Sequential design of discriminant functions

Classification with Control

Djalel Benbouzid
with Balázs Kégl and Róbert Busa-Fekete
benbouzid @ lal.in2p3.fr

Laboratoire de l'Accélérateur Linéaire, Univ. Paris-Sud, CNRS/IN2P3

January, 26th 2012

Outline

Original motivation

- Application in high energy particles detectors (*triggers*).
- Huge amount of data to classify.
- Imbalanced data distributions.

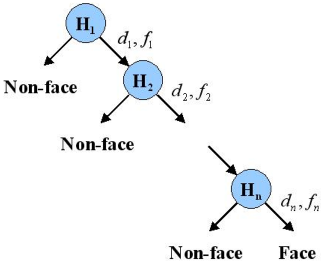
- *Accuracy* and *classification* speed are both requirements.

Classifier cascade

- P. Viola & M. Jones (2001).
- Motivated by real time face detection.
- Three characteristics :
  - The cascade architecture.
  - Feature selection through Adaboost.
  - Cheap features : Haar-like features.

Classifier cascade

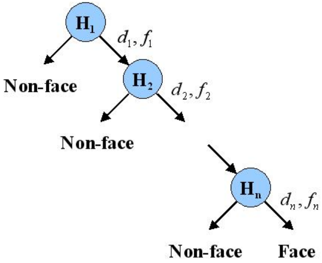
- P. Viola & M. Jones (2001).
- Motivated by real time face detection.
- Three characteristics :
  - The cascade architecture.



  - Feature selection through Adaboost.
  - Cheap features : Haar-like features.

Classifier cascade

- P. Viola & M. Jones (2001).
- Motivated by real time face detection.
- Three characteristics :
    - The cascade architecture.
    - Feature selection through Adaboost.

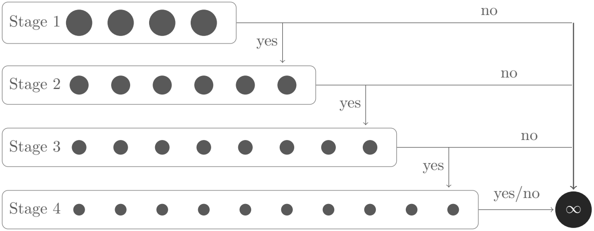


    - Cheap features : Haar-like features.

Classifier cascade

- P. Viola & M. Jones (2001).
- Motivated by real time face detection.
- Three characteristics :
  - The cascade architecture.
  - Feature selection through Adaboost.
  - Cheap features : Haar-like features.

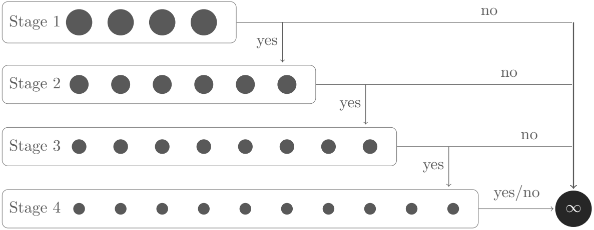
Drawbacks and follow-ups



- A set of stages H_j and thresholds $\theta_j, j = 1, \ldots, N$
- A stage is an AdaBoost strong classifier (predictor) $\mathbf{f}_j \mapsto \mathbb{R}$
- Basic controller : two actions = { Quit with -1, Carry on }

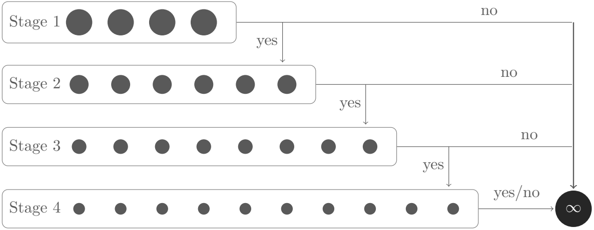
$$H_j(\mathbf{x}) = \begin{cases} -1 & \text{if } \mathbf{f}_j(\mathbf{x}) < \theta_j \\ H_{j+1} & \text{else} \end{cases} \quad \text{with } H_N(\mathbf{x}) = \begin{cases} -1 & \text{if } \mathbf{f}_N(\mathbf{x}) < \theta_j \\ +1 & \text{else} \end{cases}$$

Drawbacks and follow-ups



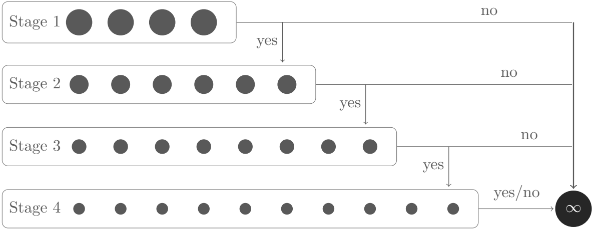
- No early classification for positives.
- The margin information is lost.
- Hand-tuning of the hyper-parameters.
- Bootstrapping the data during the learning.
- No straightforward extension to multi-class classification.

