Algorithms for Predicting Structured Data

Thomas Gärtner / Shankar Vembu

Fraunhofer IAIS / UIUC

ECML PKDD 2010

Structured Prediction

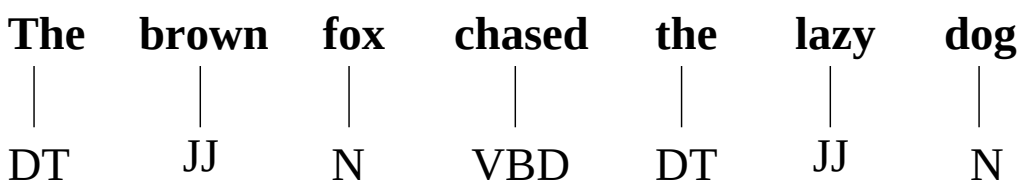
Predicting multiple outputs with complex internal structure and dependencies

Contrast with single-valued prediction problems like binary classification and regression

Part-of-Speech Tagging

Input: Sentence

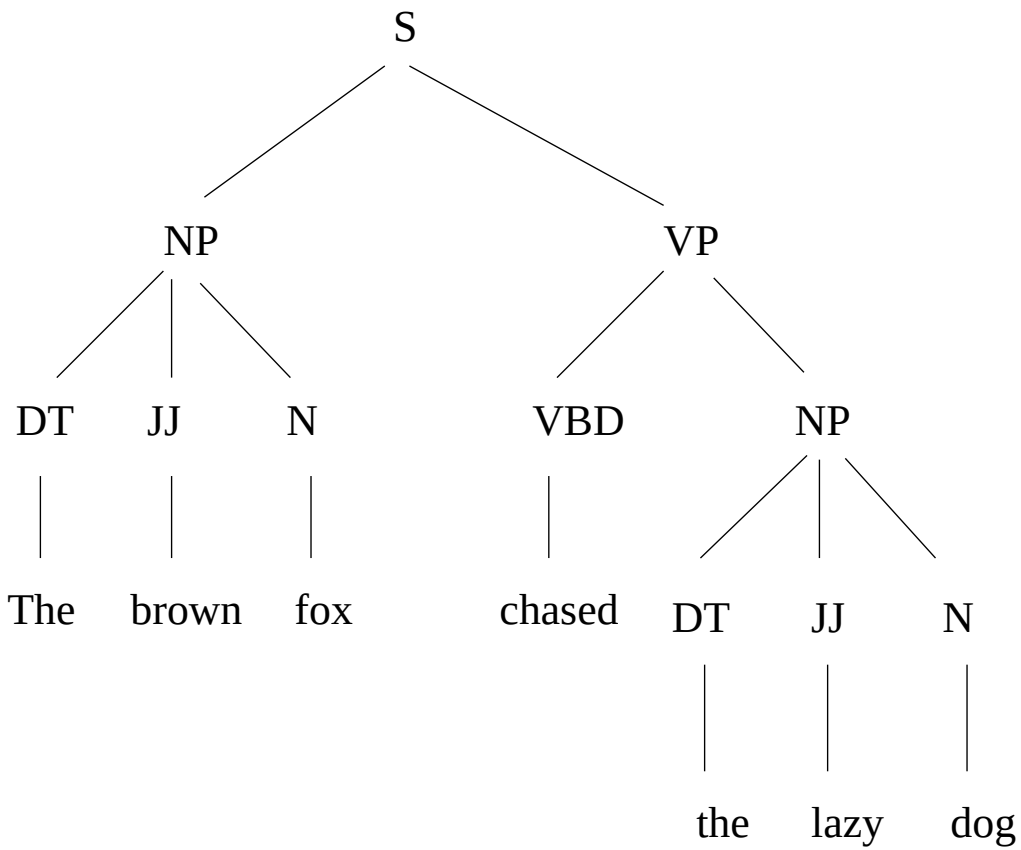
Output: Part-of-speech tags



Linguistic Parsing

Input: Sentence

Output: Parse tree



Machine Learning Problems

Multi-class prediction

Multi-label classification

Hierarchical classification

Label ranking

...

Real World Applications

Natural language processing

Computational biology

Bioinformatics

Computer vision

Robotics

...

