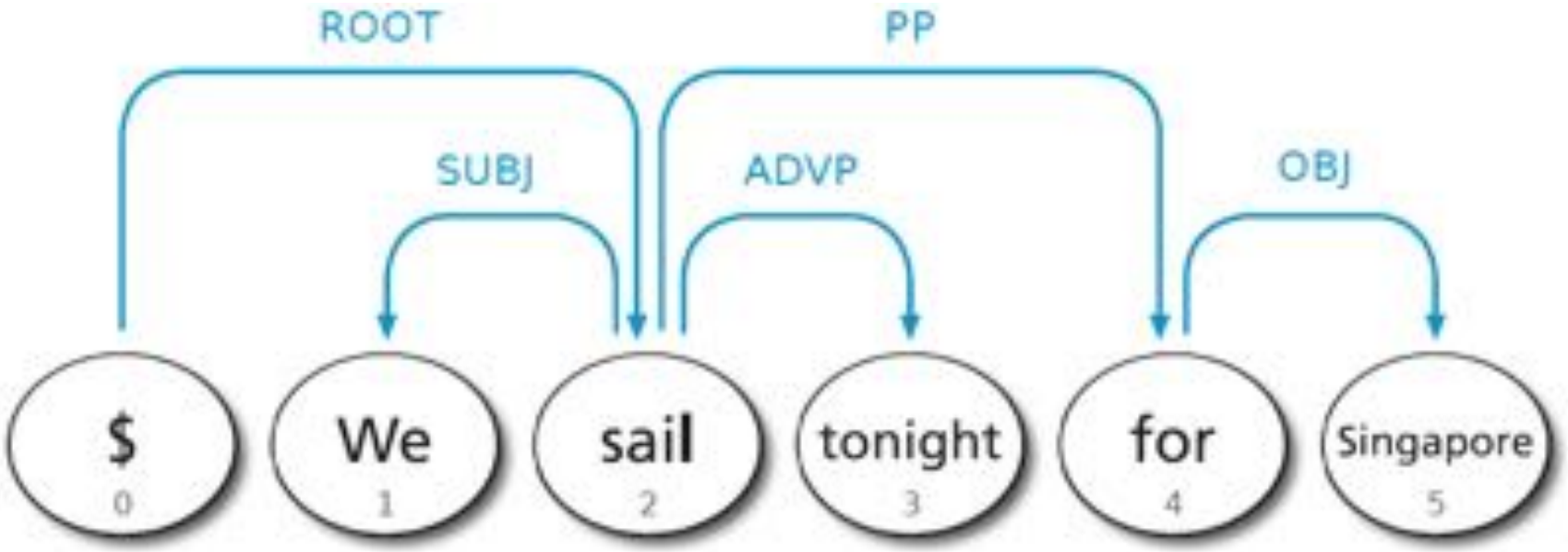
Image Segmentation



$$p_\theta(y \mid x) = \frac{1}{Z(\theta, x)} \exp\left\{\sum_c \theta_c f_c(x, y_c)\right\}$$

- Jointly segmenting/annotating images
- Image-image matching, image-text matching
- Problem:
  - Given structure (feature), learning $\vec{\theta}$
  - Learning sparse, interpretable, **predictive** structures/features

Dependency parsing of Sentences



Challenge:
Structured outputs, and globally constrained to be a valid tree

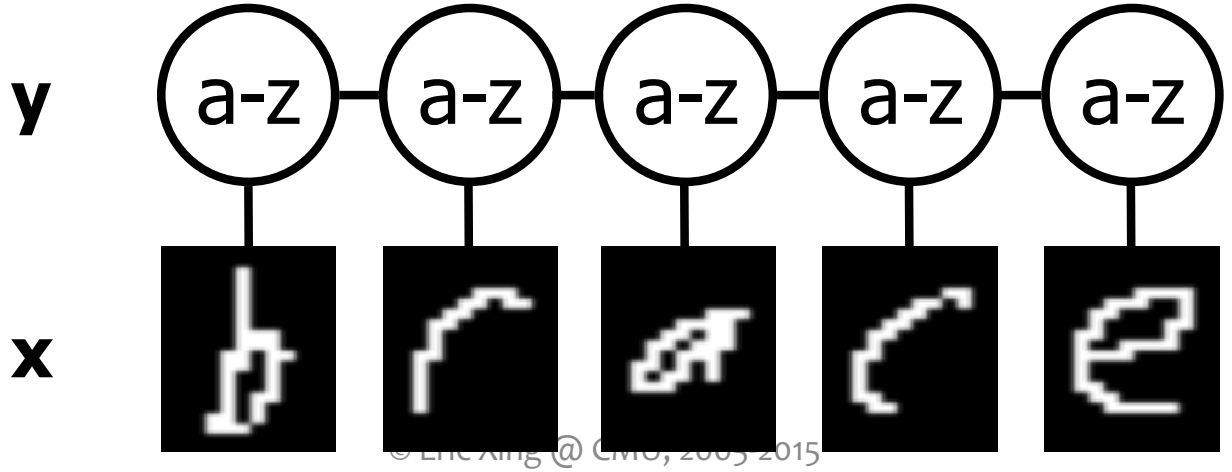
OCR example

x

y

brace

Sequential structure

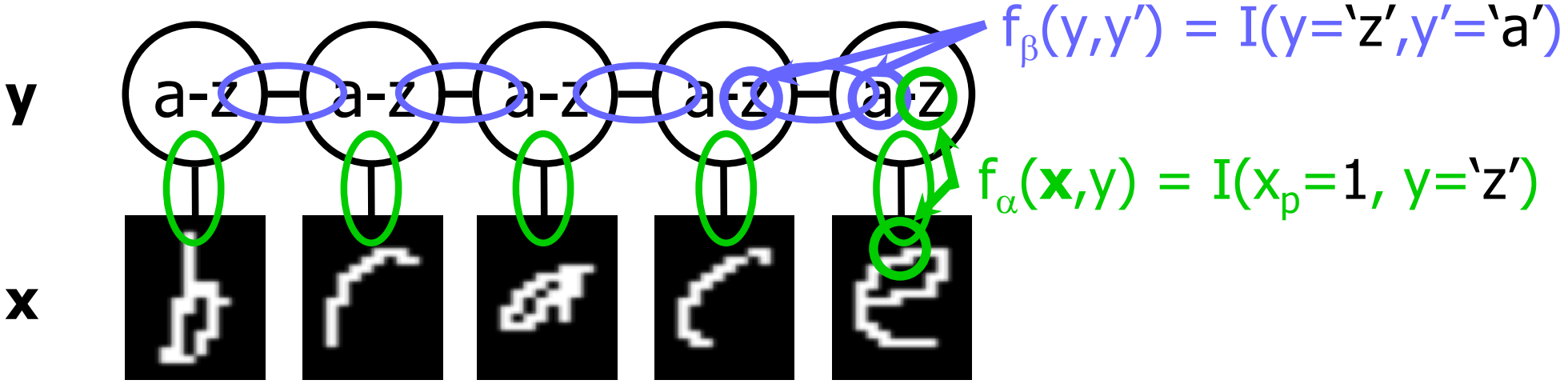


Linear-chain CRF for OCR

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_i \phi(\mathbf{x}_i, y_i) \prod_i \phi(y_i, y_{i+1})$$