Drawbacks and follow-ups



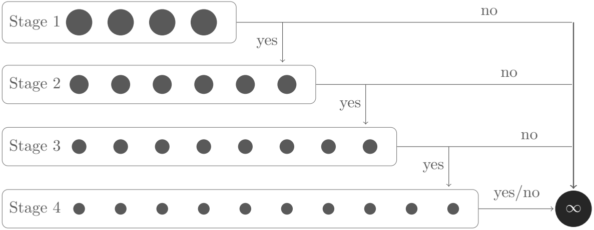
- No early classification for positives.
- The margin information is lost.
- Hand-tuning of the hyper-parameters.
- Bootstrapping the data during the learning.
- No straightforward extension to multi-class classification.

Drawbacks and follow-ups



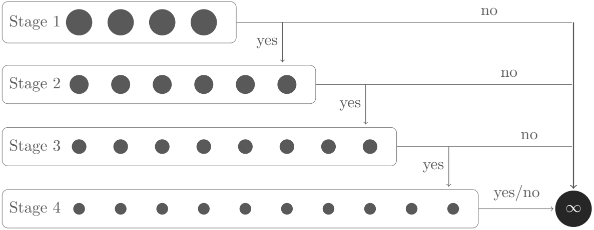
- No early classification for positives.
- The margin information is lost.
- Hand-tuning of the hyper-parameters.
- Bootstrapping the data during the learning.
- No straightforward extension to multi-class classification.

Drawbacks and follow-ups



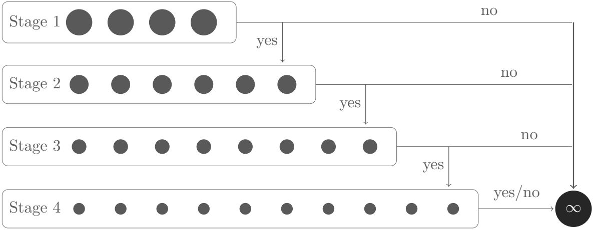
- No early classification for positives.
- The margin information is lost.
- Hand-tuning of the hyper-parameters.
- Bootstrapping the data during the learning.
- No straightforward extension to multi-class classification.

Drawbacks and follow-ups



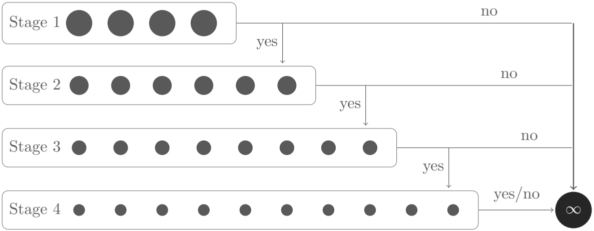
- No early classification for positives.
- The margin information is lost.
- Hand-tuning of the hyper-parameters.
- Bootstrapping the data during the learning.
- No straightforward extension to multi-class classification.

Drawbacks and follow-ups

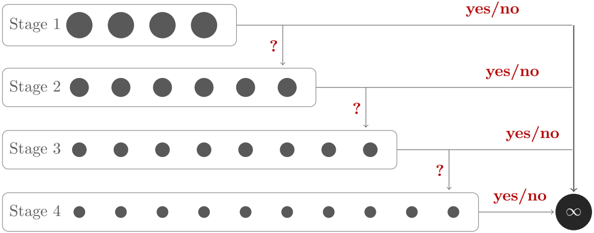


- No early classification for positives.
- The margin information is lost.
- Hand-tuning of the hyper-parameters.
- Bootstrapping the data during the learning.
- No straightforward extension to multi-class classification.

