Probabilistic Graphical Models in Computer Vision (IN2329)

Csaba Domokos

Summer Semester 2017

10. Sampling & Parameter learning

Agenda for today's lecture *

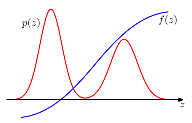
Today we are going to learn about

- **Sampling**
  We wish to draw samples in general from a distribution. Moreover, we aim to estimate *expectations*
  $$\mathbb{E}[f(Z)] = \sum_{\mathbf{z}} f(\mathbf{z}) p_Z(\mathbf{z}) \ .$$
- **Parameter learning**
  Consider an *energy function* for a *parameter vector* $\mathbf{w} = [w_1, w_2]^T$:
  $$E(\mathbf{y}; \mathbf{x}, \mathbf{w}) = w_1 \sum_{i \in \mathcal{V}} E_i(y_i; x_i) + w_2 \sum_{(i,j) \in \mathcal{E}} E_{ij}(y_i, y_j; x_i, x_j) \ .$$
  We aim to estimate *optimal parameter vector* $\mathbf{w}$ consisting of (positive) weighting factors (like $w_1, w_2 \in \mathbb{R}^+$) for $E(\mathbf{y}; \mathbf{x}, \mathbf{w})$.

Sampling

Monte Carlo

We wish to evaluate the **expectation**
$$\mathbb{E}[f(Z)] = \sum_{\mathbf{z}} f(\mathbf{z}) p_Z(\mathbf{z}) \ .$$



Source: C. Bishop. PRML, 2006.

Monte Carlo is the art of approximating an expectation by the sample mean of a given function f. The general idea behind *sampling* is to obtain a set of *i.i.d.* samples $\mathbf{z}^{(i)}$ drawn from $p_Z(\mathbf{z})$.

We define the **Monte Carlo estimator** as
$$\hat{f} = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{z}^{(i)}) \ .$$

The **(weak) law of large numbers** states that for any $\epsilon > 0$
$$\lim_{n \to \infty} P(|\hat{f} - \mathbb{E}[f]| \geqslant \epsilon) = 0 \ .$$

Monte Carlo

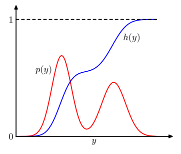
$$\mathbb{E}[\hat{f}] = \mathbb{E}\left[\frac{1}{n} \sum_{i=1}^{n} f(\mathbf{z}^{(i)})\right] = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}[f(\mathbf{z}^{(i)})] = \mathbb{E}[f(Z)] \ .$$

Note that the accuracy of the estimator $\hat{f}$ does not depend on the dimensionality of $\mathbf{z}$, but the number of samples n.

If we have a method to obtain samples $\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n)}\}$ from the distribution $p(\mathbf{y} \mid \mathbf{x})$, then we can form an estimator, that is
$$\mathbb{E}_{\mathbf{y} \sim p(\mathbf{y}|\mathbf{x})}[\varphi(\mathbf{x}, \mathbf{y})] \approx \frac{1}{n} \sum_{i=1}^{n} \varphi(\mathbf{x}, \mathbf{y}^{(i)}) \ .$$

Basic sampling

Let $h(y)$ be a **continuous** and **strictly monotonic** cumulative distribution function (cdf.) and Z be a uniformly distributed random variable on the interval $[0, 1]$. Then
$$Y = h^{-1}(Z)$$
is a *random variable* with cdf. $h(y)$, where $h^{-1}(y)$ is the inverse of $h(y)$.



Source: C. Bishop. PRML, 2006.

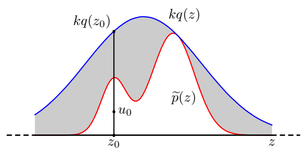
The cdf. of the uniformly distributed $Z \sim \mathcal{U}(0, 1)$ is given by
$$F_Z(z) \triangleq P(Y < z) = \begin{cases} 0, & \text{if } z \leqslant 0 \\ z, & \text{if } 0 < z \leqslant 1 \\ 1, & \text{if } 1 < z \ . \end{cases}$$

Therefore, the cdf. of Y is given by
$$F_Y(y) \triangleq P(Y < y) = P(h^{-1}(Z) < y) = P(Z < h(y)) = F_Z(h(y)) = h(y) \ .$$

Rejection sampling

Suppose we wish to sample from a distribution $p(z)$ that is a relatively complex distribution, therefore sampling directly from $p(z)$ is *difficult*.

Furthermore assume that we are able to evaluate $p(z)$ for any given value of z, up to a normalizing constant Z. That is
$$p(z) = \frac{1}{Z} \tilde{p}(z) \ ,$$
where $\tilde{p}(z)$ *can readily be evaluated*, but Z is unknown.



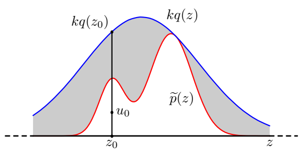
Source: C. Bishop. PRML, 2006.

We need for a simpler distribution $q(z)$, called a **proposal distribution**, from which we *can readily draw samples*. Moreover, let k be a constant such that
$$kq(z) \geqslant \tilde{p}(z) \quad \text{for all values of } z \ .$$

Rejection sampling

1. Generate a sample z_0 from the distribution $q(z)$.
2. Generate a sample $u_0 \sim \mathcal{U}(0, kq(z_0))$.

This pair of random samples has uniform distribution under the graph of the function $kq(z)$.