Outline

Part I: From Binary to Structured Prediction

- Loss Functions
- Algorithms

Part II: Exact and Approximate Inference

- Approximate Inference: + / - ve Results
- Complexity of Learning
- New Training Methods

Part III: Weakly Supervised Learning

Tutorial Focus

Discriminative methods in structured prediction

Algorithmic techniques

Topics Not Covered

Advanced optimization methods

Connections to reinforcement learning

Learning theory / Generalization bounds

Inference in graphical models

X-supervised learning

Outline

Road Map

Perceptron $\rightarrow$ Structured perceptron

Regression $\rightarrow$ Kernel dependency estimation

Support Vector Machines $\rightarrow$ Structured SVMs

Logistic Regression $\rightarrow$ Conditional random fields

Preliminaries

Supervised Learning

Input and output spaces $\mathcal{X}, \mathcal{Y}$

Classification $\mathcal{Y} = \{-1, +1\}$; Regression $\mathcal{Y} = \mathbb{R}$

Training data $X = (x_1, \ldots, x_m) \in \mathcal{X}^m$
$Y = (y_1, \ldots, y_m) \in \mathcal{Y}^m$

(drawn i.i.d. from an unknown distribution)

Function approximation $f : \mathcal{X} \to \mathcal{Y}$

Good performance on unseen examples where performance is measured w.r.t. a loss function

Generative and Discriminative Learning

Generative learning (e.g., Naïve Bayes, HMM)

- Model joint probability $p(x, y)$
- Predict using Bayes rule

$$p(y \mid x) = \frac{p(x, y)}{\sum_{z \in \mathcal{Y}} p(x, z)} = \frac{p(x \mid y)p(y)}{\sum_{z \in \mathcal{Y}} p(x, z)}$$

Discriminative learning (e.g., Logistic regression, CRF)

- Model conditional probability $p(y \mid x)$ OR
- Mapping from inputs to outputs $f : \mathcal{X} \to \mathcal{Y}$

Regularized Risk Minimization

$$\operatorname*{argmin}_{f \in \mathcal{H}} \sum_{i=1}^{m} \ell(f(x_i), y_i) + \lambda \Omega(f)$$

$\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, non-negative loss function

$\Omega : \mathcal{H} \to \mathbb{R}_+$, regularization function

$\lambda > 0$, regularization parameter

Regularized Risk Minimization

Example: Linear classifiers $f(x) = \langle w, \phi(x) \rangle$

$$\operatorname*{argmin}_{w \in \mathbb{R}^n} \sum_{i=1}^{m} \ell(\langle w, \phi(x_i) \rangle, y_i) + \lambda \|w\|_2^2$$

$\phi : \mathcal{X} \to \mathbb{R}^n$, feature map

Structured Prediction

Input and output spaces $\mathcal{X}, \mathcal{Y}$

Training data $X = (x_1, \ldots, x_m) \in \mathcal{X}^m$
$Y = (y_1, \ldots, y_m) \in \mathcal{Y}^m$

Joint scoring function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$

Prediction $\hat{y} = \underset{y \in \mathcal{Y}(x)}{\operatorname{argmax}} f(x, y) = \underset{y \in \mathcal{Y}(x)}{\operatorname{argmax}} \langle w, \phi(x, y) \rangle$

Joint Feature Maps

Extract features from inputs AND outputs

Feature map, $\phi : \mathcal{X} \times \mathcal{Y} \to \mathcal{H}$

Joint input-output kernel,
$k[(x, y), (x', y')] = \langle \phi(x, y), \phi(x', y') \rangle$

Joint Feature Maps

Example: Multi-label classification, Hierarchical classification

$$x \in \mathbb{R}^n, \quad y \in \{0,1\}^d$$

$$\phi(x,y) = x \otimes y$$

Tensor product kernel: $k[(x,y),(x',y')] = k_{\mathcal{X}}(x,x')k_{\mathcal{Y}}(y,y')$

Loss Functions

(Binary $\rightarrow$ Structured)

Regularized Risk Minimization

$$\operatorname*{argmin}_{f \in \mathcal{H}} \sum_{i=1}^{m} \ell(f(x_i), y_i) + \lambda \Omega(f)$$

$\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, non-negative loss function

$\Omega : \mathcal{H} \to \mathbb{R}_+$, regularization function

$\lambda > 0$, regularization parameter

Zero-One Loss

Binary:

$$\ell_{\text{0-1}}(y, z) = \begin{cases} 0 & \text{if } y = z \\ 1 & \text{otherwise} \end{cases}$$