Early classify positives



Early classify positives

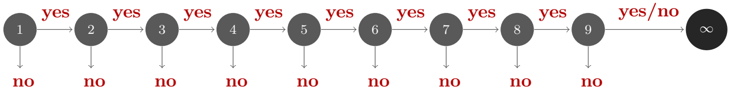


- B Póczos, Y Abbasi-Yadkori, C Szepesvári (2009)
- Controller actions $= \begin{cases} \text{Quit with -1/+1} \\ \text{Evaluate and keep going} \end{cases}$

$$H_j(\mathbf{x}) = \begin{cases} -1 & \text{if } F_j(\mathbf{x}) < \alpha_j \\ +1 & \text{if } F_j(\mathbf{x}) > \beta_j \\ H_{j+1} & \text{else} \end{cases} \text{ with } H_N(\mathbf{x}) = \begin{cases} -1 & \text{if } F_N(\mathbf{x}) < \theta_j \\ +1 & \text{else} \end{cases}$$

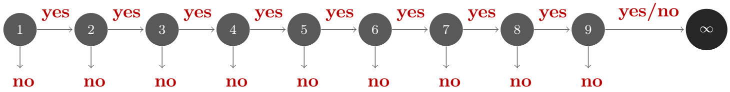
Keep the margin information

- Embedded cascade (L. Bourdev, J. Brandt, 2005)

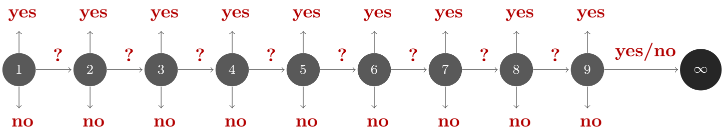


Keep the margin information

- Embedded cascade (L. Bourdev, J. Brandt, 2005)



- Waldboost (J. Sochman, J. Matas, 2005)



So far...

- ~~No early classification for positives.~~
- ~~The margin information is lost.~~
- ~~Hand-tuning of the hyper-parameters.~~
- Bootstrapping the data during the learning.
- No straightforward extension to multi-class classification.

So far...

- ~~No early classification for positives.~~
- ~~The margin information is lost.~~
- ~~Hand-tuning of the hyper-parameters.~~
- Bootstrapping the data during the learning.
- No straightforward extension to multi-class classification.
- Not data-dependant.

Data-dependant ?

- Let the example choose a subset of weak classifiers

Data-dependant ?

- Let the example choose a subset of weak classifiers
- Put otherly, let it choose to skip some...

Data-dependant ?

- Let the example choose a subset of weak classifiers
- Put otherly, let it choose to skip some...
- Controller actions = {Evaluate, Skip, Quit}

Data-dependant ?

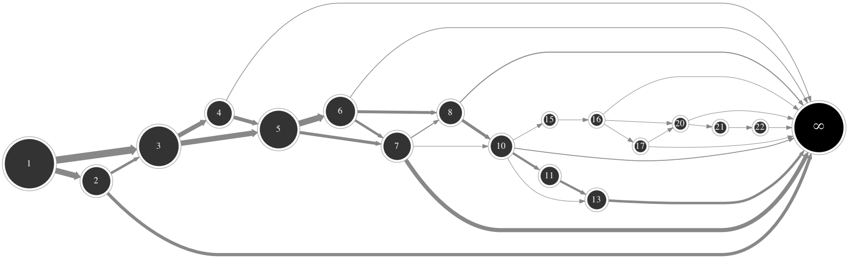
- Let the example choose a subset of weak classifiers
- Put otherly, let it choose to skip some...
- Controller actions = {Evaluate, Skip, Quit}
- The result :

Data-dependant ?

- Let the example choose a subset of weak classifiers
- Put otherly, let it choose to skip some...
- Controller actions = {Evaluate, Skip, Quit}
- The result :

Data-dependant ?

- Let the example choose a subset of weak classifiers
- Put otherly, let it choose to skip some...
- Controller actions = {Evaluate, Skip, Quit}
- The result :



Outline

The setup (I)

Assumption A set of K-class features (weak classifiers) $\mathcal{H}$

$$\mathcal{H} = (\mathbf{h}_1, \ldots, \mathbf{h}_N) ,$$

$$\mathbf{h}_j : \mathcal{X} \to \mathbb{R}^K, j = 1, \ldots, N$$

Goal A sparse, data-dependant classifier built from $\mathcal{H}$

The setup (2)

Assumption
- AdaBoost.MH satisfies the assumption
- Predictor : $\mathbf{f}(\mathbf{x}) = \sum_{j=1}^{N} \mathbf{h}_j(\mathbf{x})$
- $\mathbf{h}_j$ are sorted in order of performance

Goal
- Learn a controller π
- Actions = {Eval, Skip, Quit}

$$\mathbf{f}(\mathbf{x}) = \sum_{j=1}^{N} b_j^{\pi} \mathbf{h}_t(\mathbf{x})$$

$$b_j^{\pi}(\mathbf{x}) = \mathbb{I}\left\{\pi(.) = \text{Eval} \text{ and } \forall j' < j : \pi(.) \neq \text{Quit}\right\}$$

The setup (2)