Source: C. Bishop. PRML, 2006.

If $u_0 > \tilde{p}(z_0)$ then the sample is *rejected*, otherwise u_0 is *retained*. Note that the remaining pairs follow uniform distribution under the curve of $\tilde{p}(z)$. Hence the corresponding z values are distributed according to $p(z)$.

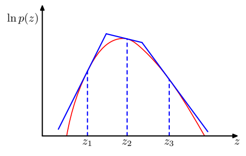
The values of z are generated from $q(z)$, and these samples are accepted with probability $\tilde{p}(z)/(kq(z))$, therefore

$$p(\text{'z is accepted'}) = \int \frac{\tilde{p}(z)}{kq(z)} q(z) \mathrm{d}z = \frac{1}{k} \int \tilde{p}(z) \mathrm{d}z = \frac{\int \tilde{p}(z) \mathrm{d}z}{\int kq(z) \mathrm{d}z} = \frac{Z}{k} .$$

Adaptive rejection sampling *

In the case of *log concave distributions*, an **envelope function** can be constructed using the tangent lines computed at a set of grid points.

A sample value is drawn from the *envelope function* considering as the scaled proposal distribution $kq(z)$.



Source: C. Bishop. PRML, 2006.

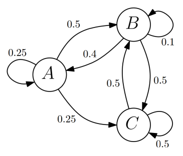
If a sample point is rejected, it is added to the set of grid points and used to refine the envelope distribution.

Markov chain *

Given a finite set $\mathcal{Y}$ and a matrix $\mathbf{T} \in \mathbb{R}^{\mathcal{Y} \times \mathcal{Y}}$, then a series of random variables $Y_1, Y_2, \ldots$ taking values from $\mathcal{Y}$ is called a **(homogeneous) Markov chain** with **transition matrix** $\mathbf{T}$, if

$$\begin{aligned} &P(Y_{t+1} = y^{(t+1)} \mid Y_1 = y^{(1)}, Y_2 = y^{(2)}, \ldots Y_t = y^{(t)}) \\ &= P(Y_{t+1} = y^{(t+1)} \mid Y_t = y^{(t)}) \\ &= \mathbf{T}_{y^{(t)}, y^{(t+1)}} . \end{aligned}$$

Example: Let us consider a *Markov chain* with $\mathbf{T} \in \mathbb{R}^{\mathcal{Y} \times \mathcal{Y}}$, where $\mathcal{Y} = \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$.



$\mathbf{T}$	A	B	C
A	0.25	0.5	0.5
B	0.4	0.1	0.5
C	0	0.5	0.5

Invariant distribution *

Given the initial probabilities $p(y^{(0)})$, this determines the behavior of the chain at all times. By making use of $\mathbf{T}$ one can find $P(Y_{t+1} = y^{(t+1)})$ as follows:

$$p(y^{(t+1)}) = \sum_{y^{(t)}} p(y^{(t+1)}, y^{(t)}) = \sum_{y^{(t)}} p(y^{(t+1)} \mid y^{(t)}) p(y^{(t)}) = \sum_{y^{(t)}} \mathbf{T}_{y^{(t)}, y^{(t+1)}} p(y^{(t)}) .$$

The distribution $p^*(y)$ is called **invariant** if

$$p^*(y) = \sum_{y'} \mathbf{T}_{y',y} p^*(y') .$$

The so-called **detailed balance**:

$$p^*(y) \mathbf{T}_{y,y'} = p^*(y') \mathbf{T}_{y',y} ,$$

provides a **sufficient** condition for a distribution to be *invariant*, since

$$\sum_{y'} \mathbf{T}_{y',y} p^*(y') = \sum_{y'} p^*(y) \mathbf{T}_{y,y'} = p^*(y) \sum_{y'} \mathbf{T}_{y,y'} = p^*(y) \sum_{y'} p(y' \mid y) = p^*(y) .$$

Ergodic Markov chain *

If $p(y^{(t)})$ converges to an *invariant distribution* as $t \to \infty$, then the *Markov chain* is called **ergodic**.

An *ergodic Markov chain* can have only one *invariant distribution*, which is referred to as its **equilibrium distribution**.

The next theorem answers the question of when a *Markov chain* is *ergodic*.

Theorem 1. *If a homogeneous Markov chain on a finite state space with transition probabilities* $\mathbf{T}_{y,y'}$ *has* p^* *as an invariant distribution and*

$$\min_{y} \min_{y' : p^*(y') > 0} \frac{\mathbf{T}_{y,y'}}{p^*(y')} > 0 ,$$

then the Markov chain is ergodic, i.e., regardless the initial probabilities $p(y^{(0)})$

$$\lim_{t \to \infty} p(y^{(t)}) = p^*(y) .$$

Markov Chain Monte Carlo (MCMC)

Let us consider *rejection sampling*, where the *proposal distribution* $q(y' \mid y)$ is a conditional distribution such that the next sample y' depends only on the current sample value y (i.e. it is a Markov chain).