Structured:

$$\ell^{\Delta}_{\max}(f, (x, y)) = \Delta(\operatorname*{argmax}_{z \in \mathcal{Y}} f(x, z), y)$$

Non-convex, non-differentiable, hard to optimize

Use surrogate losses instead; convex upper bounds

Squared Loss

$$\ell_{\text{square}}(y, z) = (y - z)^2, \quad \mathcal{Y} = \mathbb{R}$$

Used in regression problems

Extension to structured prediction is non-trivial due to inter-dependencies among the multiple outputs (more on this later)

Hinge Loss

Binary:

$$\ell_{\text{hinge}}(y, z) = \max(0, 1 - yz)$$

Structured:

$$\ell^{\Delta}_{\text{hinge}}(f, (x, y)) = \max_{z \in \mathcal{Y}} [\Delta(z, y) + f(x, z) - f(x, y)]$$

Logistic Loss

Binary:

$$\ell_{\log}(y, z) = \ln(1 + \exp(-yz))$$

Structured:

$$\ell_{\log}(f, (x, y)) = \ln \left[\sum_{z \in \mathcal{Y}} \exp(f(x, z)) \right] - f(x, y)$$

Exponential Loss

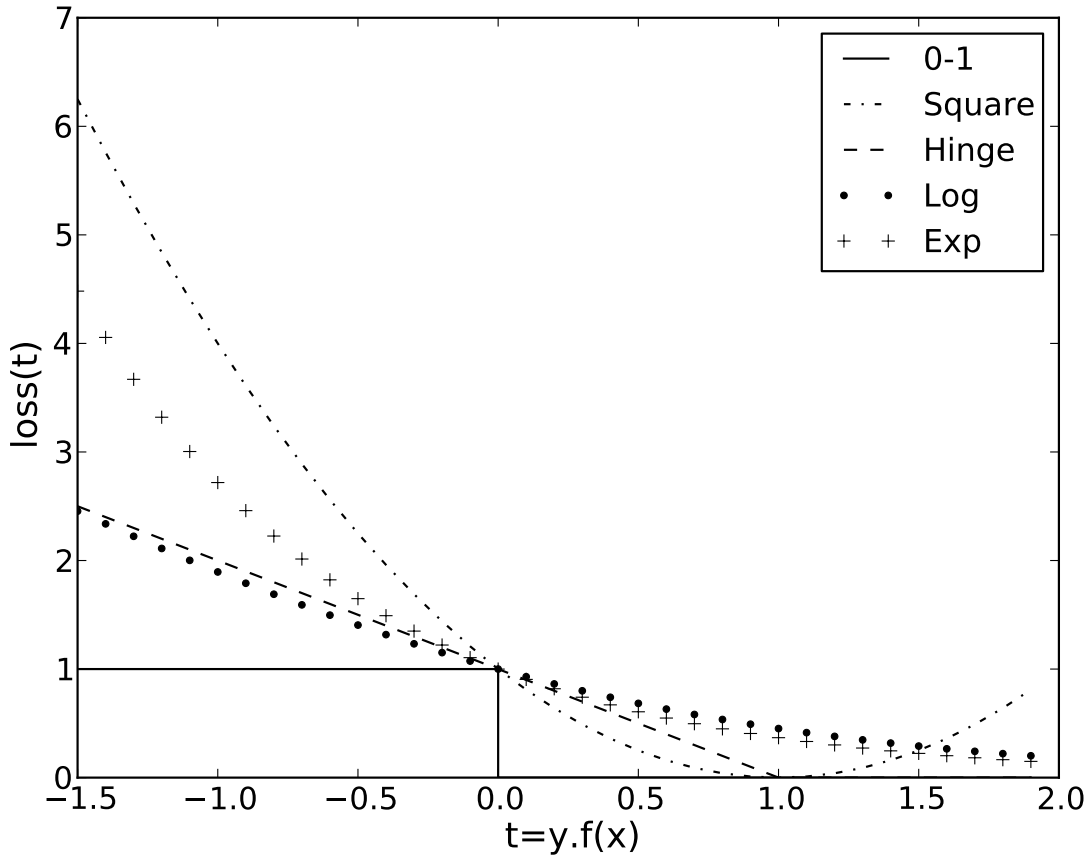
Binary:

$$\ell_{\exp}(y, z) = \exp(-yz)$$

Structured:

$$\ell_{\exp}(f, (x, y)) = \sum_{z \in \mathcal{Y}} \exp\left[f(x, z) - f(x, y)\right]$$

