# Stochastic Dual Coordinate Ascent for Conditional Random Fields

Rémi LE PRIOL

September 8, 2017

## Introduction

We apply the Stochastic Dual Coordinate Ascent (SDCA) algorithm to two machine learning optimization problems. First we consider the multinomial logistic regression to classify an input x among K classes, where K is small. Second, we extend the algorithm to Conditional Random Fields (CRF). This structured prediction problem builds a map from an input x to a structured output y which lives in a space that is finite but that is exponentially big in the size of x. The raw application of SDCA is numerically intractable. We adapt the algorithm to the structure of the output by considering the marginals instead of the joint probability thus leading to an efficient algorithm.

## 1 Multiclass Logistic Regression

### 1.1 Linear Models

We consider the following classical supervised setting. We observe vector data $x_i$ in dimension d, associated to labels $y_i \in 1, .., K$, for $i \in 1, .., n$. We note $X$ the design matrix of size n*d whose lines are the $x_i$. We note Y the label matrix of size n*K whose line $Y_i$ is the one-hot encoding of the label $y_i$. Given a new vector $x$, we want to predict what is the corresponding label $y$, or even better, a probability distribution over the classes $1, .., K$.

The simpler models are linear classifiers : the probability for each class $y$ is proportional to $\exp(w_y^T x)$, where the vectors $w_y$ are the parameters of the model. In other word, we have one predictor per class, whose parameter is $w_y$. The trick is that we will train these predictors jointly. We write $W$ the weight matrix whose lines are the $w_y$. The closed formula for the conditional probability of y given x is thus :

$$p(y|x; W) = \frac{\exp(w_y^T x)}{\sum_{y=1}^{K} \exp(w_y^T x)} \tag{1}$$

With such models, we get polygonal classification boundaries, as shown in figure 1.1.

For the sake of comparison, this is the same model as a neural network with no hidden layer, the input layer of size d, the output layer of size K, a weight matrix W of size K*d, and an exponential non-linearity.

### 1.2 Maximum Likelihood

The multinomial logistic regression is an optimization problem that we write to fit the model to the training data. It is the maximum likelihood estimator of the parameter W, regularized with the $l^2$ loss. We note $\lambda$ the regularization parameter, and $\mathscr{P}$ the primal function to optimize.