The probability of the acceptance of a new sample, therefore, can be written as

$$p(y' \mid y) = q(y' \mid y) A(y', y) .$$

If the candidate sample is accepted, then $\mathbf{y}^{(t+1)} = \mathbf{y}'$, otherwise the candidate point $\mathbf{y}'$ is discarded and $\mathbf{y}^{(t+1)}$ is set to $\mathbf{y}^{(t)}$ and another candidate sample is drawn from the distribution $q(\mathbf{y} \mid \mathbf{y}^{(t+1)})$.

Note that in *rejection sampling*, rejected samples are simply discarded.

Metropolis-Hastings algorithm *

Let us assume a *proposal distribution* q (that is not necessarily symmetric, i.e. $q(y' \mid y) \neq q(y \mid y')$) and let

$$A(y', y) = \min\left(1, \frac{p(y') q(y \mid y')}{p(y) q(y' \mid y)}\right) .$$

The **detailed balance** is satisfied, since

$$\begin{aligned} p(y)\mathbf{T}_{y,y'} &= p(y) q(y' \mid y) A(y', y) = p(y) q(y' \mid y) \min\left(1, \frac{p(y') q(y \mid y')}{p(y) q(y' \mid y)}\right) \\ &= p(y') q(y \mid y') \min\left(1, \frac{p(y) q(y' \mid y)}{p(y') q(y \mid y')}\right) = p(y') q(y \mid y') A(y, y') = p(y') \mathbf{T}_{y',y} . \end{aligned}$$

A sample $\mathbf{y}'$ is accepted with probability

$$A(\mathbf{y}', \mathbf{y}^{(t-1)}) = \min\left(1, \frac{\tilde{p}(\mathbf{y}' \mid \mathbf{x})\, q(\mathbf{y}^{(t-1)} \mid \mathbf{y}')}{\tilde{p}(\mathbf{y}^{(t-1)} \mid \mathbf{x})\, q(\mathbf{y}' \mid \mathbf{y}^{(t-1)})}\right) .$$

Metropolis-Hastings algorithm *

Input: $\tilde{p}(\mathbf{y} \mid \mathbf{x}) \propto p(\mathbf{y} \mid \mathbf{x})$, unnormalized target distribution; $q(\mathbf{y} \mid \mathbf{y}^{(t-1)})$, proposal distribution; T, the number of generated samples

Output: $\{\mathbf{y}^{(t)}\}_{t=1}^{T}$, sequence of samples with approximately $\mathbf{y}^{(t)} \sim p(\mathbf{y} \mid \mathbf{x})$

```
1: $\mathbf{y}^{(0)} \leftarrow$ arbitrary in $\mathcal{Y}$
2: for $t = 1, \ldots, T$ do
3:     $\mathbf{y}' \sim q(\mathbf{y} \mid \mathbf{y}^{(t-1)})$        ▻ Generate a candidate
4:     $a \leftarrow \min\left(1, \frac{\tilde{p}(\mathbf{y}'|\mathbf{x})\ q(\mathbf{y}^{(t-1)}|\mathbf{y}')}{\tilde{p}(\mathbf{y}^{(t-1)}|\mathbf{x})\ q(\mathbf{y}'|\mathbf{y}^{(t-1)})}\right)$        ▻ Compute acceptance prob.
5:     $\mathbf{y}^{(t)} \leftarrow \begin{cases} \mathbf{y}' & \text{with probability } a \text{ (accept)} \\ \mathbf{y}^{(t-1)} & \text{otherwise (reject)} \end{cases}$        ▻ Update
6:     output $\mathbf{y}^{(t)}$
7: end for
```

Gibbs sampling

Geman and Geman proposed a simple *MCMC algorithm* which can be seen as a special case of *Metropolis-Hasting algorithm*.

As usual y_i will denote the i^{th} component of $\mathbf{y}$. Moreover, we will use the notation $\mathbf{y}_{\setminus i}$ for $\mathbf{y}_{\mathcal{V}\setminus\{i\}}$, i.e. y_i is omitted.

Each step of the *Gibbs sampling* procedure involves replacing the value of one of the variables Y_i by a value drawn from the distribution of that variable conditioned on the values of the remaining variables, that is

$$y_i^{(t+1)} \leftarrow y_i' \sim p(y_i \mid \mathbf{y}_{\setminus i}^{(t)}, \mathbf{x}) .$$

This requires only the unnormalized distribution $\tilde{p}$ and the normalization over a single variable:

$$p(y_i \mid \mathbf{y}_{\setminus i}^{(t)}, \mathbf{x}) = \frac{p(y_i, \mathbf{y}_{\setminus i}^{(t)} \mid \mathbf{x})}{p(\mathbf{y}_{\setminus i}^{(t)} \mid \mathbf{x})} = \frac{p(y_i, \mathbf{y}_{\setminus i}^{(t)} \mid \mathbf{x})}{\sum_{y_i \in \mathcal{Y}_i} p(y_i, \mathbf{y}_{\setminus i}^{(t)} \mid \mathbf{x})} = \frac{\tilde{p}(y_i, \mathbf{y}_{\setminus i}^{(t)} \mid \mathbf{x})}{\sum_{y_i \in \mathcal{Y}_i} \tilde{p}(y_i, \mathbf{y}_{\setminus i}^{(t)} \mid \mathbf{x})}$$