Assumption

- AdaBoost.MH satisfies the assumption
- Predictor : $\mathbf{f}(\mathbf{x}) = \sum_{j=1}^{N} \mathbf{h}_j(\mathbf{x})$
- $\mathbf{h}_j$ are sorted in order of performance

Goal

- Learn a controller π
- Actions = {Eval, Skip, Quit}

$$\mathbf{f}(\mathbf{x}) = \sum_{j=1}^{N} b_j^{\pi} \mathbf{h}_t(\mathbf{x})$$

$$b_j^{\pi}(\mathbf{x}) = \mathbb{I}\left\{\pi\big(.\big) = \text{Eval} \;\; \text{and} \;\; \forall j' < j : \pi\big(.\big) \neq \text{Quit}\right\}$$

Markov Decision Processes

An MDP is a 4-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where :

- $\mathcal{S}$: state space, containing initial and terminal states, resp. $\mathbf{s}_1$ and $\mathbf{s}_\infty$
- $\mathcal{A}$: actions set
- $\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$: the transition probabilities
- $\mathcal{R}^a_{ss'} = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}$: the expected value of the next reward r_{t+1} for each state-action pair

Model-free learning methods : Sarsa(λ), Q-Learning(λ)

Markov Decision Processes

An MDP is a 4-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where :

- $\mathcal{S}$: state space, containing initial and terminal states, resp. $\mathbf{s}_1$ and $\mathbf{s}_\infty$
- $\mathcal{A}$: actions set
- $\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$: the transition probabilities
- $\mathcal{R}^a_{ss'} = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}$: the expected value of the next reward r_{t+1} for each state-action pair

Model-free learning methods : Sarsa(λ), Q-Learning(λ)

The state descriptor

$K+1$ state variables : $\big(j, (f_1, \ldots, f_K)\big)$

- The feature index : j
- The current posteriors : $\mathbf{f}_j^\pi(\mathbf{x}) \mapsto \mathbb{R}^K$

$$\mathbf{f}_j^\pi(\mathbf{x}) = \sum_{j'=1}^{j} b_{j'}^\pi(\mathbf{x})\mathbf{h}_{j'}(\mathbf{x})$$
$$= \mathbf{f}_{j-1}^\pi(\mathbf{x}) + b_j^\pi(\mathbf{x})\mathbf{h}_j(\mathbf{x})$$

$$b_j^\pi(\mathbf{x}) = \mathbb{I}\left\{\pi\big(j, \mathbf{f}_{j-1}^\pi(\mathbf{x})\big) = \text{Eval} \ \text{ and } \ \forall j' < j : \pi\big(j', \mathbf{f}_{j'}^\pi(\mathbf{x})\big) \neq \text{Quit}\right\}$$

The policy and the rewards

- Deterministic policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$

$$\pi\Big(\underbrace{j, \quad \overbrace{(f_1, \ldots, f_K)}^{\text{discriminant function outputs}}}_{\text{state descriptor}}\Big) \mapsto \Big\{\underbrace{\text{Eval}, \text{Skip}, \text{Quit}}_{\text{actions}}\Big\}$$

- The rewards
  - Correct classification : $r_t = 1$
  - Penalizing a classification evaluation : $r_t = -\beta, 0 < \beta < 1$
- Objective function

$$\varrho^{\pi} = \mathbb{E}_{(\mathbf{x},\ell)\sim\mathfrak{D}} \left\{ \underbrace{\mathbb{I}\left\{\arg\max_{\ell'} f^{\pi}_{N,\ell'}(\mathbf{x}) = \ell\right\}}_{\text{correct classification}} - \beta \underbrace{\sum_{j=1}^{N} b^{\pi}_j(\mathbf{x})}_{L_0 \text{ penalty}} \right\}$$

State representation (I)

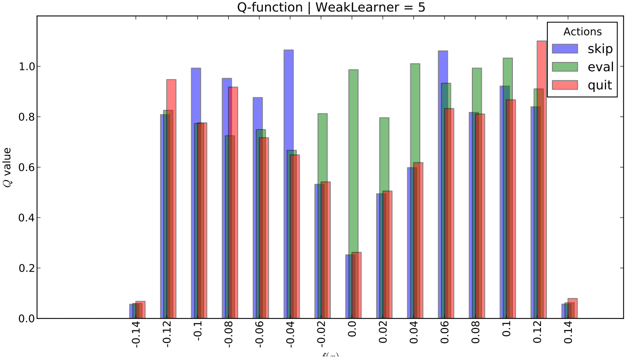
- Action-Value based methods

$$Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\}$$
$$= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \;\middle|\; s_t = s, a_t = a\right\}$$

- For the continuous part of the state $(f_1, \ldots, f_K)$:
  - Discritization (only for the binary case)
  - Function approximation