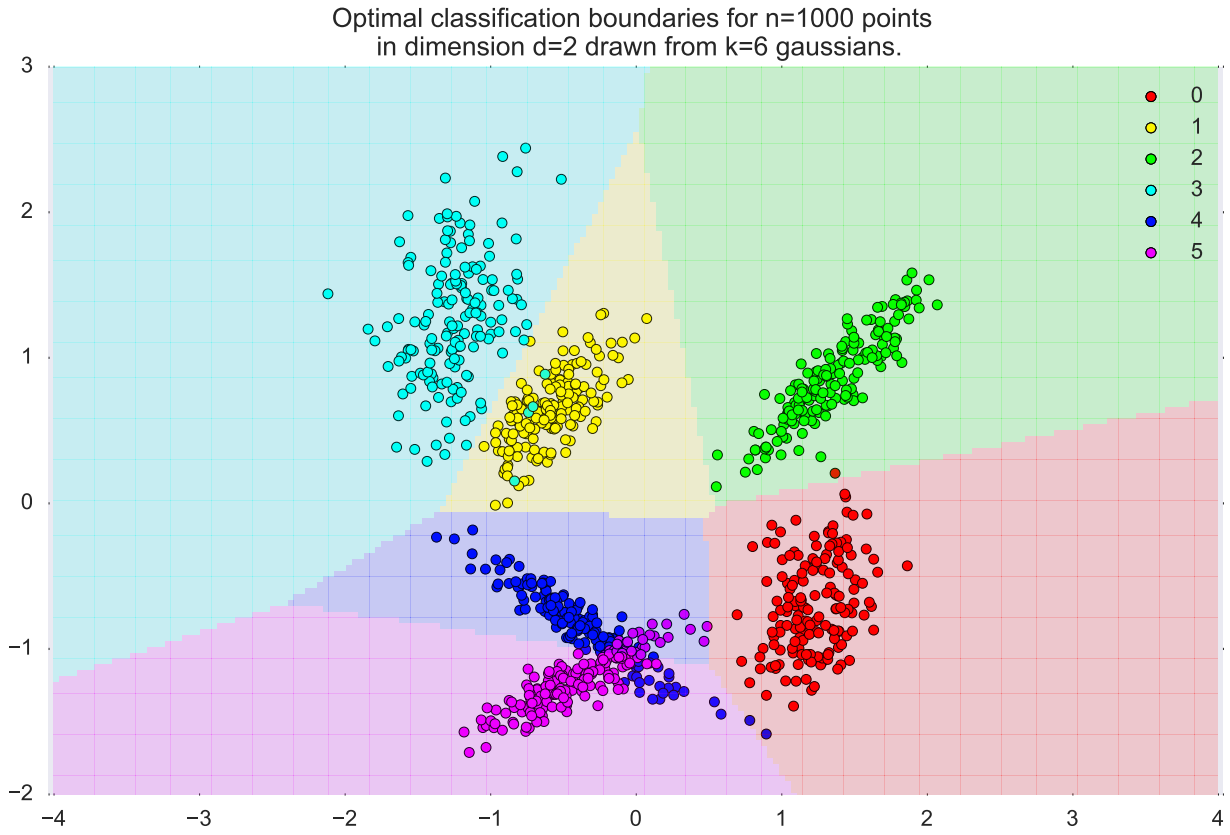


Figure 1: Optimal decision areas for the multinomial logistic regression.

$$\min_{W \in \mathbb{R}^{K*d}} \mathscr{P}(W) = \frac{\lambda}{2} \sum_{y=1}^{K} \|w_y\|_2^2 - \frac{1}{n} \sum_{i=1}^{n} \log p(y_i|x_i; W) \tag{2}$$

We further abbreviate $\|W\|^2 = \sum_{y=1}^{K} \|w_y\|_2^2\|$ considering the Frobenius norm of $W$. We also expand the expression of the conditional probability to get the classical formulation of the logistic regression problem. We note $\mathrm{Tr}(Z)$ the trace of the matrix Z.

$$\min_{W \in \mathbb{R}^{K*d}} \frac{\lambda}{2} \|W\|^2 + \frac{1}{n} \sum_{i=1}^{n} \log \left( \sum_{y} \exp(w_y^T x_i) \right) - w_{y_i}^T x_i \tag{3}$$

We can write this problem in a more vectorial form by noting the log-partition function (the log-sum-exp) $\phi(z) = \log \left( \sum_{y=1}^{K} \exp(z_y) \right)$, and by introducing binary encoding of the ground truth Y.

$$\min_{W \in \mathbb{R}^{K*d}} \frac{\lambda}{2} \|W\|^2 + \frac{1}{n} \sum_{i=1}^{n} \phi(W x_i) - \frac{1}{n} \mathrm{Tr}(Y W X^T) \tag{4}$$

This is a convex problem in W. Indeed the log-sum-exp is a convex function, as can be seen by verifying that it's Hessian is a semi-definite positive matrix. The $l^2$ regularization guarantees that the function we are optimizing is at least $\lambda$ strongly convex.

Let's notice that the gradient of the log-partition function evaluated in $Wx$ is the conditional probability vector of the classes given x : $\nabla \phi(Wx) = (p(y|x; W))_{y \in 1,..,K}$. This will prove useful in the following.

### 1.3 Dual Formulation

As any convex problem, the logistic regression admits dual formulations. In this case, as the problem is unconstrained, these duals are called Fenchel duals. We are given a splitting of the initial function to optimize in the form $f(x) + g(Ax)$ with x a vector, A a matrix, f and g two convex functions. We note $f^*$ and $g^*$ the Fenchel duals of $f$ and $g$ respectively. For simplicity, we do not write down the definition domains of these functions. The Fenchel dual associated to this problem is derived as follows :

$$\begin{aligned}\min_x \; f(x) + g(Ax) &= \min_x \; \max_z z^T x - f^*(z) + \max_y y^T Ax - g*(y) \\ &\geq \max_{z,y} \; \min_x x^T(z + A^T y) - f*(z) - g*(y) \\ &= -\min_y \; f^*(-A^T y) + g^*(y) \quad \text{if} \quad z = -A^T y \quad , -\infty \text{ otherwise.}\end{aligned}$$