$$\phi(\mathbf{x}_i, y_i) = \exp\{\sum_\alpha w_\alpha f_\alpha(\mathbf{x}_i, y_i)\}$$

$$\phi(y_i, y_{i+1}) = \exp\{\sum_\beta w_\beta f_\beta(y_i, y_{i+1})\}$$

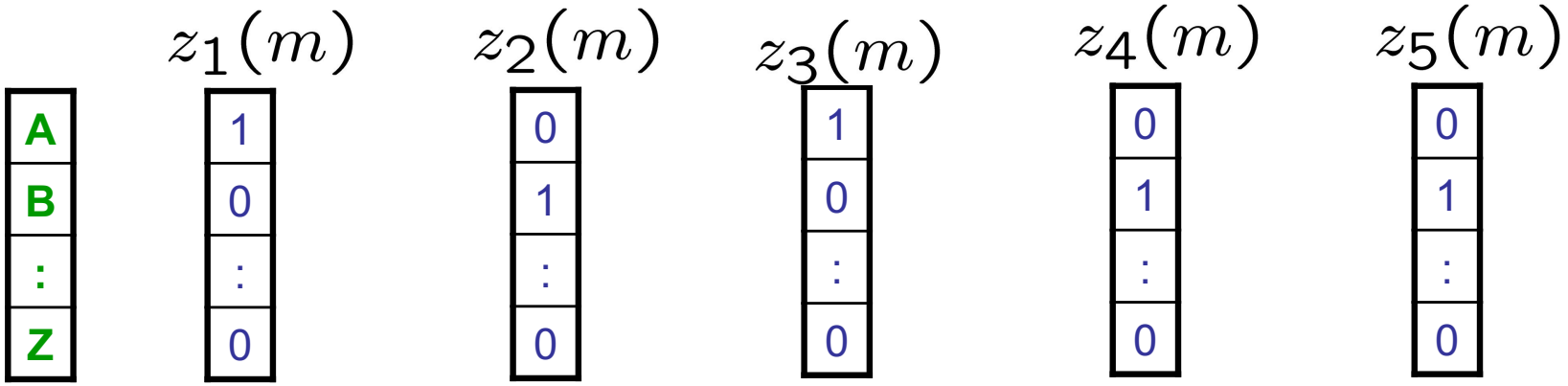


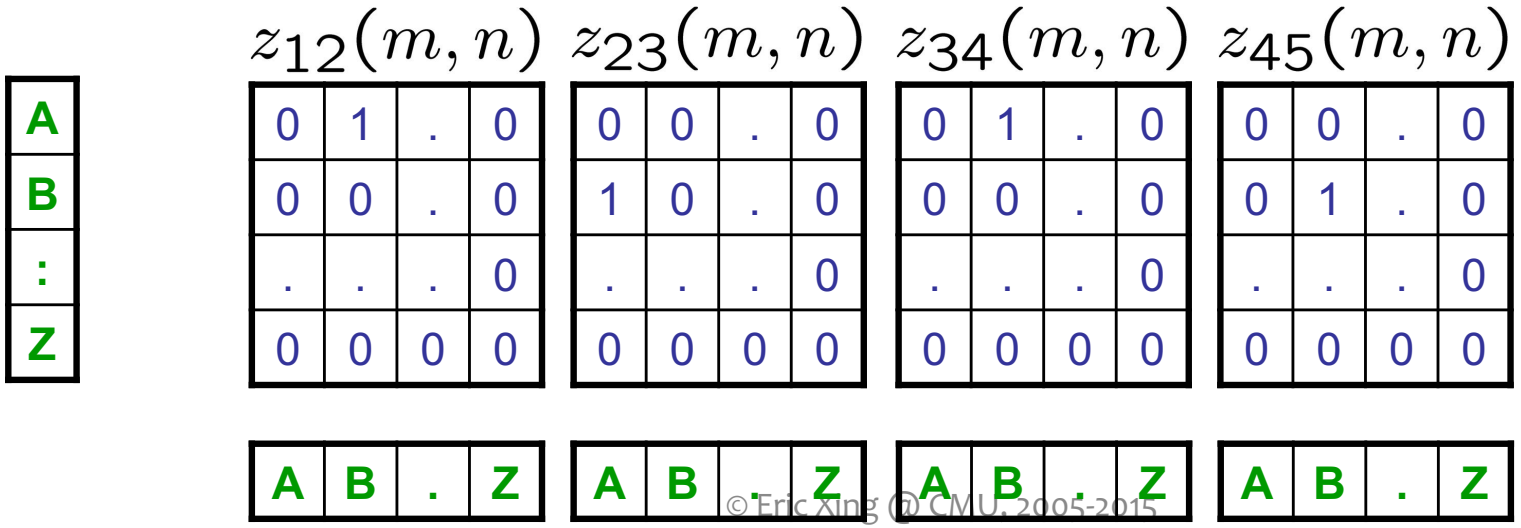
*Lafferty et al. 01

$y \Rightarrow z$ map for linear chain structures

OCR example: $\mathbf{y} =$ 'ABABB';

$\mathbf{z}$'s are the indicator variables for the corresponding classes (alphabet)





y ⇒ z map for linear chain structures

$$\max_{\mathbf{y}} \sum_j \mathbf{w}^T f_{\text{node}}(x_j, y_j) + \sum_{j,k} \mathbf{w}^T f_{\text{edge}}(\mathbf{x}_{jk}, y_j, y_k)$$

Rewriting the maximization function in terms of indicator variables:

$$\max_{\mathbf{z}} \sum_{j,m} z_j(m) \left[\mathbf{w}^\top \mathbf{f}_{\text{node}}(\mathbf{x}_j, m)\right] + \sum_{jk,m,n} z_{jk}(m,n) \left[\mathbf{w}^\top \mathbf{f}_{\text{edge}}(\mathbf{x}_{jk}, m, n)\right]$$

$z_k(n)$

0	1	0	0

$z_j(m)$

0
0
1
0

$z_{jk}(m,n)$

0	0	0	0
0	0	0	0
0	1	0	0
0	0	0	0

$$z_j(m) \geq 0; \; z_{jk}(m,n) \geq 0;$$

normalization $\sum_m z_j(m) = 1$

agreement $\sum_n z_{jk}(m,n) = z_j(m)$

integer $z_j(m) \in \mathcal{Z}, \; z_{jk}(m,n) \in \mathcal{Z}$

y ⇒ z map for linear chain structures

$$\max_{\mathbf{y}} \sum_j \mathbf{w}^T f_{\text{node}}(x_j, y_j) + \sum_{j,k} \mathbf{w}^T f_{\text{edge}}(\mathbf{x}_{jk}, y_j, y_k)$$