Gibbs sampling

$$\begin{aligned} p(y_i \mid \mathbf{y}_{\setminus i}^{(t)}, \mathbf{x}) &= \frac{\tilde{p}(y_i, \mathbf{y}_{\setminus i}^{(t)} \mid \mathbf{x})}{\sum_{y_i \in \mathcal{Y}_i} \tilde{p}(y_i, \mathbf{y}_{\setminus i}^{(t)} \mid \mathbf{x})} \\ &= \frac{\prod_{F \in M(i)} \exp(-E_F(y_i, \mathbf{y}_{N(F)\setminus\{i\}}^{(t)}; \mathbf{x}_F))}{\sum_{y_i \in \mathcal{Y}_i} \prod_{F \in M(i)} \exp(-E_F(y_i, \mathbf{y}_{N(F)\setminus\{i\}}^{(t)}; \mathbf{x}_F))} . \end{aligned}$$

The basic idea is that while sampling from $p(\mathbf{y} \mid \mathbf{x})$ is hard, sampling from the conditional distributions $p(y_i \mid \mathbf{y}_{\setminus i}, \mathbf{x})$ can be performed efficiently.

Gibbs sampling as the special case of the Metropolis-Hastings algorithm *

Consider a *Metropolis-Hastings sampling* step involving the variable y_i in which the remaining variables $\mathbf{y}_{\setminus i}$ remain fixed.

The transition probability from $\mathbf{y}^{(t-1)}$ to $\mathbf{y}'$ is given by

$$q_i(\mathbf{y}' \mid \mathbf{y}^{(t-1)}) = p(y_i' \mid \mathbf{y}_{\setminus i}, \mathbf{x}) .$$

Note that $\mathbf{y}_{\setminus i}' = \mathbf{y}_{\setminus i}^{(t-1)}$ because these components are unchanged by the sampling step.

One can see that each proposal is then always accepted, i.e.

$$\begin{aligned} A_i(\mathbf{y}', \mathbf{y}^{(t-1)}) &= \frac{p(\mathbf{y}' \mid \mathbf{x})\, q_i(\mathbf{y}^{(t-1)} \mid \mathbf{y}')}{p(\mathbf{y}^{(t-1)} \mid \mathbf{x})\, q_i(\mathbf{y}' \mid \mathbf{y}^{(t-1)})} \\ &= \frac{p(y_i' \mid \mathbf{y}_{\setminus i}', \mathbf{x})\; p(\mathbf{y}_{\setminus i}' \mid \mathbf{x})\; p(y_i^{(t-1)} \mid \mathbf{y}_{\setminus i}', \mathbf{x})}{p(y_i^{(t-1)} \mid \mathbf{y}_{\setminus i}^{(t-1)}, \mathbf{x})\; p(\mathbf{y}_{\setminus i}^{(t-1)} \mid \mathbf{x})\; p(y_i' \mid \mathbf{y}_{\setminus i}^{(t-1)}, \mathbf{x})} = 1 . \end{aligned}$$

Gibbs sampler *

```
Input: $\tilde{p}(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) \propto p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$, unnormalized target distribution; $T$, the number
    of generated samples
Output: $\{\mathbf{y}^{(t)}\}_{t=1}^{T}$, sequence of samples with approximately $\mathbf{y}^{(t)} \sim p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$
1: $\mathbf{y}^{(0)} \leftarrow$ arbitrary in $\mathcal{Y}$
2: for $t = 1, \ldots, T$ do
3:     $\mathbf{y}^{(t)} \leftarrow \mathbf{y}^{(t-1)}$
4:     for all $i \in \mathcal{V}$ do
5:         Sample $y_i^{(t)} \sim p(y_i \mid \mathbf{y}_{\setminus i}^{(t)}, \mathbf{x}) = \frac{\tilde{p}(y_i, \mathbf{y}_{\setminus i}^{(t)} | \mathbf{x})}{\sum_{y_i \in \mathcal{Y}_i} \tilde{p}(y_i, \mathbf{y}_{\setminus i}^{(t)} | \mathbf{x})}$        ▷ Sweep
6:     end for
7:     output $\mathbf{y}^{(t)}$
8: end for
```

Summary

We wish to obtain samples $\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n)}\}$ from the distribution $p(\mathbf{y} \mid \mathbf{x})$, in order to form an estimator

$$\mathbb{E}_{\mathbf{y} \sim p(\mathbf{y}|\mathbf{x})}[\varphi(\mathbf{x}, \mathbf{y})] \approx \frac{1}{n} \sum_{i=1}^{n} \varphi(\mathbf{x}, \mathbf{y}^{(i)}) .$$

MCMC is a method of *rejection sampling*, where the *proposal distribution* is defined as a *Markov chain*.

Gibbs sampling is a special case of the *Metropolis-Hastings algorithm* (i.e. *MCMC*), where each step involves replacing the value of one of the variables by a value drawn from the distribution of that variable conditioned on the values of the remaining variables via *basic sampling*.