Importantly enough, each splitting of the function in arbitrary pairs of convex functions can yield a different dual formulation. For the logistic regression, rather than splitting the function in this canonical form, we get the dual formulation by injecting the equation $f(x) = \max_y x^T y - f^*(y)$ for the $l^2$ regularization and each of the log-sum-exp. We then get the canonical dual problem :

Should I derive the dual formulation here?

$$\max_{\alpha \in \Delta_K^n} \mathscr{D}(\alpha) = -\frac{\lambda}{2}\|W(\alpha)\|^2 + \frac{1}{n}\sum_{i=1}^{n} H(\alpha_i) \tag{5}$$

where $\Delta_K$ is the simplex of dimension K, meaning that $\alpha$ is a n*K matrix whose lines live in the simplex, and H is the entropy of dimension K : $H(\alpha_i) := -\sum_y \alpha_i(y)\log(\alpha_i(y)$. $\alpha$ should be interpreted as a distribution of weight on the different classes for each example. What we call the primal parameters associated to alpha is defined by the equation :

$$w_y(\alpha) := \frac{1}{\lambda n}\sum_i (\mathbb{1}_{y=y_i} - \alpha_i(y))x_i \tag{6}$$

We can also write it with matrices :

$$W(\alpha) = \frac{1}{\lambda n}(Y - \alpha)^T X \tag{7}$$

Or we can separate the influence of each example i through $\alpha_i$ on the resulting vector W :

$$W(\alpha) = \frac{1}{n}\sum_i W_i(\alpha_i) \quad \text{where} \quad W_i(\alpha_i) := \frac{1}{\lambda}(Y_i - \alpha_i)x_i^T \tag{8}$$

This equation correspond to an optimality condition that appears when one derives the dual formulation. Similarly, when we derive the primal problem from the dual problem, we get the optimality condition :

$$\alpha_i(W) = \nabla\phi(Wx_i) = p(y_i|x_i; W) \tag{9}$$

This optimality condition means that at the optimum, the dual problem and the primal problem give the same distribution on the classes of each instance i.

As the primal problem is unconstrained, strong duality holds for this problem.Thus the optimum points $W^*$ and $\alpha^*$ verify equality in both equations 7 and 9. See `https://en.wikipedia.org/wiki/Fenchel%27s_duality_theorem` real conditions.

### 1.4 Stochastic Dual Coordinate Ascent

As the name suggests the SDCA algorithm updates each line of the matrix $\alpha$ at a time. At each time step, one $i \in 1, ..n$ is picked at random. Then $\alpha_i$, which lives in the simplex, is updated so as to maximize the

dual objective. Finding the optimal update for $\alpha_i$ is a constrained optimization problem in dimension K, which can be itself difficult. In his 2013 paper [4], Shalev-Shwartz analyzes the brilliant idea of updating $\alpha_i$ in the direction of a subgradient of the primal loss $\nabla\phi(W(\alpha)x_i)$. This is also what $\alpha_i$ should be equal to : $\alpha_i(W(\alpha))$. One can then either take steps of fixed size in the ascent direction $d_i := \alpha_i(W(\alpha)) - \alpha_i$, or can perform a line search on the segment between $\alpha_i$ and $\alpha_i(W(\alpha))$.

insert drawing of the segment in the simplex.

**Line search** : the function of the step size we want to optimize is

$$f_i(\gamma) = -\frac{\lambda n}{2}\|W(\alpha + \gamma d_{[i]})\|^2 + H(\alpha_i + \gamma d_i) + \text{cst} \tag{10}$$

where he $d_{[i]}$ is the matrix n*K whose line i is $d_i$ and where the rest is 0. The first term is a quadratic function in $\gamma$. It can be expanded with equation 7. We note $\beta = Y - \alpha$ the difference between the ground truth and the dual estimation.

$$\begin{aligned}
-\frac{\lambda n}{2}\|W(\alpha + \gamma d_{[i]})\|_F^2 &= -\frac{\lambda n}{2}\operatorname{Tr}(W(\alpha + \gamma d_{[i]})W(\alpha + \gamma d_{[i]})^T) \\
&= -\frac{1}{2\lambda n}\operatorname{Tr}((\beta - \gamma d_{[i]})^T XX^T(\beta - \gamma d_{[i]})) \\
&= -\frac{1}{2\lambda n}\left[\operatorname{Tr}(\beta XX^T\beta) - 2\gamma\operatorname{Tr}(\beta^T XX^T d_{[i]}) + \gamma^2\operatorname{Tr}(d_{[i]}^T XX^T d_{[i]})\right] \\
&= -\frac{\lambda n}{2}\|W(\alpha)\|_F^2 + \gamma\operatorname{Tr}(W(\alpha)X^T d_{[i]}) - \gamma^2\frac{\|d_i\|^2\|x_i\|^2}{2\lambda n} \\
&= -\frac{\lambda n}{2}\|W(\alpha)\|_F^2 + \gamma\ d_i^T W(\alpha)x_i - \gamma^2\frac{\|d_i\|^2\|x_i\|^2}{2\lambda n}
\end{aligned}$$