Rewriting the maximization function in terms of indicator variables:

$$\max_{\mathbf{z}} \sum_{j,m} z_j(m) \left[\mathbf{w}^\top \mathbf{f}_{\text{node}}(\mathbf{x}_j, m)\right] + \sum_{jk,m,n} z_{jk}(m,n) \left[\mathbf{w}^\top \mathbf{f}_{\text{edge}}(\mathbf{x}_{jk}, m, n)\right] \quad \Big\} (\mathbf{F}^\top \mathbf{w} \quad)^\top \mathbf{z}$$

$z_k(n)$

$z_j(m)$

0	1	0	0

0
0
1
0

0	0	0	0
0	0	0	0
0	1	0	0
0	0	0	0

$z_{jk}(m,n)$

$$z_j(m) \geq 0; \; z_{jk}(m,n) \geq 0;$$

normalization $\sum_m z_j(m) = 1$

agreement $\sum_n z_{jk}(m,n) = z_j(m)$

$\Big\} \mathbf{Az} = \mathbf{b}$

$$\max_{A\mathbf{z}=\mathbf{b}} (\mathbf{F}^\top \mathbf{w} \quad)^\top \mathbf{z}$$

MAP Inference

Suppose we want to predict the highest likelihood structure y, given observations x and parameters w.

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \log p_w(y|x)$$

$$= \underset{\mathbf{y}}{\operatorname{argmax}} \sum_j \mathbf{w}^T f_{\text{node}}(x_j, y_j) + \sum_{j,k} \mathbf{w}^T f_{\text{edge}}(\mathbf{x}_{jk}, y_j, y_k)$$

Idea:

1. Reformulate the problem as an integer linear program (ILP) – note that this is just going to be a new way of writing down the problem: y → z
2. Then remove the integer constraints (i.e. solve the linear program (LP) relaxation)

Lemma: (Wainwright et al., 2002) If there is a unique MAP assignment, the LP relaxation of the ILP above is guaranteed to have an integer solution, which is exactly the MAP solution!

STRUCTURED PERCEPTRON

Linear Models

Setting: training examples are (x, y) where $\vec{x} \in \mathbb{R}^p$ $y \in \{1, \ldots, k\}$

Model: parameters $\theta \in \mathbb{R}^M$
feature function $f(x, y) \in \mathbb{R}^M$

Predict: $\hat{y} = h_\theta(x) = \underset{y \in \{1,\ldots,k\}}{\operatorname{argmax}} \theta^T f(\vec{x}, y)$

Ex#1: $f(\vec{x}, y) = \text{vectorize}\left(\begin{bmatrix} 0 & 0 & \cdots & 0 \\ \vdots & & & \\ x_1 & x_2 & \cdots & x_p \\ \vdots & & & \\ 0 & 0 & \cdots & 0 \end{bmatrix} \right)$ ← $k \times p$ matrix

rows $1, 2, \ldots, y, \ldots, k-1, k$

only the y^{th} row is nonzero

$= [0 0 \cdots 0 \; x_1 \; x_2 \cdots x_p \; 0 0 \cdots 0]^T$

$\Rightarrow M = k \times p$

Linear Models

Setting: training examples are (x, y) where $\vec{x} \in \mathbb{R}^p$ $y \in \{1, \ldots, K\}$

Model: parameters $\theta \in \mathbb{R}^M$
feature function $f(x, y) \in \mathbb{R}^M$

Predict: $\hat{y} = h_\theta(x) = \underset{y \in \{1,\ldots,K\}}{\operatorname{argmax}} \theta^T f(\vec{x}, y)$

Ex 4c: Suppose $\{1, \ldots, K\}$ exist in some hierarchy and $\vec{x} \in \{0,1\}^p$

$$f(\vec{x}, y) = \begin{bmatrix} \mathbb{1}(x_1 = 1 \text{ and } y \in \text{mammal}) \\ \mathbb{1}(x_1 = 1 \text{ and } y = \text{cat}) \\ \mathbb{1}(x_1 = 1 \text{ and } y = \text{dog}) \\ \vdots \\ \mathbb{1}(x_p = 1 \text{ and } y \in \text{amph.}) \\ \vdots \end{bmatrix}$$

Structured Perceptron

Whiteboard

- Multiclass Perceptron
- Structured Perceptron

Structured Perceptron

Mistake Bound:

Definition 1 *Let* $\overline{\mathbf{GEN}}(x_i) = \mathbf{GEN}(x_i) - \{y_i\}$. *In other words* $\overline{\mathbf{GEN}}(x_i)$ *is the set of* incorrect *candidates for an example* x_i. *We will say that a training sequence* (x_i, y_i) *for* $i = 1 \ldots n$ *is* **separable with margin** $\delta > 0$ *if there exists some vector* $\mathbf{U}$ *with* $||\mathbf{U}|| = 1$ *such that*

$$\forall i, \forall z \in \overline{\mathbf{GEN}}(x_i), \quad \mathbf{U} \cdot \Phi(x_i, y_i) - \mathbf{U} \cdot \Phi(x_i, z) \geq \delta \quad (3)$$

($||\mathbf{U}||$ *is the 2-norm of* $\mathbf{U}$, *i.e.,* $||\mathbf{U}|| = \sqrt{\sum_s \mathbf{U}_s^2}$.*)*

Theorem 1 *For any training sequence* (x_i, y_i) *which is separable with margin* δ, *then for the perceptron algorithm in figure 2*

$$\textit{Number of mistakes} \leq \frac{R^2}{\delta^2}$$

where R *is a constant such that* $\forall i, \forall z \in \overline{\mathbf{GEN}}(x_i)$ $||\Phi(x_i, y_i) - \Phi(x_i, z)|| \leq R$.

from Collins (2002)

Structured Perceptron

- Results from Collins (2002) on two **sequence tagging** problems
- Metrics:
  - **F-measure:** higher is better
  - **Error:** lower is better
- Comparison of…
  - Structured Perceptron **with** and **without** averaging
  - Maximum entropy Markov model (**ME**MM)
- Takeaways:
  - incredibly **easy to implement**
  - typically **blazing fast**

NP Chunking Results

Method	F-Measure	Numits
Perc, avg, cc=0	93.53	13
Perc, noavg, cc=0	93.04	35
Perc, avg, cc=5	93.33	9
Perc, noavg, cc=5	91.88	39
ME, cc=0	92.34	900
ME, cc=5	92.65	200

POS Tagging Results

Method	Error rate/%	Numits
Perc, avg, cc=0	2.93	10
Perc, noavg, cc=0	3.68	20
Perc, avg, cc=5	3.03	6
Perc, noavg, cc=5	4.04	17
ME, cc=0	3.4	100
ME, cc=5	3.28	200

Figure 4: Results for various methods on the part-of-speech tagging and chunking tasks on development data. All scores are error percentages. Numits is the number of training iterations at which the best score is achieved. Perc is the perceptron algorithm, ME is the maximum entropy method. Avg/noavg is the perceptron with or without averaged parameter vectors. cc=5 means only features occurring 5 times or more in training are included, cc=0 means all features in training are included.