Loss Functions



Learning Algorithms

(parameter estimation)

Perceptron $\rightarrow$ Structured perceptron

Perceptron

Online learning algorithm

Prediction $\hat{y} = f(x) = \operatorname{sgn} \langle w, x \rangle$

Algorithm:

Initialize $w = \mathbf{0}$

For $t = 1 \dots T$

1. receive input x_t
2. predict $\hat{y}_t = \operatorname{sgn} \langle w, x_t \rangle$
3. receive true label $y_t \in \{-1, +1\}$
4. if $y_t \neq \hat{y}_t$, then $w \leftarrow w + y_t \cdot x_t$

Mistake Bound

Separable data [Block, 62; Novikoff, 62]

Theorem

Let $(x_1, y_1), \ldots, (x_m, y_m)$ be a sequence of training examples with $\|x_i\| \leq R$ for all $i \in [\![m]\!]$. Suppose there exists a unit norm vector u such that $y_i(\langle u, x_i \rangle) \geq \gamma$ for all the examples. Then the number of mistakes made by the perceptron algorithm on this sequence is at most $(R/\gamma)^2$.

Mistake Bound

Inseparable data [Freund and Schapire, 99]

Theorem

Let $(x_1, y_1), \ldots, (x_m, y_m)$ *be a sequence of training examples with* $\|x_i\| \leq R$ *for all* $i \in [\![m]\!]$*. Let* u *be any unit norm weight vector and let* $\gamma > 0$*. Define the deviation of each example as* $d_i = \max[0, \gamma - y_i(\langle u, x_i \rangle)]$ *and define* $D = \sqrt{\sum_{i=1}^{m} d_i^2}$*. Then the number of mistakes made by the perceptron algorithm on this sequence of examples is at most*

$$\frac{(R+D)^2}{\gamma^2} \, .$$

Constraint Classification

(Har-Peled et al., ALT 2002)

Generalizes perceptron for various problems including multiclass prediction, multilabel classification and label ranking

Example: Multiclass prediction, $\mathcal{Y} = \{1, \ldots, d\}$

Weights $(w_1, \ldots, w_d) \in \mathbb{R}^{nd}$

Prediction $\hat{y} = f(x) = \text{argmax}_{i \in [\![d]\!]} \langle w_i, x \rangle$

Constraint Classification

Algorithm:

Initialize $(w_1, \ldots, w_d) = \mathbf{0}$

For $t = 1 \ldots T$

1. receive input x_t and label $y_t \in [\![d]\!]$
2. construct $\tilde{y}_t = \{(y_t, i) \mid i \in [\![d]\!] \setminus y_t\}$
3. for all $(p, q) \in \tilde{y}_t$,
   if $\langle w_p - w_q, x_t \rangle < 0$, then
   $w_p \leftarrow w_p + x$,
   $w_q \leftarrow w_q - x$

Structured Perceptron

(Collins, EMNLP 2002)

Linear scoring function $f(x, y) = \langle w, \phi(x, y) \rangle$

Algorithm:

Initialize $w = \mathbf{0}$

For $t = 1 \ldots T$

1. receive input x_t
2. predict $\hat{y}_t = \text{argmax}_{y \in \mathcal{Y}} f(x_t, y)$
3. receive true label $y_t \in \mathcal{Y}$
4. $w \leftarrow w + \phi(x_t, y_t) - \phi(x_t, \hat{y}_t)$

Mistake Bound

Separable data [Collins, 02]

Theorem

Let $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ *be a sequence of training examples which is separable with margin* γ*. Let* R *denote a constant that satisfies* $\|\phi(x_i, y_i) - \phi(x_i, z)\| \leq R$*, for all* $i \in [\![m]\!]$*, and for all* $z \in G\bar{E}N(x_i)$*. Then the number of mistakes made by the perceptron algorithm on this sequence of examples is at most* R^2/γ^2*.*

Mistake Bound

Inseparable data [Collins, 02]