We expand the first term of 10 with the formulation above to highlight the quadratic dependency on $\gamma$.

$$f_i(\gamma) = -\gamma^2\frac{\|d_i\|^2\|x_i\|^2}{2\lambda n^2} + \gamma\ d_i^T W(\alpha)x_i + H(\alpha_i + \gamma d_i) + \text{cst} \tag{11}$$

This is indeed a concave function. Its derivatives are as follow.

$$f_i'(\gamma) = -\gamma\frac{\|d_i\|^2\|x_i\|^2}{\lambda n^2} + d_i^T W(\alpha)x_i - \sum_y d_i\log(\alpha_i + \gamma d_i) \tag{12}$$

$$f_i''(\gamma) = -\frac{\|d_i\|^2\|x_i\|^2}{\lambda n^2} - \sum_y \frac{d_i^2}{\alpha_i + \gamma d_i} \tag{13}$$

We can find the $2\epsilon$-approximate maximum of such a function defined on [0,1] by looking at the $\epsilon$-approximate root of its derivative on $[\epsilon, 1-\epsilon]$. This can be done with a stabilized Newton method for instance, as described in the section 9-4 of the book [3].

## 1.5 Duality Gap and Non-Uniform Sampling

As SDCA keeps track of both a dual variable $\alpha$ and a primal variable $W(\alpha)$, abbreviated $W$ in the following, we can evaluate the duality gap between the primal problem evaluated in $W$ and the dual problem evaluated

pseudo code of my algorithm. Specify why I can keep W as well as alpha. What are the memory and time cost of SDCA. How does it compare with SAG etc...

in $\alpha$.

$$
\begin{aligned}
g(\alpha, W) &= \mathscr{P}(W) - \mathscr{D}(\alpha) \\
&= \lambda\|W\|^2 + \frac{1}{n}\sum_{i=1}^{n}\left[\phi(Wx_i) - H(\alpha_i)\right] - \frac{1}{n}\operatorname{Tr}(YWX^T) \\
&= \frac{1}{n}\sum_{i=1}^{n}\left[\lambda\langle W_i(\alpha_i), W\rangle + \phi(Wx_i) - H(\alpha_i) - Y_i^T W x_i\right] \quad \text{where} \quad W_i(\alpha_i) := \frac{1}{\lambda}(Y_i - \alpha_i)x_i^T \\
&= \frac{1}{n}\sum_{i=1}^{n}\left[\operatorname{Tr}(x_i(Y_i - \alpha_i)^T W) + \phi(Wx_i) - Y_i W x_i - H(\alpha_i)\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\left[\phi(Wx_i) - H(\alpha_i) - \alpha_i^T W x_i\right]
\end{aligned}
$$

If we note $s_i = Wx_i$ the score given by our primal model to the different classes on the example i, the total duality gap can then be written as the mean of individual gaps :

$$g_i(\alpha_i, s_i) = \phi(s_i) - H(\alpha_i) - \langle \alpha_i, s_i\rangle \tag{14}$$

$g_i$ is really a duality gap. The log-partition $\phi$ and the entropy $-H$ are convex conjugates. Consequently, $g_i$ is the Fenchel duality gap between $\phi(s_i)$ and $H(\alpha_i)$. It is always positive and it represents the sub-optimality we have on the datapoint i. It can also be expressed as the Kullback-Leibler divergence between the dual distribution $\alpha_i$ and the primal distribution $\nabla_i := \nabla\phi(s_i) = \exp(s_i - \phi(s_i))$. Indeed $\langle \alpha_i, s_i\rangle$ becomes $\langle \alpha_i, \log(\nabla_i)\rangle + \phi(s_i)\langle \alpha_i, \mathbb{1}\rangle = \mathbf{E}_{\alpha_i}[log(\nabla_i)] + \phi(s_i)$