Table from Collins (2002)

aka. Max-Margin Markov Networks (M^3Ns)

STRUCTURED SVM

Structured Perceptron

Whiteboard

- Warmup: Binary SVM
- Warmup: Binary SVM Hinge Loss
- Structured Large Margin
- Structured Hinge Loss
- Gradient of Structured Hinge Loss
- SGD for Structured SVM
- Loss Augmented MAP Inference

Max vs "Soft-Max" Margin

- SVMs:

$$\min_{\mathbf{w}} k||\mathbf{w}||^2 - \sum_i \left(\underbrace{\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(y) \right)}_{\text{Hard (Penalized) Margin}} \right)$$

- Maxent:

$$\min_{\mathbf{w}} \; k||w||^2 - \sum_i \left(\underbrace{\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \right)}_{\text{Soft Margin}} \right)$$

- Very similar! Both try to make the true score better than a function of the other scores.
  - The SVM tries to beat the augmented runner-up
  - The maxent classifier tries to beat the "soft-max"

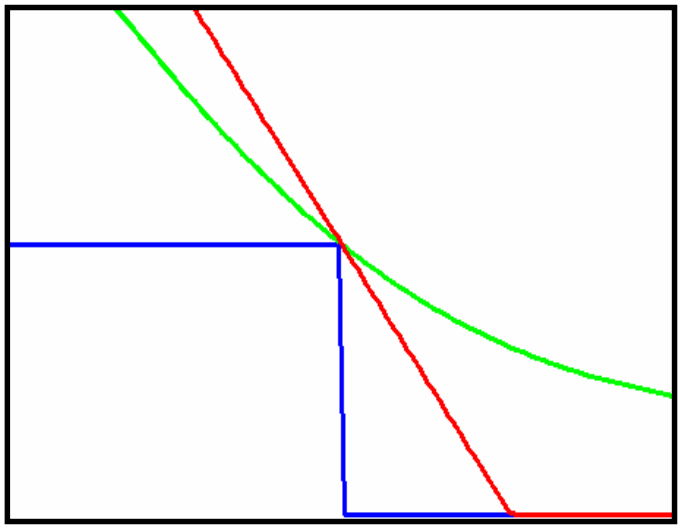
Slide from Klein & Taskar (ACL 2005 Tutorial)

Hinge Loss

- Consider the per-instance SVM objective:

$$\min_{\mathbf{w}} k||\mathbf{w}||^2 - \sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y}} \left[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(y) \right] \right)$$

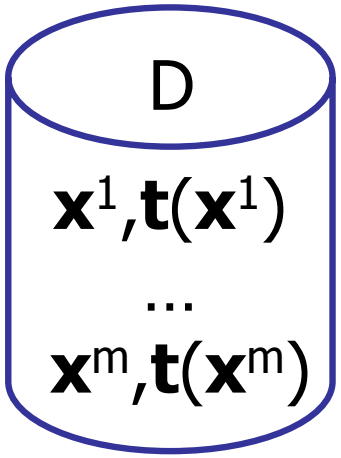
- This is called the "hinge loss"
  - Upper bounds zero-one loss
  - Unlike maxent / log loss, you stop gaining objective once the true label wins by enough
  - You can start from here and derive the SVM objective



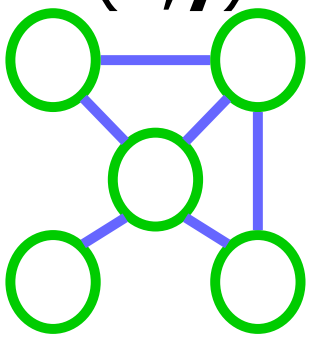
$$\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y} \neq \mathbf{y}^i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$$

Slide from Klein & Taskar (ACL 2005 Tutorial)

Max (Conditional) Likelihood



f(x,y)



Estimation

$$\text{maximize}_{\mathbf{w}} \sum_{\mathbf{x} \in D} \log P_{\mathbf{w}}(\mathbf{t}(\mathbf{x}) \mid \mathbf{x})$$

Classification

$$\arg\max_{\mathbf{y}} \mathbf{w}^{\top} \mathbf{f}(\mathbf{x}, \mathbf{y})$$

$$\log P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x}) = \mathbf{w}^{\top} \mathbf{f}(\mathbf{x}, \mathbf{y}) - \log Z_{\mathbf{w}}(\mathbf{x})$$

Don't need to learn entire distribution!

Structured SVM

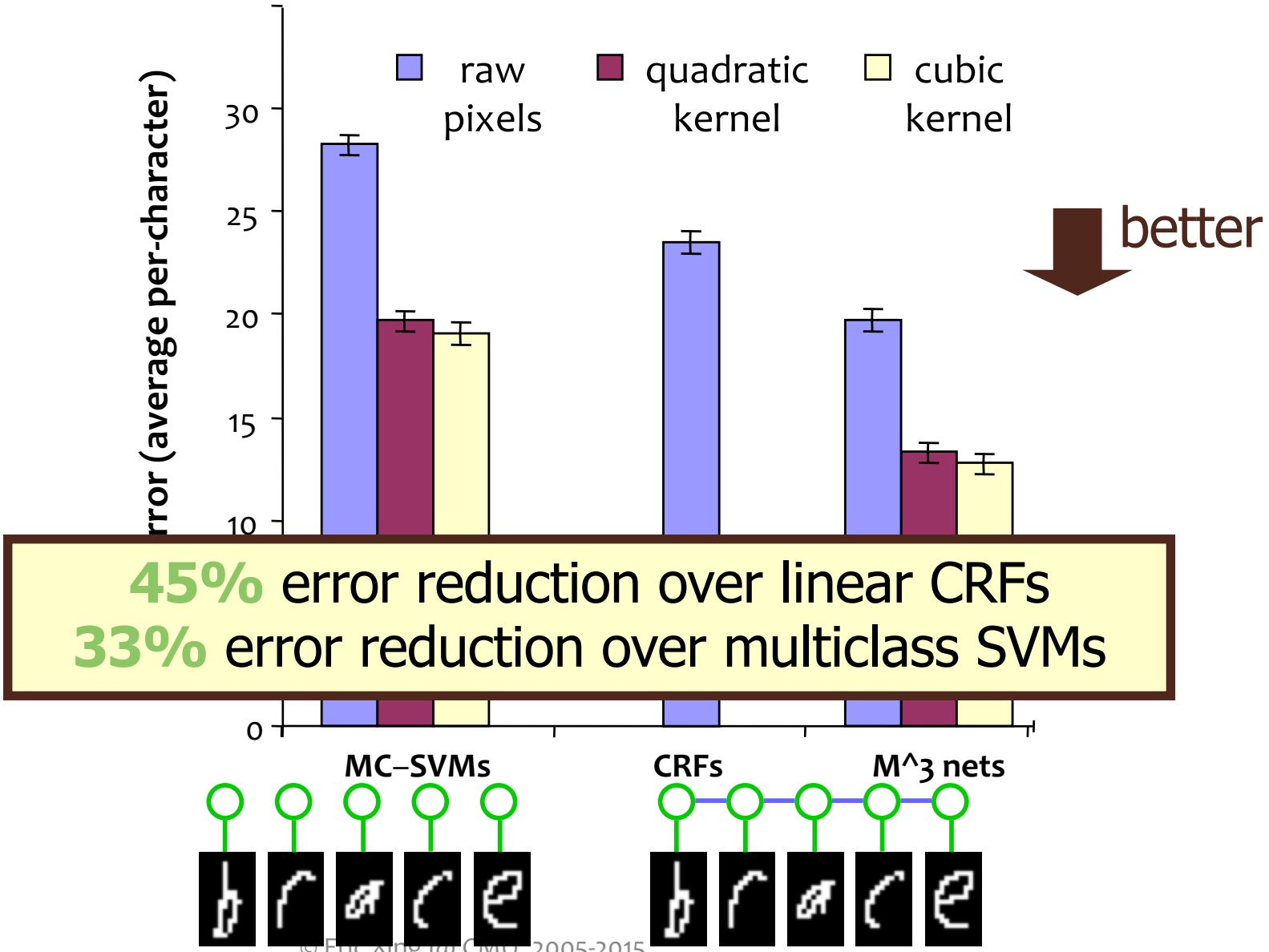
The original name for **Structured SVM:**