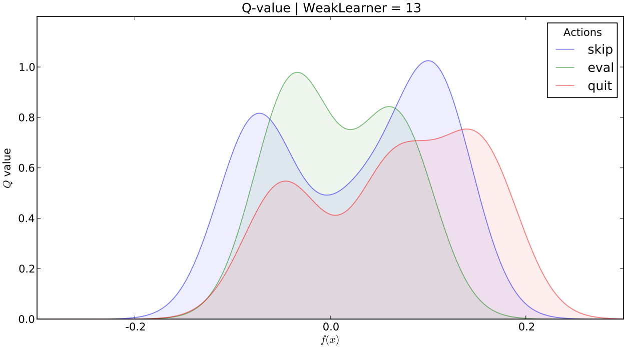
State representation (2)

- Discretization (only for the binary case)



State representation (3)

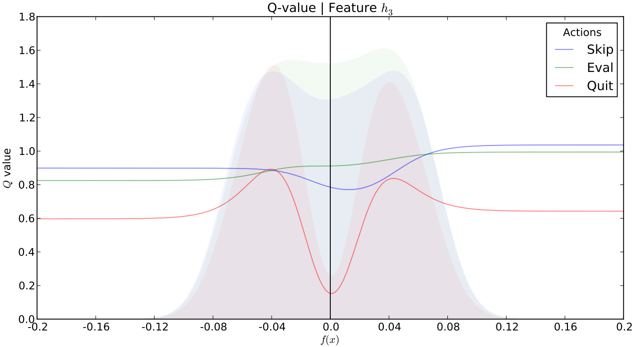
- Function approximation : Radial Basis Function Network



$$a_i(\mathbf{x}) = exp(-1/2(\mathbf{x} - \mathbf{c}_i/\sigma_i)^2)$$
$$\hat{g}(\mathbf{f}; \mathbf{w}, \mathbf{C}, \boldsymbol{\sigma}) = \sum_{i=0}^{n} w_i a_i(\mathbf{f})$$

State representation (4)

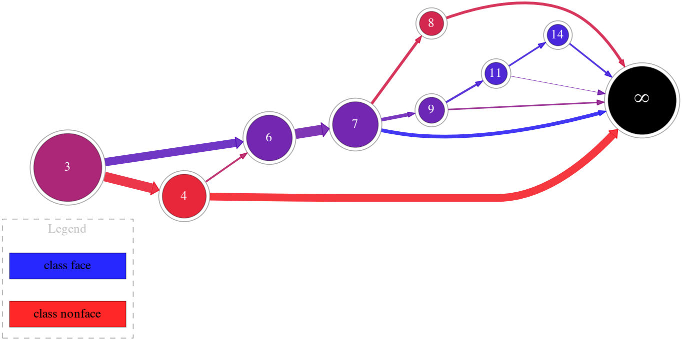
- Function approximation : Gaussian Softmax Basis Function Network (GSBFN)



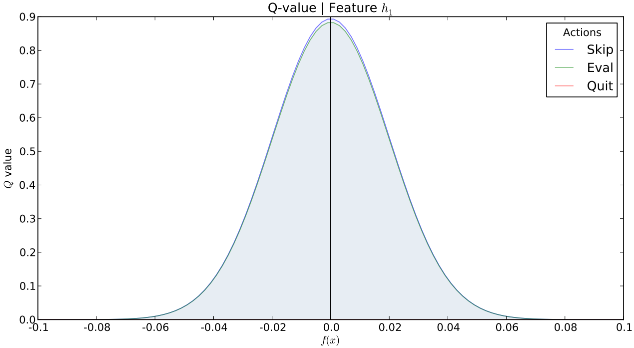
$$\phi_i = \frac{a_i(\mathbf{x})}{\sum a_j(\mathbf{x})}$$
$$\hat{g}(\mathbf{f}; \mathbf{w}, \mathbf{C}, \boldsymbol{\sigma}) = \sum_{i=0}^{n} w_i \phi_i(\mathbf{f})$$