$$g_i(\alpha_i, \nabla_i) = D_{KL}(\alpha_i || \nabla_i) \tag{15}$$

In the article [2], the author apply a non-unform sampling scheme on the algorithm Block Coordinate Frank-Wolfe [1]. Their scheme is to sample proportionally to past-estimates of the individual duality gaps. In SDCA, we have access to duality gap $g_i$ if we pick the element i for almost no extra cost. Indeed each step involves computing $s_i = Wx_i$ to get $\alpha_i(W(\alpha)) = \nabla\phi(s_i)$. So we can get $\phi(s_i)$ in time $O(K)$. $H(\alpha_i)$ is already computed during the line search, and is done in $O(K)$ as well. Finally the scalar product $\langle \alpha_i, s_i\rangle$ has to be computed for a similar cost $O(K)$.

### 1.6 Results

SDCA gives proper results on synthetic gaussian mixtures datasets.

Comparison with the other non-uniform sampling method on a real dataset?

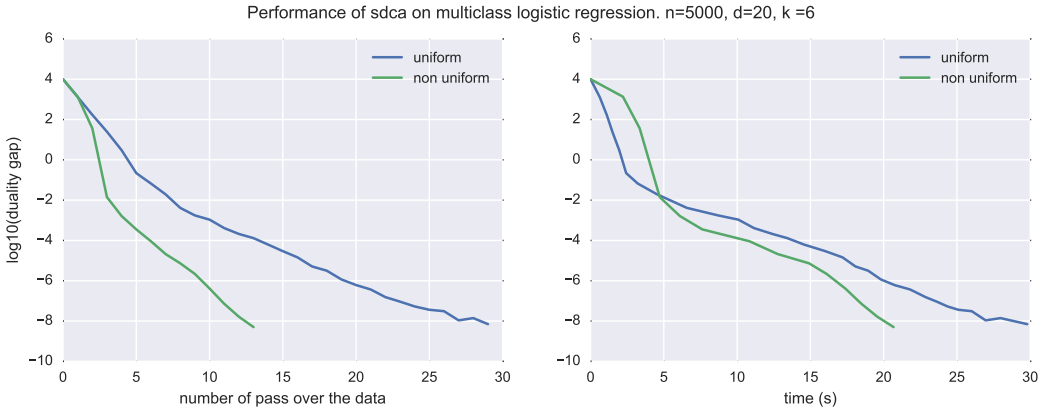


Figure 2: Comparison of the performance of various methods on a synthetic Gaussian mixture dataset.

# 2 Conditional Random Fields

## 2.1 Structured Prediction

Structured prediction is a sub genre of multi-class classification problem. The particularity is that we have structure information about the class themselves. For instance we would like to identify words from images of letters, and we know that there is a chain structure on these letters (OCR). Or we could parse written mathematical expressions and we know that there is a natural tree structure on these expressions. Another typical instance is semantic segmentation in images : we know a priori that boundaries between objects should be sharp and that objects are usually connected areas. Let's write x for the input data point and y for the output class that lives in $\mathcal{Y}_x$. We index $x$ to $\mathcal{Y}$ because the space of classes can depend on $x$. For instance, the word we predict will have as many letters as we have images, and the semantic segmentation will be an image of the same size as the input image.

put citation here

Compared to standard multi-class classification, we need to exploit the structure because the output space $\mathcal{Y}_x$ has a size that is typically exponential in the size of the input x. A brute force approach listing the probabilities for each class to take the max is thus intractable. The structure generally allows us to explore the space of classes in a clever way, to compute the mode of this distribution or the marginal probabilities on some part of the structure. We call the algorithm that returns the mode the *max oracle* and the one that returns the marginals the *marginalization oracle*.