- **Max-Margin Markov Networks**
- **abbreviated as** M^3Ns

Results: Handwriting Recognition

Length: ~8 chars
Letter: 16x8 pixels
10-fold Train/Test
5000/50000 letters
600/6000 words

Models:
- Multiclass-SVMs*
- CRFs
- M³ nets

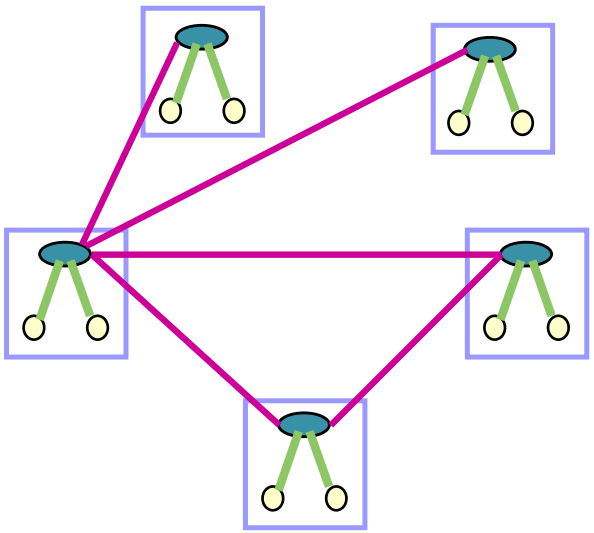


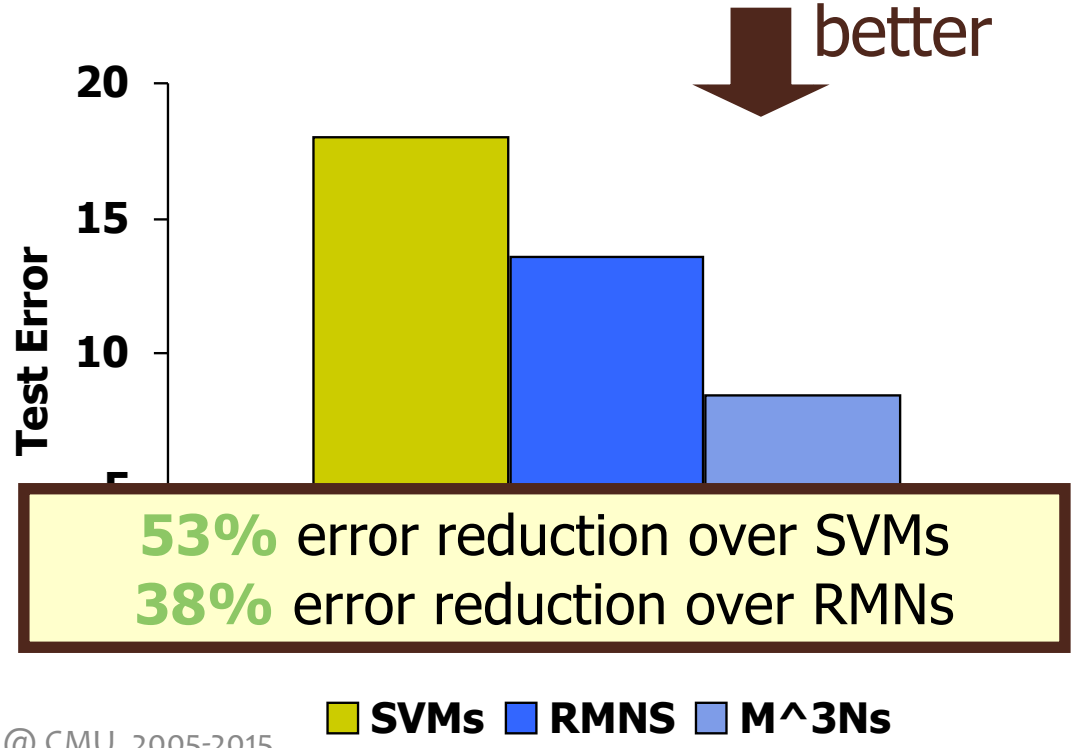
45% error reduction over linear CRFs
33% error reduction over multiclass SVMs

*Crammer & Singer 01

Results: Hypertext Classification

- WebKB dataset
  - Four CS department websites: 1300 pages/3500 links
  - Classify each page: faculty, course, student, project, other
  - Train on three universities/test on fourth
- Inference: loopy belief propagation
- Learning: relaxed dual

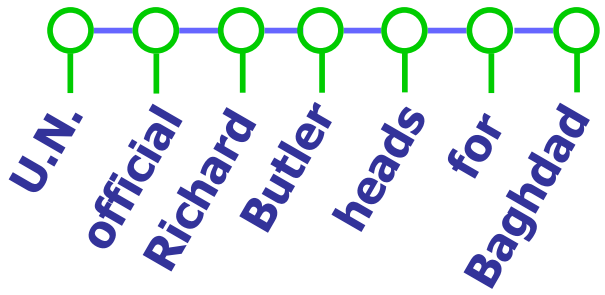




*Taskar et al 02

Named Entity Recognition

- Locate and classify named entities in sentences:
  - 4 categories: organization, person, location, misc.
  - e.g. "U.N. official Richard Butler heads for Baghdad".
- CoNLL 03 data set (200K words train, 50K words test)