Example

- Face instance 
- Path : 3, 4, 6, 7, 9, ∞

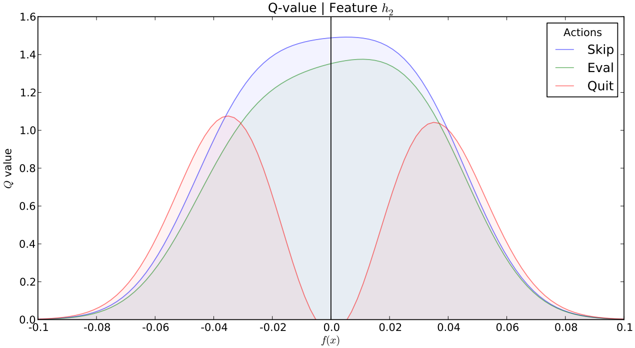


Example



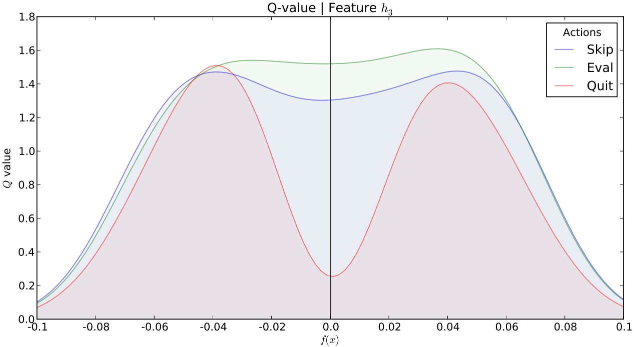
Path : 3, 4, 6, 7, 9, ∞

Example



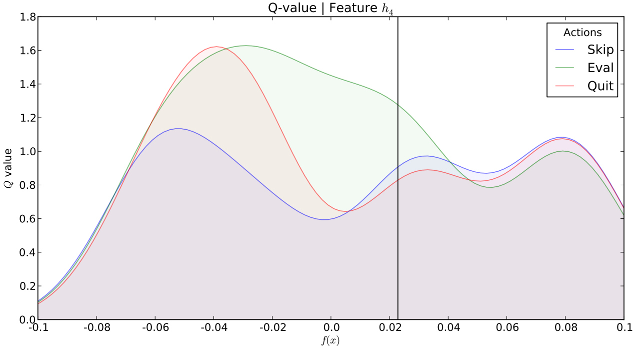
Path : 3, 4, 6, 7, 9, ∞

Example



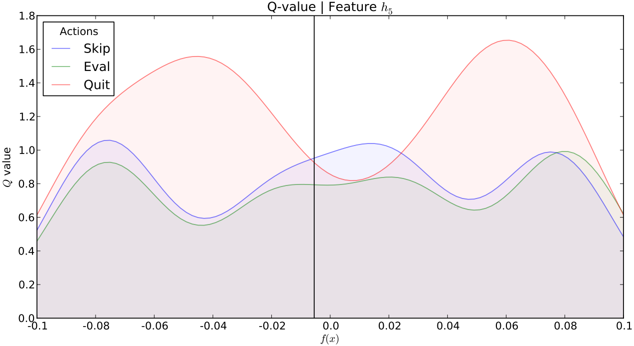
Path : 3, 4, 6, 7, 9, ∞

Example



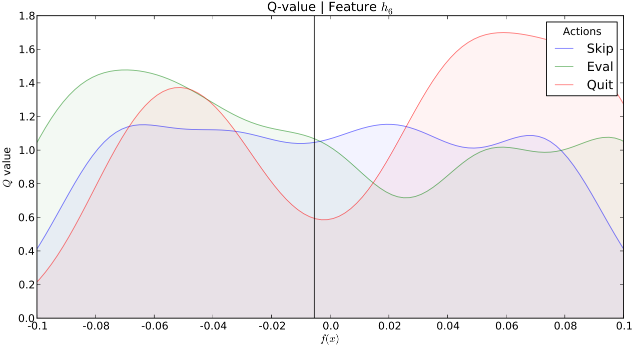
Path : 3, 4, 6, 7, 9, ∞

Example



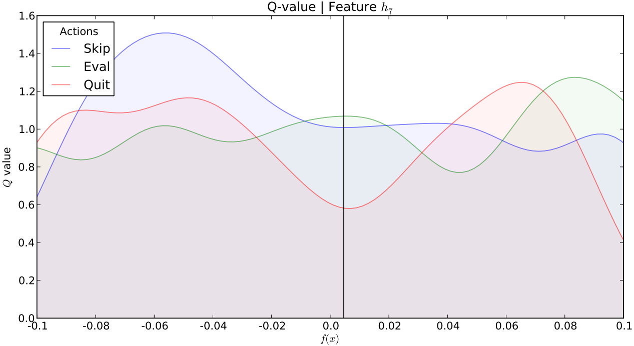
Path : 3, 4, 6, 7, 9, ∞

Example