Parameter learning

Parameterization

Let us consider the following example for an energy function:

$$E(\mathbf{y}; \mathbf{x}) = \sum_{i \in \mathcal{V}} E_i(y_i; \mathbf{x}_i) + \sum_{(i,j) \in \mathcal{E}} E_{ij}(y_i, y_j) .$$

Instead, one may want to apply weighting factors $w_1, w_2 \in \mathbb{R}_+$:

$$E(\mathbf{y}; \mathbf{x}, \mathbf{w}) = w_1 \sum_{i \in \mathcal{V}} E_i(y_i; x_i) + w_2 \sum_{(i,j) \in \mathcal{E}} E_{ij}(y_i, y_j) = \left\langle \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \begin{bmatrix} \sum_{i \in \mathcal{V}} E_i(y_i; x_i)) \\ \sum_{(i,j) \in \mathcal{E}} E_{ij}(y_i, y_j) \end{bmatrix} \right\rangle .$$

In a more general form, one can write the *energy functions* as a **linear combination** for a **parameter vector** $\mathbf{w} \in \mathbb{R}^D$, $D = |\mathcal{F}|$:

$$E(\mathbf{y}; \mathbf{x}, \mathbf{w}) = \left\langle \begin{bmatrix} w_1 \\ \vdots \\ w_D \end{bmatrix}, \underbrace{\begin{bmatrix} E_{F_1}(\mathbf{y}_{F_1}; \mathbf{x}_{F_1})) \\ \vdots \\ E_{F_D}(\mathbf{y}_{F_D}; \mathbf{x}_{F_D})) \end{bmatrix}}_{\varphi(\mathbf{x}, \mathbf{y})} \right\rangle = \langle \mathbf{w}, \varphi(\mathbf{x}, \mathbf{y}) \rangle .$$

Parameter learning

Learning graphical models (from training data) is a way to find among a large class of possible models a single one that is *best* in some sense for the task at hand.

We assume a fixed underlying graphical model with **parameterized conditional probability distribution**

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \exp(-E(\mathbf{y}; \mathbf{x}, \mathbf{w})) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \exp(-\langle \mathbf{w}, \varphi(\mathbf{x}, \mathbf{y}) \rangle) ,$$

where $Z(\mathbf{x}, \mathbf{w}) = \sum_{\mathbf{y} \in \mathcal{Y}} \exp(-\langle \mathbf{w}, \varphi(\mathbf{x}, \mathbf{y}) \rangle)$. The only unknown quantity is the *parameter vector* $\mathbf{w}$, on which the energy $E(\mathbf{y}; \mathbf{x}, \mathbf{w})$ depends **linearly**.

In principle each part of a graphical model (i.e. random variables, factors and parameters) can be learned. However we assume that the model structure and parameterization are specified manually, and learning amounts to finding a vector of real-valued parameters.

Probabilistic parameter learning

Let $d(\mathbf{y} \mid \mathbf{x})$ be the (*unknown*) conditional distribution of labels for a problem to be solved. For a parameterized conditional distribution $p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$ with parameters $\mathbf{w} \in \mathbb{R}^D$, **probabilistic parameter learning** is the task of finding a point estimate of the parameter $\mathbf{w}^*$ that makes $p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}^*)$ *closest* to $d(\mathbf{y} \mid \mathbf{x})$.

Probabilistic parameter learning

We aim at identifying a *weight vector* $\mathbf{w}^*$ that makes $p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$ as close to the **true conditional label distribution** $d(\mathbf{y} \mid \mathbf{x})$ as possible. The label distribution itself is unknown to us, but we have an *i.i.d.* sample set $\mathcal{D} = \{(\mathbf{x}^n, \mathbf{y}^n)\}_{n=1,\dots,N}$ from $d(\mathbf{x}, \mathbf{y})$ that we can use for learning.

We now define what we mean by "closeness" between conditional distributions $p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$ and $d(\mathbf{x}, \mathbf{y})$ for any $\mathbf{x} \in \mathcal{X}$. We measure the dissimilarity by making use of **Kullback-Leibler (KL) divergence**:

$$\mathsf{KL}(d(\mathbf{y} \mid \mathbf{x}) \| p(\mathbf{y} \mid \mathbf{x})) = \sum_{\mathbf{y} \in \mathcal{Y}} d(\mathbf{y} \mid \mathbf{x}) \log \frac{d(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})} .$$

From this we obtain a **total measure** of how much p differs from d by their **expected dissimilarity** over all $\mathbf{x} \in \mathcal{X}$:

$$\mathsf{KL}_{\mathsf{tot}}(d \| p) \triangleq \mathbb{E}[\mathsf{KL}(d(\mathbf{y} \mid \mathbf{X}) \| p(\mathbf{y} \mid \mathbf{X}))] = \sum_{\mathbf{x} \in \mathcal{X}} d(\mathbf{x}) \sum_{\mathbf{y} \in \mathcal{Y}} d(\mathbf{y} \mid \mathbf{x}) \log \frac{d(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})} .$$

Probabilistic parameter learning

We choose the *parameter* $\mathbf{w}^*$ that minimizes *expected dissimilarity*, i.e.

$$\begin{aligned}
\mathbf{w}^* &\in \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \mathsf{KL}_{\mathsf{tot}}(d \| p) = \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \sum_{\mathbf{x} \in \mathcal{X}} d(\mathbf{x}) \sum_{\mathbf{y} \in \mathcal{Y}} d(\mathbf{y} \mid \mathbf{x}) \log \frac{d(\mathbf{y} \mid \mathbf{x})}{p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})} \\
&= \operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} \sum_{\mathbf{x} \in \mathcal{X}} \sum_{\mathbf{y} \in \mathcal{Y}} d(\mathbf{y} \mid \mathbf{x}) d(\mathbf{x}) \log p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) \\
&= \operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim d(\mathbf{x}, \mathbf{y})} [\log p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})] .
\end{aligned}$$