Formally, for any pair (input, output) we want to define a structured score $s_w(x, y)$ parametrized by a vector w. This score should represent the confidence that we have that the point x belongs to the class y through the equation $p(y|x; w) \propto \exp(s_w(x, y))$. Physically, the score is thus the opposite of the energy of the state y for the system x.

$$p(y|x; w) = \frac{e^{s_w(x,y)}}{\sum_{y'} e^{s_w(x,y')}} \tag{16}$$

In the following, we assume that we have a feature extractor for pairs $x, y \mapsto F(x, y) \in \mathbb{R}^d$, either handcrafted or pre-trained. Our score will be a linear combination of these features. We call the parameter w *the weights*.

$$s_w(x, y) = \langle w, F(x, y) \rangle \tag{17}$$

We consider output spaces specified by undirected graphical models, aka Markov Random Fields. The output y is a random variable that factors over the graph $G = (V, E)$. Formally, this means that the joint probability over y can be factorized into potentials over the maximal cliques of the graph. We denote the $\mathcal{C}$ the set of maximal cliques of G, and $\mathcal{S}$ the set of separations between these cliques.

$$\begin{aligned} p(y|x; w) &\propto \exp(s_w(x, y) \\ &\propto \prod_{c \in \mathcal{C}} \exp(s_{w,c}(x, y_c)) \\ &\propto \prod_{c \in \mathcal{C}} \exp(\langle w, F_c(x, y_c) \rangle) \\ &\propto \exp(\langle w, \sum_{c \in \mathcal{C}} F_c(x, y_c) \rangle \end{aligned}$$

We go from the second to the third line by assuming that the score of each clique is itself linear. The consequence of these derivations is that if y factors over a graph, we want the features to be separable the same way :

$$F(x, y) = \sum_{c \in \mathcal{C}} F_c(x, y_c) \tag{18}$$

## 2.2 Maximum Likelihood

We have a set of n pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}_i$ independently sampled. We also have a feature extractor that maps these pairs to $\mathbb{R}^d$. This feature extractor is separable over graphs of the appropriate dimension corresponding to each $\mathcal{Y}_i$. The Conditional Random Field model aims at maximizing the log-likelihood of the weights $w \in \mathbb{R}^d$ given these samples. Once we have learnt the weights, we can predict the class of a new data point x by taking the mode of $p(y|x; w)$. The prediction function is thus $x \mapsto \hat{y} = h_w(x) = \arg\max_{y \in \mathcal{Y}_x} s_w(x, y)$. To avoid overfitting and to make the problem strongly convex, we penalize the squared $l^2$ norm of the weights. The variational CRF problem is written below.

$$\min_{w \in \mathbb{R}^d} f(w) = \frac{\lambda}{2}\|w\|^2 - \frac{1}{n}\sum_{i=1}^{n} \log(p(y_i|x_i; w)) \tag{19}$$

Let's expand the probability term in this formula. We define the corrected features $\psi_i(y)$ of a class y for the point i as the difference between the ground truth features and the features of $(x_i, y)$.

$$\psi_i(y)) := F(x_i, y_i) - F(x_i, y) \tag{20}$$

The negative log-likelihood then becomes the log-partition function (log-sum-exp) $\phi_i(z) := \log(\sum_{y \in \mathcal{Y}_i} e^{z(y)})$ over these features.

$$-\log(p(y_i|x_i; w)) = \log\left(\sum_{y \in \mathcal{Y}_i} \exp(-w^T \psi_i(y))\right) = \phi_i(-A_i^T w) \tag{21}$$

where $A_i$ is the $d \times |\mathcal{Y}_i|$ matrix whose columns are the $\psi_i(y)$ for $y \in \mathcal{Y}_i$. We index $i$ to $\phi_i$ because it affects the range of the sum. The closed formulation of the CRF is thus :

Change the name of $A_i$ for a more typical one. Idem for $\phi_i$

$$\min_{w \in \mathbb{R}^d} f(w) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \phi_i(-A_i^T w) \tag{22}$$

## 2.3 Dual Formulation

We can derive a Fenchel dual to the CRF, as done in

provide reference.

$$\max_{\alpha \in \Delta_1 \times \cdots \times \Delta_n} -\frac{\lambda}{2}\|w(\alpha)\| + \frac{1}{n}\sum_{i=1}^{n} H_i(\alpha_i) \tag{23}$$

where $\Delta_i$ is the simplex of dimension $|\mathcal{Y}_i|$, meaning that $\alpha$ is a n*$|\mathcal{Y}_i|$ matrix whose lines live in the simplex, and $H_i$ is the entropy for distributions over $\mathcal{Y}_i$ : $H(\alpha_i) := -\sum_y \alpha_i(y) \log(\alpha_i(y))$. To simplify the notation, we omit the domain of summation for y in the sum as it can be deduced from the context. We define the *dual weights* given by the optimality condition:

$$w(\alpha) = \frac{1}{\lambda n}\sum_i \sum_y \alpha_i(y)\psi_i(y)$$

At the dual optimum $\alpha^*$, $w(\alpha^*)$ is optimum for the primal problem. This formula can be written in a number of ways, each time outlining some property.