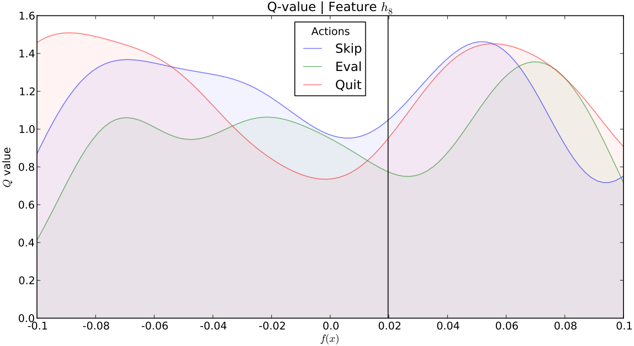
Path : 3, 4, 6, 7, 9, ∞

Example



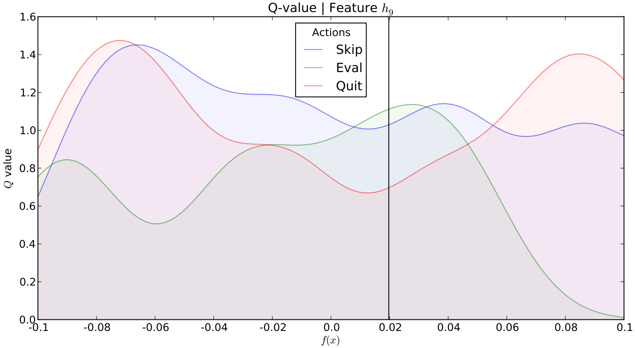
Path : 3, 4, 6, 7, 9, ∞

Example



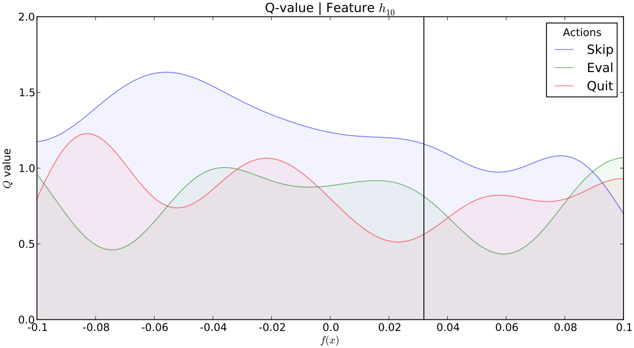
Path : 3, 4, 6, 7, 9, ∞

Example



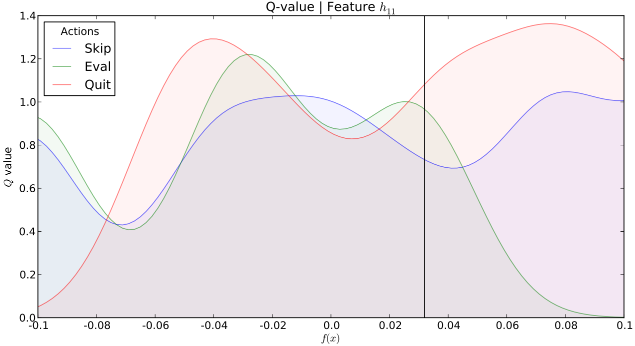
Path : 3, 4, 6, 7, 9, ∞

Example



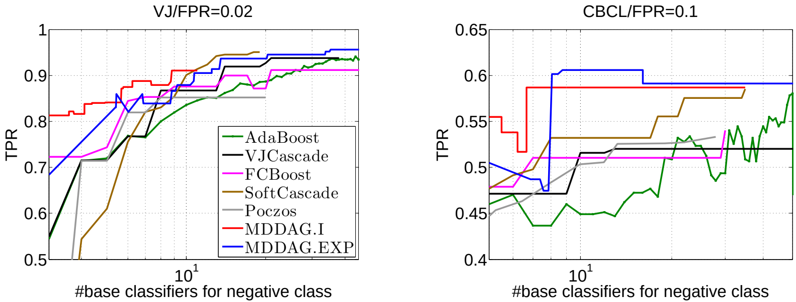
Path : 3, 4, 6, 7, 9, ∞

Example



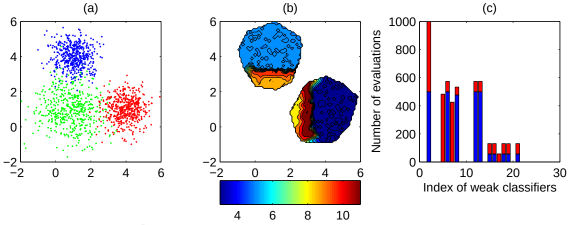
Path : 3, 4, 6, 7, 9, ∞

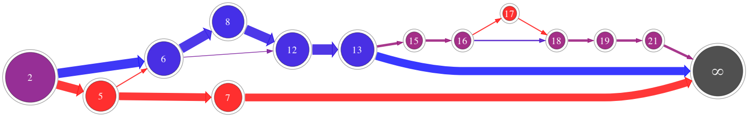
Experiments



Deep structures (I)