For a training example (x_i, y_i) and a v, γ pair, define $m_i = \langle v, \phi(x_i, y_i) \rangle - \max_{z \in \bar{\text{GEN}}(x_i)} \langle v, \phi(x_i, z) \rangle$ and $\epsilon_i = \max\{0, \gamma - m_i\}$ and define $D_{v,\gamma} = \sqrt{\sum_{i=1}^{m} \epsilon_i^2}$. Then the number of mistakes made by the structured perceptron is at most

$$\min_{v,\gamma} \frac{(R + D_{v,\gamma})^2}{\gamma^2} .$$

Regression $\rightarrow$ KDE

Regularized Least-Squares Regression

OPT: $\min_{w \in \mathbb{R}^n} \lambda \|w\|^2 + \sum_{i=1}^{m} (w^\top x_i - y_i)^2$

Solution: $w^* = (X^\top X + \lambda \mathbf{I})^{-1} X^\top y$

where
$X \in \mathbb{R}^{m \times n}$: data/input matrix
$y \in \mathbb{R}^{m \times 1}$: label/output vector
$\mathbf{I}$: identity matrix

Kernelized Version

OPT1: $\min_{f \in \mathcal{H}} \lambda \|f\|^2 + \sum_{i=1}^{m} (f(x_i) - y_i)^2$

Representer theorem: $f^* = \sum_{i=1}^{m} c_i k(x_i, \cdot)$

OPT2: $\min_{c \in \mathbb{R}^m} \lambda c^\top K c + \|y - Kc\|^2$

Solution: $c^* = (K + \lambda \mathbf{I})^{-1} y$

Kernel Dependency Estimation

(Weston et al., NIPS 2002)

Output “feature" map $\psi : \mathcal{Y} \to \mathcal{H}_{\mathcal{Y}}, \quad k_{\mathcal{Y}}(y, y') = \langle \psi(y), \psi(y') \rangle$

Note: Possible to consider a large class of non-linear loss functions in the output space

KDE uses kernel PCA to “decorrelate" the outputs, and trains independent RLSRs

Kernel Dependency Estimation

(Weston et al., NIPS 2002)

Algorithm:

1. decompose $\{\psi(y_i) \mid i \in [\![m]\!]\}$ into k orthogonal directions using KPCA
2. learn $f_j : \phi \to \mathbb{R}_j$ for each direction $j \in [\![k]\!]$ using RLSR
3. predict by solving the pre-image problem:

$$\hat{y}(x) = \operatorname*{argmin}_{y \in \mathcal{Y}} \left([v_1^\top \psi(y), \ldots, v_k^\top \psi(y)] - [f_1(x), \ldots, f_k(x)] \right)^2$$

Kernel Dependency Estimation

(Weston et al., NIPS 2002)

Note: Projection of $\psi(y)$ onto the p^{th} principal component $v_p = \sum_{i=1}^{m} \alpha_i^p \psi(y_i)$ is given by $v_p^\top \psi(y) = \sum_{i=1}^{m} \alpha_i^p k_{\mathcal{Y}}(y_i, y)$, where α^p is the p^{th} eigenvector of the kernel matrix $K_{\mathcal{Y}}$ (after centering it)

SVM $\rightarrow$ Structured SVM

Support Vector Machine

Linear large-margin classifier $f(x) = \langle w, \phi(x) \rangle$

Minimizes the hinge loss

OPT: $\min_w \lambda \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \langle w, \phi(x_i) \rangle\}$

OPT with slack variables:

$$\begin{aligned} \min_{w,\xi} \quad & \lambda \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i \\ \text{s.t. :} \quad & y_i \langle w, \phi(x_i) \rangle \geq 1 - \xi_i, \quad \forall i \in [\![m]\!] \\ & \xi_i \geq 0, \quad \forall i \in [\![m]\!] \end{aligned}$$

Support Vector Machine

SVM dual

$$\begin{aligned} \min_{c \in \mathbb{R}^m} \quad & \tfrac{1}{2} c^\top Y K Y c - \mathbf{1}^\top c \\ \text{s.t.:} \quad & 0 \leq c_i \leq \tfrac{1}{m\lambda}, \quad \forall i \in [\![m]\!] \end{aligned}$$

Representer theorem: $f(\cdot) = \sum_{i=1}^{m} c_i k(x_i, \cdot)$

Structured SVM / Max-margin Markov Network

Minimizes structured hinge loss

Two formulations:

1. Slack rescaling
2. Margin rescaling

Structured SVM

Slack rescaling

$$\min_{w,\xi} \quad \lambda\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i$$

$$\text{s.t. :} \quad \langle w, \phi(x_i, y_i)\rangle - \langle w, \phi(x_i, z)\rangle \geq 1 - \frac{\xi_i}{\Delta(y_i, z)}, \quad \forall z, \forall i$$

$$\xi_i \geq 0, \quad \forall i \in [\![m]\!]$$

Structured SVM

Margin rescaling

$$\min_{w,\xi} \quad \lambda \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i$$

$$\text{s.t. :} \quad \langle w, \phi(x_i, y_i) \rangle - \langle w, \phi(x_i, z) \rangle \geq \Delta(y_i, z) - \xi_i, \quad \forall z, \forall i$$

$$\xi_i \geq 0, \quad \forall i \in [\![m]\!]$$

Solving the OPT

Major issue: Exponential number of constraints

Suffices to design a sub-routine (loss-augmented inference) to compute

$$\hat{y} = \operatorname*{argmax}_{z \in \mathcal{Y}} [1 - \langle w, \phi(x, y) - \phi(x, z) \rangle] \Delta(z, y)$$

$$\hat{y} = \operatorname*{argmax}_{z \in \mathcal{Y}} [\Delta(z, y) - \langle w, \phi(x, y) - \phi(x, z) \rangle]$$

Iteratively add constraints

Cutting-Plane Method

(Tsochantaridis et al., JMLR 2005)

Input: $(x_1, y_1), \ldots, (x_m, y_m), \lambda, \epsilon$
$S_i \leftarrow \emptyset$, for all $i \in [\![m]\!]$
repeat
  for $i = 1, \ldots, m$ **do**
    $H(y) \equiv \Delta(y_i, y) + w^\top \phi(x_i, y) - w^\top \phi(x_i, y_i)$
    compute $\hat{y} = \operatorname{argmax}_{y \in \mathcal{Y}} H(y)$
    compute $\xi_i = \max\{0, \max_{y \in S_i} H(y)\}$
    if $H(\hat{y}) > \xi_i + \epsilon$ **then**
      $S_i \leftarrow S_i \cup \{\hat{y}\}$
      $w \leftarrow$ optimize primal over $\bigcup_i S_i$
    end if
  end for
until no S_i has changed during iteration

Theoretical Guarantees with Exact Inference

(Tsochantaridis et al., 2005; Finley and Joachims, ICML 2008)

Polynomial time termination Algorithm terminates in a polynomial number of iterations

Correctness Algorithm solves OPT accurately to a desired precision ϵ

Empirical risk bound $\frac{1}{m}\sum_{i=1}^{m}\xi_i$ upper bounds empirical risk

Kernelized SSVM

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^{m|\mathcal{Y}|}} \quad \sum_{i,j\in[\![m]\!],z,z'\in\mathcal{Y}} \boldsymbol{\alpha}_{iz}\boldsymbol{\alpha}_{jz'}k[(x_i,z),(x_j,z')] - \sum_{i,z}\Delta(y_i,z)\boldsymbol{\alpha}_{iz}$$

$$\text{s.t. :} \quad \sum_{z\in\mathcal{Y}} \boldsymbol{\alpha}_{iz} \leq \lambda,\ \forall i \in [\![m]\!]$$
$$\boldsymbol{\alpha}_{iz} \geq 0,\ \forall i \in [\![m]\!],\ \forall z \in \mathcal{Y}$$

Representer theorem:

$$f(\cdot,\cdot) = \sum_{i\in[\![m]\!],z\in\mathcal{Y}} \boldsymbol{\alpha}_{iz}k[(x_i,z),(\cdot,\cdot)]$$

Structured Perceptron Revisited

Linear scoring function $f(x, y) = \langle w, \phi(x, y) \rangle$

Algorithm:

Initialize $w = \mathbf{0}$

For $t = 1 \ldots T$

1. receive input x_t
2. predict $\hat{y}_t = \text{argmax}_{y \in \mathcal{Y}} f(x_t, y)$
3. receive true label $y_t \in \mathcal{Y}$
4. $w \leftarrow w + \phi(x_t, y_t) - \phi(x_t, \hat{y}_t)$

Note: Update rule is not influenced by the loss function $\Delta(y, z)$

Structured Perceptron Revisited

Replace inference with loss-augmented inference

Algorithm:

Initialize $w = \mathbf{0}$

For $t = 1 \ldots T$

1. receive example pair x_t, y_t
2. predict $\hat{y}_t = \text{argmax}_{y \in \mathcal{Y}} [f(x_t, y) + \Delta(y_t, y)]$
3. $w \leftarrow w + \eta_t(\phi(x_t, y_t) - \phi(x_t, \hat{y}_t))$

Min-Max Formulation

(Taskar et al., ICML 2005)

Recall brute force enumeration

$$\min_{w,\xi} \quad \lambda\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i$$

$$\text{s.t. :} \quad \langle w, \phi(x_i, y_i)\rangle - \langle w, \phi(x_i, z)\rangle \geq \Delta(y_i, z) - \xi_i, \quad \forall z, \forall i$$

$$\xi_i \geq 0, \quad \forall i \in [\![m]\!]$$

Min-Max Formulation

(Taskar et al., ICML 2005)

$$\min_{w,\xi} \quad \lambda\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i$$

$$\text{s.t. :} \quad \langle w, \phi(x_i, y_i)\rangle + \xi_i \geq \max_{z\in\mathcal{Y}}[\langle w, \phi(x_i, z) + \Delta(y_i, z)], \quad \forall i \in [\![m]\!]$$

$$\xi_i \geq 0, \quad \forall i \in [\![m]\!]$$

Key steps:

- Plug-in LP inference
- Use LP duality to replace max with min
- Rewrite the OPT as a *concise* QP with polynomial number of variables and constraints (depends on the output structure)

Logistic Regression $\rightarrow$ CRF

Logistic Regression

Probabilistic (binary) classifier

Likelihood function:

$$p(y \mid x, w) = \frac{\exp(y \langle \phi(x), w \rangle)}{\exp(y \langle \phi(x), w \rangle) + \exp(-y \langle \phi(x), w \rangle)}$$

Estimate parameters w by minimizing negative log-likelihood

Exponential Family

Family of probability distributions

$$p(x; w) = \exp(\langle \phi(x), w \rangle - \ln Z(w))$$

$\phi(x)$ - Sufficient statistics of x

$Z(w)$ - Partition function

$$Z(w) = \int_x \exp(\langle \phi(x), w \rangle) dx$$

Log-Partition Function

$$g(w) = \ln Z(w) = \ln \int_x \exp(\langle \phi(x), w \rangle) dx$$

$g(w)$ is a cumulant generator, i.e.,

$$\partial_w g(w) = \frac{\int \phi(x) \exp(\langle \phi(x), w \rangle) dx}{\int \exp(\langle \phi(x), w \rangle) dx} = \mathop{\mathbf{E}}_{x \sim p(x;w)}[\phi(x)]$$

$$\partial_w^2 g(w) = \mathbf{Cov}_{x \sim p(x;w)}[\phi(x)]$$

Note: $g(w)$ is convex!

Examples

Binomial

Multinomial

Gaussian

Laplace

Poisson

Dirichlet

...

(Conditional) Exponential Family

Family of conditional distributions

$$p(y \mid x, w) = \exp(\langle \phi(x, y), w \rangle - \ln Z(w \mid x))$$

$\phi(x, y)$ - Joint sufficient statistics of x and y

$Z(w \mid x)$ - Partition function

$$Z(w \mid x) = \int_{y \in \mathcal{Y}} \exp(\langle \phi(x, y), w \rangle)$$

MAP Estimation

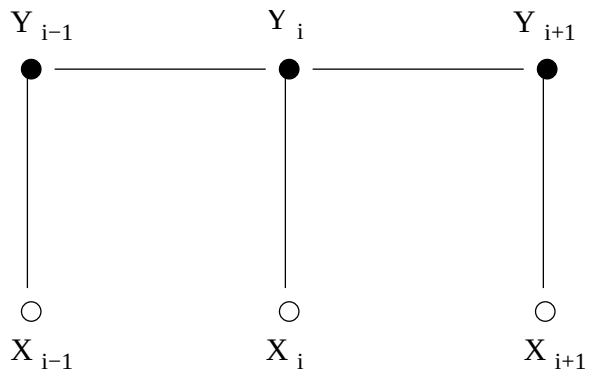
$$p(w \mid Y, X) \approx p(w, Y \mid X) = p(Y \mid X, w)p(w)$$