Unfortunately, we cannot compute this expression directly, because $d(\mathbf{x}, \mathbf{y})$ is unknown to us. However, we can approximate it using the sample set $\mathcal{D}$.

$$\begin{aligned}
&\approx \operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} \sum_{(\mathbf{x}^n, \mathbf{y}^n) \in \mathcal{D}} \log p(\mathbf{y}^n \mid \mathbf{x}^n, \mathbf{w}) = \operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} \sum_{n=1}^{N} \log \frac{\exp(-\langle \mathbf{w}, \varphi(\mathbf{x}^n, \mathbf{y}^n) \rangle)}{Z(\mathbf{x}^n, \mathbf{w})} \\
&= \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \sum_{n=1}^{N} \langle \mathbf{w}, \varphi(\mathbf{x}^n, \mathbf{y}^n) \rangle + \sum_{n=1}^{N} \log Z(\mathbf{x}^n, \mathbf{w}) .
\end{aligned}$$

Maximum conditional likelihood *

By making use of *i.i.d.* assumption of the sample set $\mathcal{D}$, we can write that

$$\begin{aligned}
&\operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim d(\mathbf{x}, \mathbf{y})} [\log p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})] \\
\approx\ &\operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} \sum_{(\mathbf{x}^n, \mathbf{y}^n) \in \mathcal{D}} \log p(\mathbf{y}^n \mid \mathbf{x}^n, \mathbf{w}) \\
=\ &\operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} \log \prod_{n=1}^{N} p(\mathbf{y}^n \mid \mathbf{x}^n, \mathbf{w}) \\
=\ &\operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} \prod_{n=1}^{N} p(\mathbf{y}^n \mid \mathbf{x}^n, \mathbf{w}) \\
=\ &\operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} p(\mathbf{y}^1, \dots, \mathbf{y}^N \mid \mathbf{x}^1, \dots, \mathbf{x}^N, \mathbf{w}) ,
\end{aligned}$$

from which the name **maximum conditional likelihood** (MCL) stems.

Prior distribution on $\mathbf{w}$

When the number of training instances is *small* compared to the number of degrees of freedom (D) in $\mathbf{w}$, then the approximation

$$\operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim d(\mathbf{x}, \mathbf{y})} [\log p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})] \approx \operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} \sum_{(\mathbf{x}^n, \mathbf{y}^n) \in \mathcal{D}} \log p(\mathbf{y}^n \mid \mathbf{x}^n, \mathbf{w})$$

becomes *unreliable*, and $\mathbf{w}^*$ will vary strongly with respect to the training set $\mathcal{D}$, which means MCL training is prone to **overfitting**.

To overcome this limitation, we treat $\mathbf{w}$ not as a deterministic parameter but as yet another random variable. For any prior distribution $p(\mathbf{w})$ over the space of weight vectors, the posterior probability of $\mathbf{w}$ for given observations $\mathcal{D} = \{(\mathbf{x}^n, \mathbf{y}^n)\}_{n=1,\dots,N}$ is given by (see Exercise):

$$p(\mathbf{w} \mid \mathcal{D}) = p(\mathbf{w}) \prod_{n=1}^{N} \frac{p(\mathbf{y}^n \mid \mathbf{x}^n, \mathbf{w})}{p(\mathbf{y}^n \mid \mathbf{x}^n)} .$$

Negative conditional log-likelihood *

Assume a prior distribution of $p(\mathbf{w})$, then we can get

$$\begin{aligned}
\mathbf{w}^* &\in \operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^D} p(\mathbf{w} \mid \mathcal{D}) \\
&= \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \{-\log p(\mathbf{w} \mid \mathcal{D})\} \\
&= \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \left\{ -\log \left(p(\mathbf{w}) \prod_{n=1}^{N} \frac{p(\mathbf{y}^n \mid \mathbf{x}^n, \mathbf{w})}{p(\mathbf{y}^n \mid \mathbf{x}^n)} \right) \right\} \\
&= \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \left\{ -\log p(\mathbf{w}) - \sum_{n=1}^{N} \log p(\mathbf{y}^n \mid \mathbf{x}^n, \mathbf{w}) + \sum_{n=1}^{N} \log p(\mathbf{y}^n \mid \mathbf{x}^n) \right\} \\
&= \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \left\{ -\log p(\mathbf{w}) - \sum_{n=1}^{N} \log p(\mathbf{y}^n \mid \mathbf{x}^n, \mathbf{w}) \right\} .
\end{aligned}$$

Regularized conditional log-likelihood *

$$\mathbf{w}^* \in \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \left\{ -\log p(\mathbf{w}) - \sum_{n=1}^{N} \log p(\mathbf{y}^n \mid \mathbf{x}^n, \mathbf{w}) \right\}$$

Assuming a zero-mean Gaussian prior $p(\mathbf{w}) \propto \exp\left(-\frac{\|\mathbf{w}\|^2}{2\sigma^2}\right)$, then we get

$$\begin{aligned}
\mathbf{w}^* &\in \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \left\{ \frac{\|\mathbf{w}\|^2}{2\sigma^2} - \sum_{n=1}^{N} \log p(\mathbf{y}^n \mid \mathbf{x}^n, \mathbf{w}) \right\} \\
&= \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \left\{ \lambda \|\mathbf{w}\|^2 + \sum_{n=1}^{N} \langle \mathbf{w}, \varphi(\mathbf{x}^n, \mathbf{y}^n) \rangle + \sum_{n=1}^{N} \log Z(\mathbf{x}^n, \mathbf{w}) \right\} ,
\end{aligned}$$

where $\lambda = \frac{1}{2\sigma^2}$.