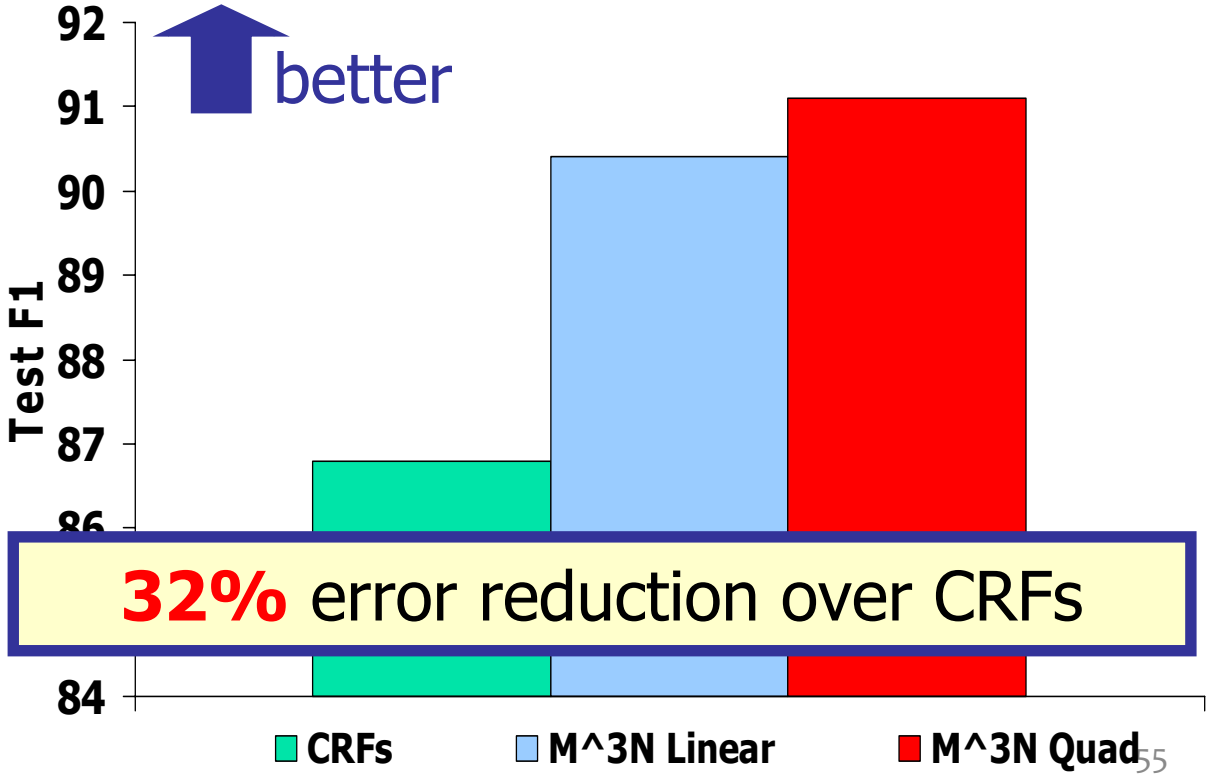


y_i = org/per/loc/misc/none

f(y_i, x) = [...,
I(y_i=org, x_i="U.N."),
I(y_i=per, x_i=capitalized),
I(y_i=loc, x_i=known city),
...,]

y

x



Associative Markov networks

$$P(\mathbf{y} \mid \mathbf{x}) \propto \underbrace{\prod_i \phi_i(y_i, \mathbf{x}_i)}_{\text{Point features}} \underbrace{\prod_{ij} \phi_{ij}(y_i, y_j, \mathbf{x}_{ij})}_{\text{Edge features}} = \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})\}$$

spin-images, point height

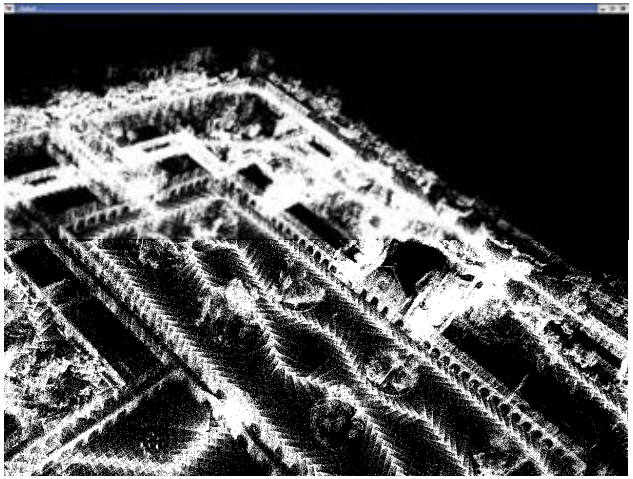
length of edge, edge orientation

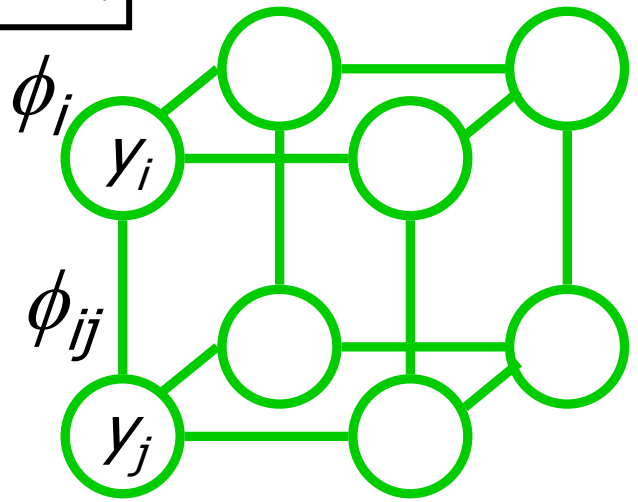
"associative" restriction

$$\phi_{ij}(y_i, y_j) = \begin{bmatrix} \phi_{ij}(1,1) & & 1 \\ & \ddots & \\ 1 & & \phi_{ij}(K,K) \end{bmatrix}$$