$$w(\alpha) = \frac{1}{\lambda n} A\alpha \tag{24}$$

$$= \frac{1}{\lambda}\mathbf{E}_i[\mathbf{E}_{y \sim \alpha_i}[\psi_i(y)]] \tag{25}$$

$$= \frac{1}{\lambda}\mathbf{E}_i[F(x_i, y_i)] - \frac{1}{\lambda}\mathbf{E}_i[\mathbf{E}_{y \sim \alpha_i}[F(x_i, y)]] \tag{26}$$

where A is the horizontal concatenation of the $A_i$. It is a matrix of size $d \times \sum_i |\mathcal{Y}_i|$. The expectations over i assume i is uniform random variable taking its values between 1 and n.

**Interpretation :** The primal problem is a regularized maximization of the likelihood of w. In the dual problem, we control directly the probabilities given to each class on the training samples. There is two conflicting terms. The first one aims at minimizing the size of the centroid of the corrected features. Since $\psi_i(y_i) = 0$, it is minimal for the distribution that puts all the weights on the ground truth. The second term aims at maximizing the entropy of this distribution. It pushes the $\alpha_i$ towards a more uniform distribution. Thus the role of the terms is the inverse of the primal model : the data fitting term is on the left. It is the squared distance between the centroid of the ground truth features and the centroids predicted by the dual model. The entropy on the right is the regularization.

We can also derive the primal problem from the dual problem. We then get another optimality condition $\alpha(w^*) = \alpha*$ where $\alpha(w)$ is the density on the training set defined by the weights w in equation 16.

$$\forall i, \alpha_i(w) = \nabla\phi_i(-A_i^T w) = p(.|x; w) \propto \exp(-w^T \psi_i(.)) \tag{27}$$

**Naming :** We call *primal model*, the one where we are given *primal weights* w, from which we deduce *primal probabilities* $\alpha_i(w)$. We call *dual model*, the one where we are given *dual probabilities* $\alpha_i$, from which we deduce *dual weights* $w(\alpha)$ as the centroid of the corrected feature vectors. The optimality conditions tell us that at the optimum, these two models are equal.

## 2.4 Duality Gaps

The duality gap $g(w, \alpha)$ is the difference between the value of the primal problem evaluated in $w$, and the value of the dual problem evaluated in $\alpha$. It is interesting to look at the duality gap for both the primal model and the dual model, i.e. the duality gap between the primal weights and the primal probability, and the duality gap between the dual weights and the dual probability.

**Primal model:**

$$g(w, \alpha(w)) = \frac{\lambda}{2}\|w - w(\alpha(w))\|^2 \tag{28}$$

In this formula appears $w(\alpha(w))$, what the dual model created by the primal probabilities think the weights should be. $w(\alpha(w))$ is actually proportional to the derivative of the partition function with respect to w :

$$w(\alpha(w)) = \frac{1}{\lambda n}\sum_i \mathbf{E}_{p(y|x;w)}[\psi_i(y)] = -\frac{1}{\lambda}\nabla_w(\phi_i(-A_i^T w)) \tag{29}$$

*Remark:* In full batch gradient descent, the update formula for a step size $\gamma$ is : $w^+ = (1 - \gamma\lambda)w - \gamma\nabla_w(\phi_i(-A_i^T w))$. For a (very large) step size $\gamma = 1/\lambda$, the update yields $\|w^+ - w(\alpha(w))\| = 0$.

**Dual model:**

$$g(w(\alpha), \alpha) = \frac{1}{n}\sum_i D_{KL}(\alpha_i||\alpha_i(w(\alpha)) \tag{30}$$

Once again, we observe a loop. The duality gap is the Kullback-Leibler divergence between the dual probabilities, and the primal probabilities of the primal problem created by $w(\alpha)$. Each of the divergence in the sum above is of course positive. They can be thought of as the duality gaps associated with each data point for the dual model. They are also equal to the Fenchel duality gap between the negative entropy and the partition function :

$$g_i(w, \alpha_i) = D_{KL}(\alpha_i||\alpha_i(w(\alpha)) = \phi_i(-A_i^T w) - H_i(\alpha_i) - \langle\alpha_i, -A_i^T w\rangle$$

*Remark:* Performing SDCA with a step-size $\gamma = 1$ gives the update formula $\alpha_i^+ = \alpha_i(w(\alpha))$, i.e. $D(\alpha_i^+||\alpha_i(w(\alpha))) = 0$.