The parameter λ is generally considered as a free hyper-parameter that determines the regularization strength. Unregularized situation can be seen as the limit case for $\lambda \to 0$.

Regularized maximum conditional likelihood training

Let $p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \exp(-\langle \mathbf{w}, \varphi(\mathbf{x}, \mathbf{y}) \rangle)$ be a **probability distribution parameterized by** $\mathbf{w} \in \mathbb{R}^D$, and let $\mathcal{D} = \{(\mathbf{x}^n, \mathbf{y}^n)\}_{n=1,\dots,N}$ be a set of **training examples**. For any $\lambda > 0$, **regularized maximum conditional likelihood** (RMCL) training chooses the parameter as

$$\mathbf{w} \in \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \lambda \|\mathbf{w}\|^2 + \sum_{n=1}^{N} \langle \mathbf{w}, \varphi(\mathbf{x}^n, \mathbf{y}^n) \rangle + \sum_{n=1}^{N} \log Z(\mathbf{x}^n, \mathbf{w}) .$$

For $\lambda = 0$ the simplified rule

$$\mathbf{w} \in \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^D} \sum_{n=1}^{N} \langle \mathbf{w}, \varphi(\mathbf{x}^n, \mathbf{y}^n) \rangle + \sum_{n=1}^{N} \log Z(\mathbf{x}^n, \mathbf{w})$$

results in **maximum conditional likelihood** (MCL) training.

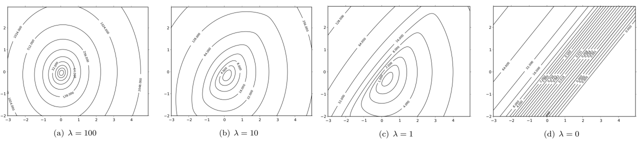
Negative conditional log-likelihood: Toy example *

Consider a simple CRF model with a single variable, where $\mathcal{Y} = \{-1, +1\}$. We define the energy function as

$$E(x, y, \mathbf{w}) = w_1\varphi_1(x, y) + w_2\varphi_2(x, y) \ .$$

Assuming a training set $\mathcal{D} = \{(-10, +1), (-4, +1), (6, -1), (5, -1)\}$ with

$$\varphi_1(x, y) = \begin{cases} 0, & \text{if } y = -1 \\ x, & \text{if } y = +1 \end{cases} \quad \text{and} \quad \varphi_2(x, y) = \begin{cases} x, & \text{if } y = -1 \\ 0, & \text{if } y = +1 \end{cases} .$$



Source: Nowozin and Lampert. Structured prediction and learning in computer vision, 2010.

$$L(\mathbf{w}) = \lambda\|\mathbf{w}\|^2 + \sum_{n=1}^{N}\langle\mathbf{w}, \varphi(x^n, y^n)\rangle + \sum_{n=1}^{N}\log Z(x^n, \mathbf{w}).$$

Steepest descent minimization *

Let us consider the *negative conditional log-likelihood* function

$$L(\mathbf{w}) = \lambda\|\mathbf{w}\|^2 + \sum_{n=1}^{N}\langle\mathbf{w}, \varphi(\mathbf{x}^n, \mathbf{y}^n)\rangle + \sum_{n=1}^{N}\log Z(\mathbf{x}^n, \mathbf{w}) \ .$$

Obviously, L is C^∞-differentiable, i.e. smooth function, on all $\mathbb{R}^D$.

```
1: $\mathbf{w}_{\text{cur}} \leftarrow 0$
2: repeat
3:     $\mathbf{d} \leftarrow -\nabla_{\mathbf{w}} L(\mathbf{w}_{\text{cur}})$
4:     $\eta \leftarrow \operatorname{argmin}_{\eta>0} L(\mathbf{w}_{\text{cur}} + \eta\mathbf{d})$
5:     $\mathbf{w}_{\text{cur}} \leftarrow \mathbf{w}_{\text{cur}} + \eta\mathbf{d}$
6: until $\|\mathbf{d}\| < \epsilon$
7: return $\mathbf{w}_{\text{cur}}$
```