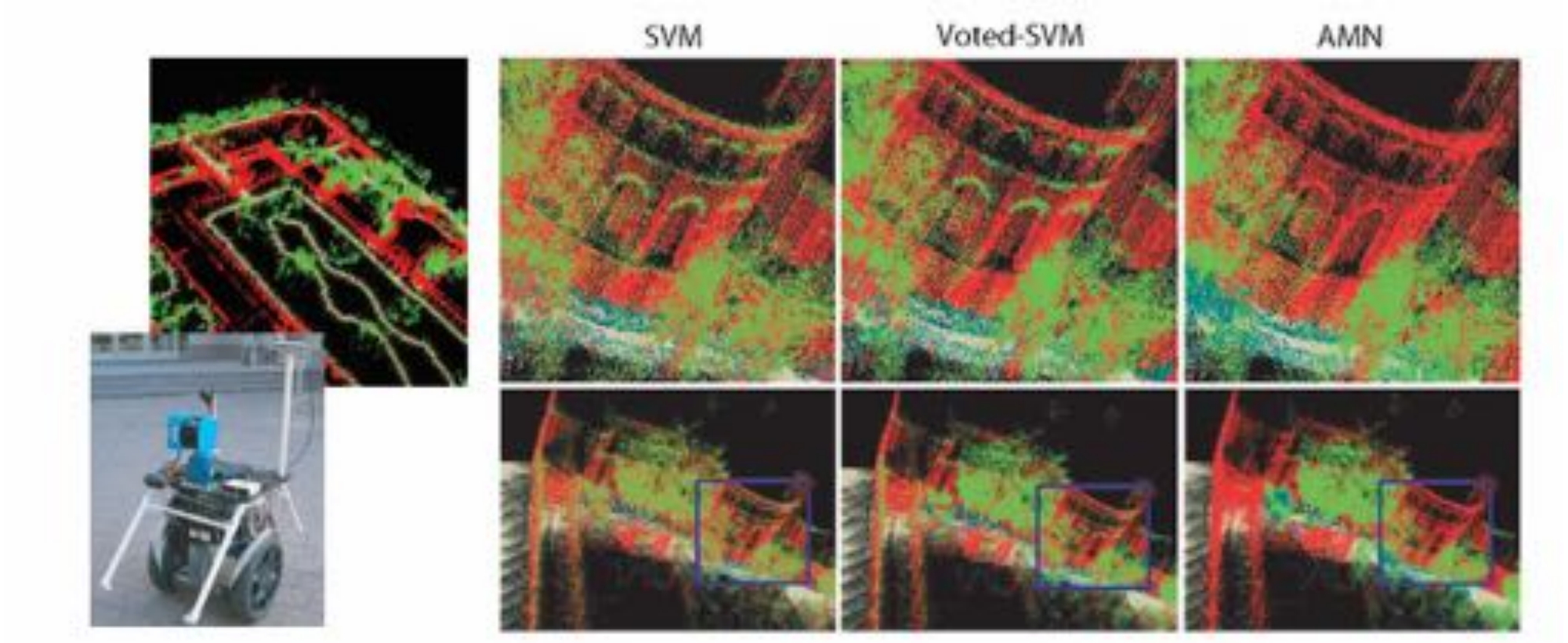
bonus

$$\phi_{ij}(k,k) \geq 1$$





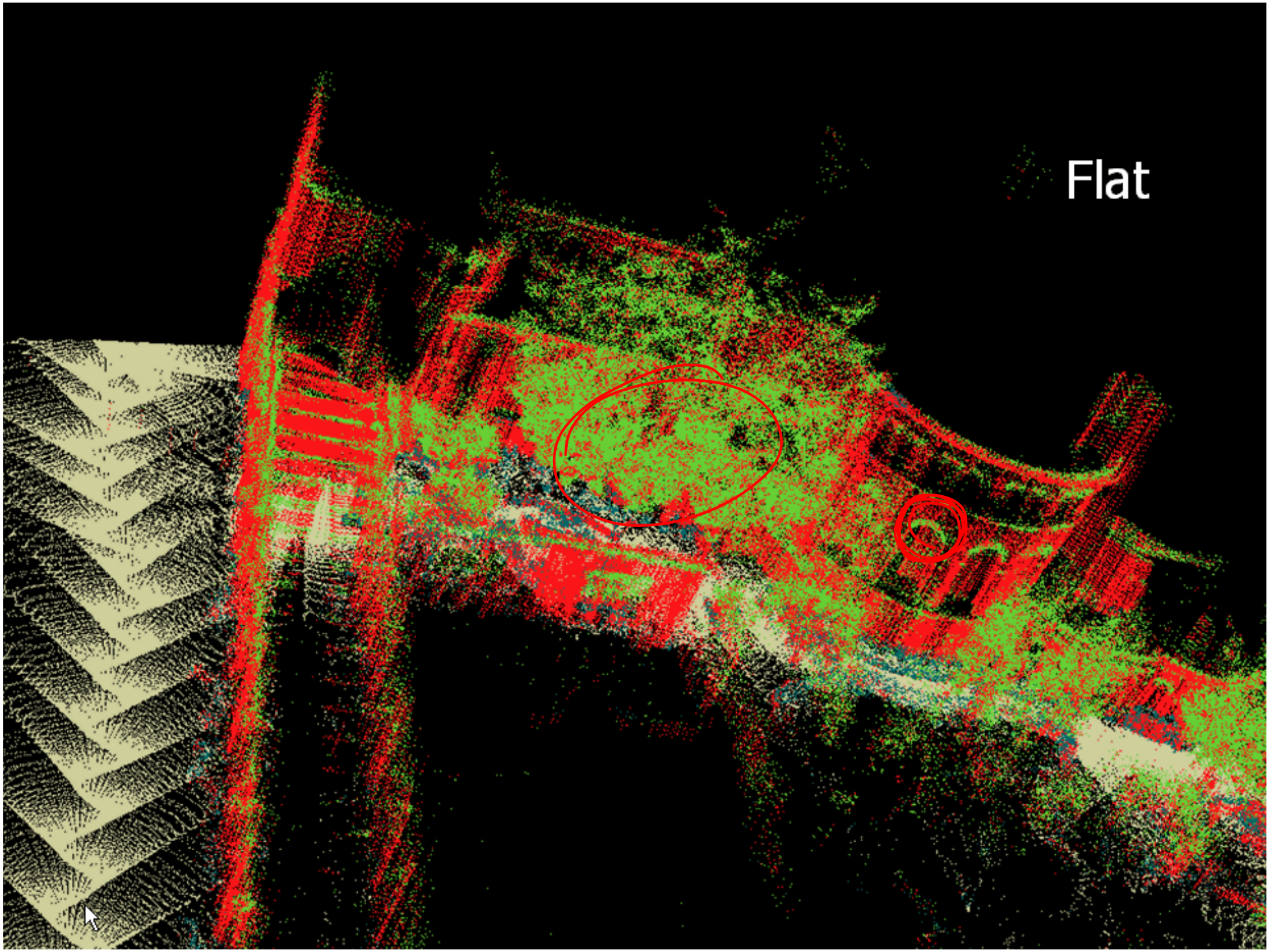
Max-margin AMNs results



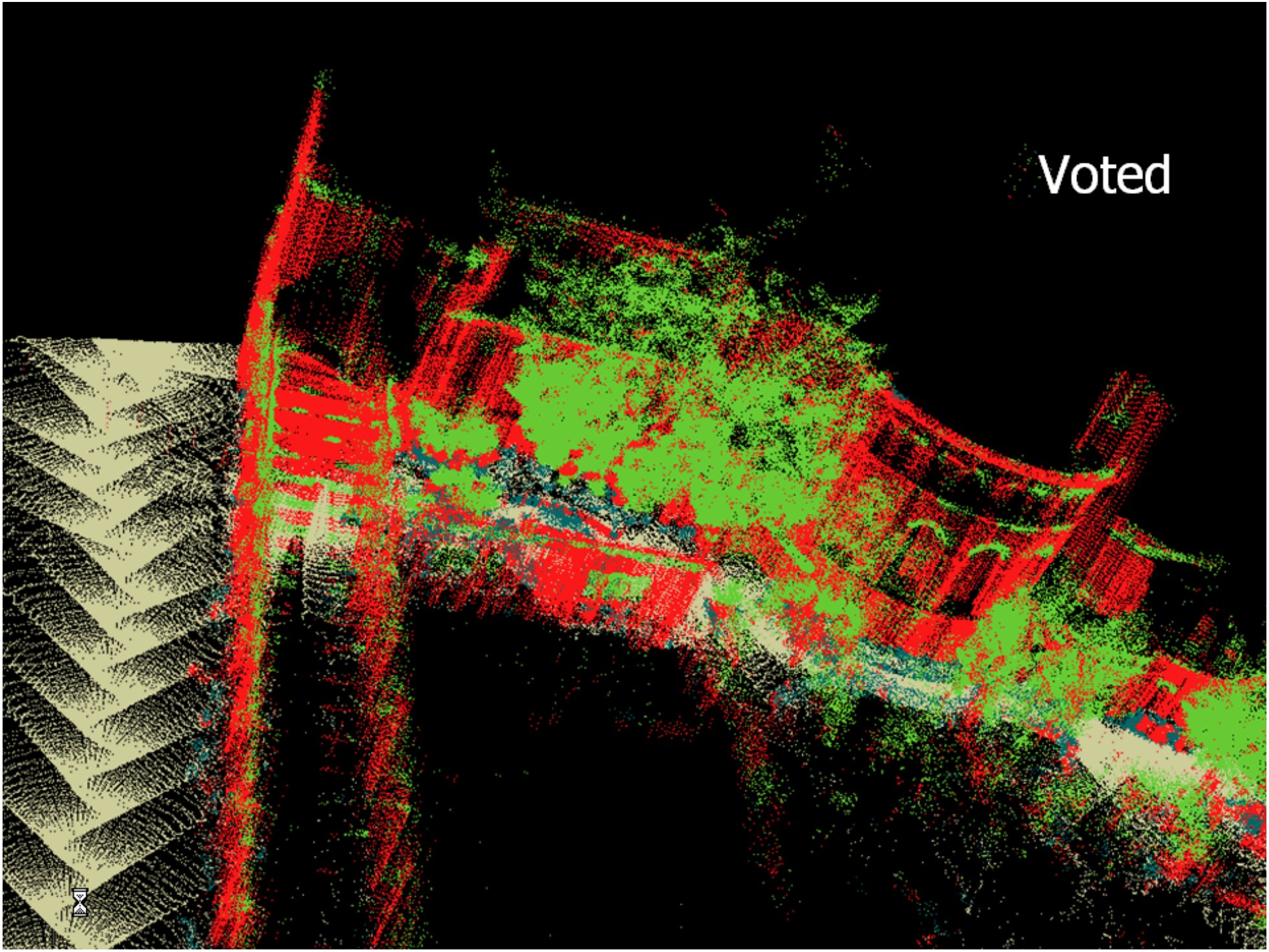
Label: ground, building, tree, shrub

Training: 30 thousand points Testing: 3 million points