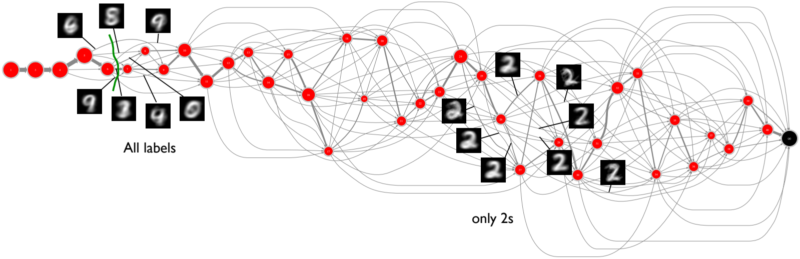
- Toy example



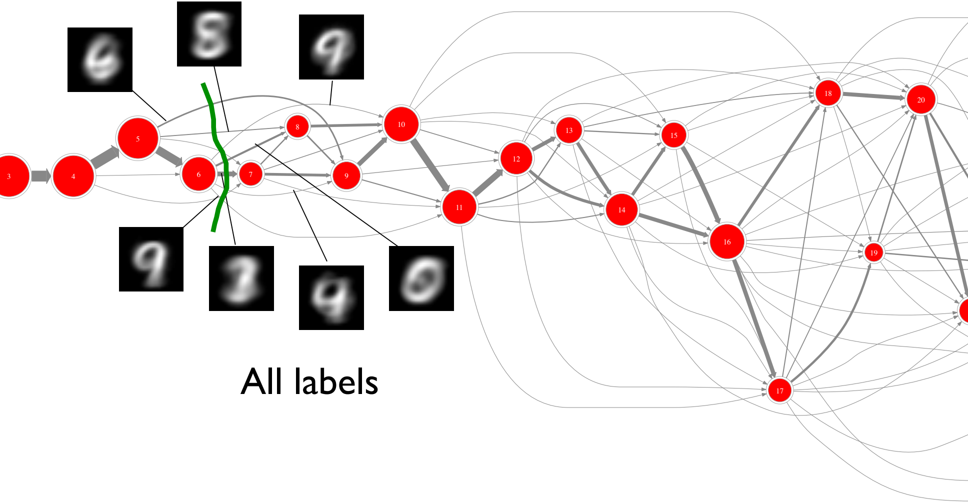


Deep structures (2)

- MNIST

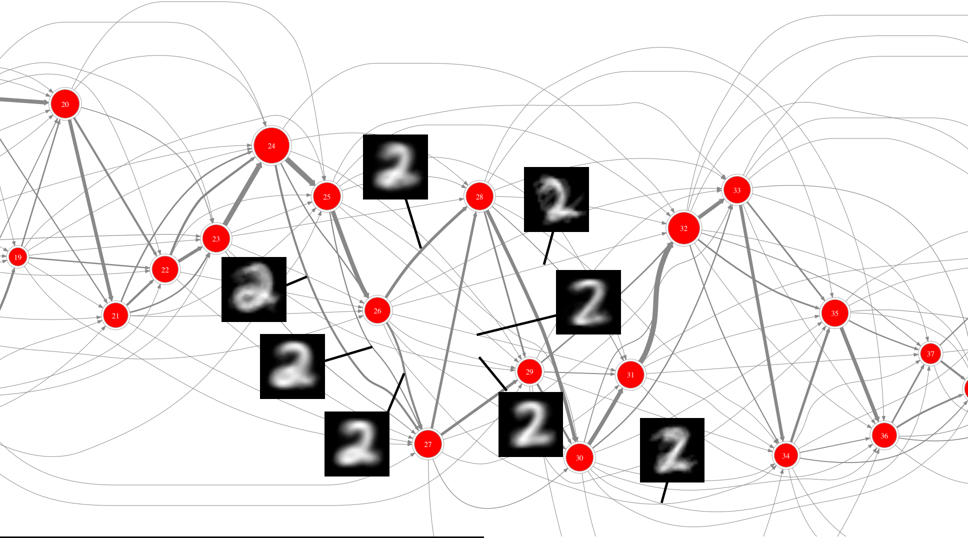


Deep structures (2)



All labels

Deep structures (2)



Conclusion and future works

- Alternative to cascade architectures
- Interesting osmosis between machine learning subdomains
- Data-dependent / Deep structures
- Curse of dimensionality
- Classification-based Policy Iteration

Questions?