Point estimate with a Gaussian prior on w,
$p(w) = \exp(-\lambda \|w\|^2)$

$$\begin{aligned}\hat{w} &= \operatorname*{argmax}_w \left[\ln p(w, Y \mid X)\right] \\ &= \operatorname*{argmax}_w \left[\frac{1}{m}\sum_{i=1}^{m} p(y_i \mid x_i, w) - \lambda \|w\|^2\right]\end{aligned}$$

Note: Convex optimization problem

Linear-Chain Conditional Random Fields

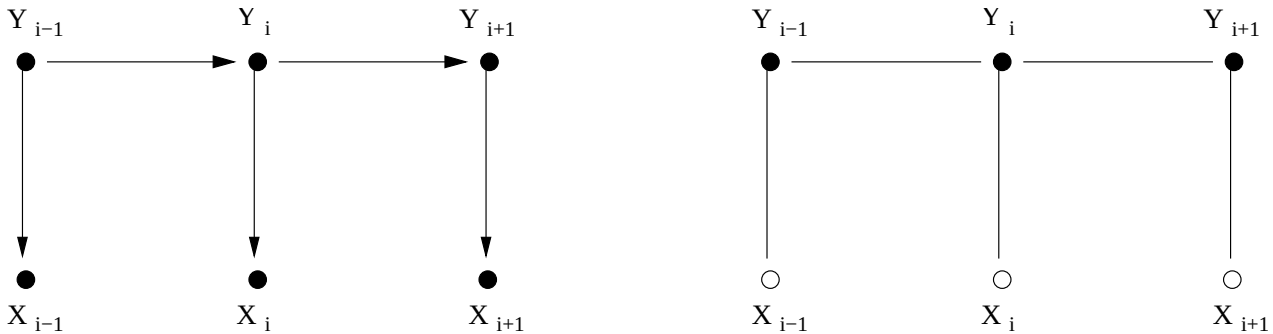


Cliques: $(y_t, x_t), (y_{t-1}, y_t)$, for all t

$$p(y \mid x, w) =$$
$$\exp\left(\sum_t [\langle \phi(y_t, x_t), w_{xy,t} \rangle + \langle \phi(y_{t-1}, y_t), w_{yy,t} \rangle] - \ln Z(w \mid x)\right)$$

- Efficient inference using dynamic programming
- MLE / MAP estimation of parameters

HMMs and CRFs



Hidden Markov Model: Maximize $p(x, y \mid w)$

Conditional Random Field: Maximize $p(y \mid x, w)$

Learning Reductions

(ICML 2009 tutorial)

Reductions: Transform complex learning problems into simpler, core problems

Desideratum: Good performance on the core problem should imply good performance on the complex problem

Examples

- Multi-class prediction $\rightarrow$ Binary classification
- Ranking $\rightarrow$ Classification
- Cost-sensitive classification $\rightarrow$ Binary classification
- Structured prediction $\rightarrow$ Binary classification (SEARN)

Structured $\rightarrow$ Binary (SEARN)

(Daumé et al., 2009)

Decomposable outputs $y = (y_1, \ldots, y_T)$

SEARN learns a policy π that maps tuples $(x, y_1, \ldots, y_{t-1})$ to y_T

Reduces structured prediction to cost-sensitive (multi-class) classification

Good performance on cost-sensitive classification implies good performance on the structured prediction problem

Note: No argmax problem!

Reduction to Cost-Sensitive Classification

Distribution over cost-sensitive examples $(\text{input}, c_1, \ldots, c_K)$

For every structured example (x, y)

1. sample t uniformly from $[\![T]\!]$
2. run policy π for $t-1$ steps to yield $(\hat{y}_1, \ldots, \hat{y}_{t-1})$
3. Input: $(x, \hat{y}_1, \ldots, \hat{y}_{t-1})$
4. Costs:

$$c_k = \mathop{\mathbf{E}}_{\hat{y}_{t+1},\ldots,\hat{y}_T \sim \pi} \ell(y, (\hat{y}_1, \ldots, \hat{y}_{t-1}, k, \hat{y}_{t-1}, \ldots, \hat{y}_T)), \forall k \in [\![K]\!]$$

Intermediate Summary

Loss Functions (binary $\rightarrow$ structured)

Discriminative Structured Prediction Algorithms

- Perceptron $\rightarrow$ Structured perceptron
- RLSR $\rightarrow$ KDE
- SVM $\rightarrow$ Struct-SVM, M3N
- LogReg $\rightarrow$ CRF
- Structured prediction $\rightarrow$ Cost-sensitive classification