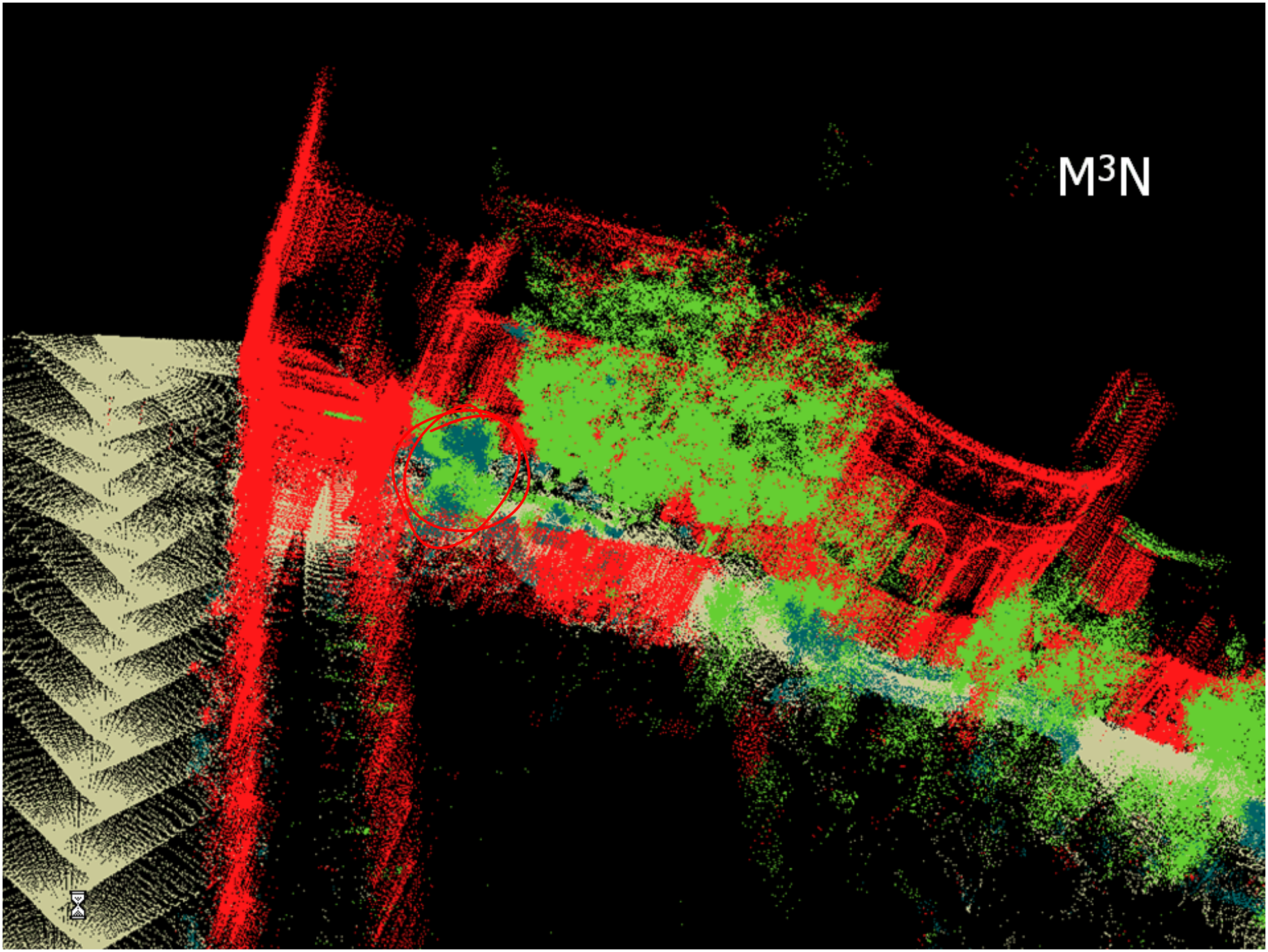
Slide from Guestrin, 10-701, 2005



Slide from Guestrin, 10-701, 2005



Slide from Guestrin, 10-701, 2005



Segmentation results

Hand labeled 180K test points

Model	Accuracy
SVM	68%
V-SVM	73%
M^3N	93%