Gradient-based optimization

The gradient vector (cf. Analysis I/II) of $L(\mathbf{w})$ is given by

$$\begin{aligned}
\nabla_{\mathbf{w}} L(\mathbf{w}) =& \nabla_{\mathbf{w}}\left(\lambda\|\mathbf{w}\|^2 + \sum_{n=1}^{N}\langle\mathbf{w}, \varphi(\mathbf{x}^n, \mathbf{y}^n)\rangle + \sum_{n=1}^{N}\log Z(\mathbf{x}^n, \mathbf{w})\right) \\
=& 2\lambda\mathbf{w} + \sum_{n=1}^{N}\left(\varphi(\mathbf{x}^n, \mathbf{y}^n) + \sum_{\mathbf{y}\in\mathcal{Y}}\frac{\exp(-\langle\mathbf{w}, \varphi(\mathbf{x}^n, \mathbf{y})\rangle)}{\sum_{\mathbf{y}'\in\mathcal{Y}}\exp(-\langle\mathbf{w}, \varphi(\mathbf{x}^n, \mathbf{y}')\rangle)}(-\varphi(\mathbf{x}^n, \mathbf{y}))\right) \\
=& 2\lambda\mathbf{w} + \sum_{n=1}^{N}\left(\varphi(\mathbf{x}^n, \mathbf{y}^n) - \sum_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y} \mid \mathbf{x}^n, \mathbf{w})\varphi(\mathbf{x}^n, \mathbf{y})\right) \\
=& 2\lambda\mathbf{w} + \sum_{n=1}^{N}\left(\varphi(\mathbf{x}^n, \mathbf{y}^n) - \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|\mathbf{x}^n,\mathbf{w})}[\varphi(\mathbf{x}^n, \mathbf{y})]\right) \ .
\end{aligned}$$

Interpretation: we aim for *expectation matching*, that is

$$\varphi(\mathbf{x}^n, \mathbf{y}^n) = \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|\mathbf{x}^n,\mathbf{w})}[\varphi(\mathbf{x}^n, \mathbf{y})] \quad \text{for} \quad \mathbf{x}^1, \ldots, \mathbf{x}^n \ .$$

Hessian of $L(\mathbf{w})$ *

By differentiating of $\nabla_{\mathbf{w}} L(\mathbf{w})$, the Hessian matrix (cf. Analysis I/II) of $L(\mathbf{w})$ is given by (see Exercise):

$$\begin{aligned}
\Delta_{\mathbf{w}} L(\mathbf{w}) = 2\lambda\mathbf{I} + \sum_{n=1}^{N}\Big(& \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|\mathbf{x}^n,\mathbf{w})}[\varphi(\mathbf{x}^n, \mathbf{y})\varphi(\mathbf{x}^n, \mathbf{y})^T] \\
& - \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|\mathbf{x}^n,\mathbf{w})}[\varphi(\mathbf{x}^n, \mathbf{y})]\mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|\mathbf{x}^n,\mathbf{w})}[\varphi(\mathbf{x}^n, \mathbf{y})]^T\Big) \ .
\end{aligned}$$

Reminder: for any random vector $\mathbf{X}$ the covariance $\text{Cov}(\mathbf{X}, \mathbf{X})$ can be written as:

$$\text{Cov}(\mathbf{X}, \mathbf{X}) \overset{\Delta}{=} \mathbb{E}[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^T] = \mathbb{E}[\mathbf{X}\mathbf{X}^T] - \mathbb{E}[\mathbf{X}]\mathbb{E}[\mathbf{X}]^T \ .$$

Note that $\Delta_{\mathbf{w}} L(\mathbf{w})$ is a **covariance matrix**, hence it is *positive semi-definite*. Therefore, $L(\mathbf{w})$ is **convex**, which guarantees that every local minimum will also be a global one minimum of $L(\mathbf{w})$.

Gradient approximation via sampling

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = 2\lambda\mathbf{w} + \sum_{n=1}^{N}\left(\varphi(\mathbf{x}^n, \mathbf{y}^n) - \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|\mathbf{x}^n,\mathbf{w})}[\varphi(\mathbf{x}^n, \mathbf{y})]\right) \ .$$

In a naive way, the complexity of the gradient computation is $\mathcal{O}(K^{|\mathcal{V}|}ND)$, where

- N is the number of samples,
- D is the dimension of weight vector,
- $K = \max_{i\in\mathcal{V}}|\mathcal{Y}_i|$ is (maximal) number of possible labels of each output nodes.

The computationally demanding part in the gradient computation has the form of the *expectation* of $\varphi(\mathbf{x}, \mathbf{y})$ with respect to the distribution $p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$.

As we have seen *sampling methods* often offer a viable alternative, as they provide a universal tool for evaluating expectations over random variables.

Summary *

Probabilistic parameter learning aims at identifying a weight vector $\mathbf{w}^*$ that makes $p(\mathbf{y} \mid \mathbf{x}, \mathbf{w})$ close to the **true conditional label distribution** $d(\mathbf{y} \mid \mathbf{x})$ in terms of the *expected KL divergence*.

This is achieved by **regularized maximum conditional likelihood** training for $\lambda > 0$ as

$$\mathbf{w}^* \in \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^D} L(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^D} \lambda\|\mathbf{w}\|^2 + \sum_{n=1}^{N}\langle\mathbf{w}, \varphi(\mathbf{x}^n, \mathbf{y}^n)\rangle + \sum_{n=1}^{N}\log Z(\mathbf{x}^n, \mathbf{w}) \ .$$

In the **next lecture** we will learn about various numerical solutions to calculate the gradient

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = 2\lambda\mathbf{w} + \sum_{n=1}^{N}\left(\varphi(\mathbf{x}^n, \mathbf{y}^n) - \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|\mathbf{x}^n,\mathbf{w})}[\varphi(\mathbf{x}^n, \mathbf{y})]\right) \ .$$

Literature *

Sampling

1. Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006
2. Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November 1984
3. Radford M. Neal. Probabilistic inference using Markov Chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, September 1993

Parameter learning

4. Sebastian Nowozin and Christoph H. Lampert. Structured prediction and learning in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3–4), 2010