Look at the step sizes time series in SDCA.

**Idea:** Sampling according to these individual duality gaps, i.e. sampling more often points that are more suboptimal should improve the convergence rate.

## 2.5 Marginalization

Taking the raw dual problem is intractable. The variable $\alpha$ itself is a priori too big to fit in memory : for each data point, it stores a probability vector that is exponential in the size of this data point. This is where the structure comes into play. We assume that the output class y factors over a triangulated Markov random field $G = (V, E)$. Then the joint probability $\alpha(y)$ can be written as a function of its marginals $\mu$. We keep the notation $\mathcal{C}$ for the set of maximal cliques of G, and $\mathcal{S}$ the set of separations between these cliques. The marginal over a set of nodes $s$, is given by $\mu_s(y_s) = \sum_{y'|y'_s=y_s} \alpha(y)$.

$$\alpha(y) = \frac{\prod_{c\in\mathcal{C}} \mu_c(y_c)}{\prod_{s\in\mathcal{S}} \mu_s(y_s)} \tag{31}$$

$\mu$ should be thought of as the marginals over the cliques only. The $\mu_s$ are a byproduct of these. When we go from the joint $\alpha_i$ to the marginal $\mu_i$, we go from a size $\mathcal{Y}_i$, to a size $\sum_{c\in\mathcal{C}_i} |\,\mathcal{Y}_c\,|$. If each component of y can take K values, then we go from $K^{|V_i|}$ to $\sum_{c\in\mathcal{C}_i} K^{|c|}$ which should be considerably smaller. The formula 31 make sense as long as the marginals on the maximal cliques are coherent, meaning that they agree about the values of the marginals on the separations. Our algorithm will make sure that this coherence is preserved at each step. We could get a computationally more efficient formula by considering a junction tree of G, but 31 is more readable. It allows us to translate each of the functions previously seen with the joint, in functions of the marginals. First the entropy and the Kullback-Leibler divergence.

$$H_{|\,\mathcal{Y}\,|}(\alpha) = \sum_c H_{|c|}(\mu_c) - \sum_s H_{|s|}(\mu_s) =: \mathcal{H}(\mu) \tag{32}$$

$$D(\alpha||\alpha') = \sum_c D(\mu_c||\mu'_c) - \sum_s D(\mu_s||\mu'_s) =: \mathcal{D}(\mu||\mu') \tag{33}$$

We can also write the dual weights as a function of the marginals :

$$\begin{aligned}
\mathbf{E}_{\alpha_i}[\psi_i] &= \sum_{y\in\mathcal{Y}_i} \alpha_i(y)\psi_i(y) \\
&= \sum_{c\in\mathcal{C}_i} \sum_{y\in\mathcal{Y}_i} \alpha_i(y)\psi_{i,c}(y_c) \\
&= \sum_{c\in\mathcal{C}_i} \sum_{y_c\in\mathcal{Y}_c} \Big( \sum_{y'|y'_c=y_c} \alpha_i(y') \Big) \psi_{i,c}(y_c) \\
&= \sum_{c\in\mathcal{C}_i} \sum_{y_c\in\mathcal{Y}_c} \mu_{i,c}(y_c)\psi_{i,c}(y)
\end{aligned}$$

As we have done with $w(\alpha)$, we can reformulate this. Let $B_i$ be the matrix of size $d \times \sum_{c\in\mathcal{C}_i} |\,\mathcal{Y}_c\,|$, whose columns are the $\psi_{i,c}(y)$. Let $B$ be the horizontal concatenation of the $B_i$. Let $\mu$ be the vector containing all the $\mu_i$.

$$w(\mu) = \frac{1}{\lambda n} \sum_i \sum_{c\in\mathcal{C}_i} \mathbf{E}_{\mu_i}[\psi_{i,c}] \tag{34}$$

$$= \frac{1}{\lambda} B\mu \tag{35}$$

With equations 32 and 35 we can write the dual problem as a maximization over the marginals.

$$\max_{\forall i \forall c, \mu_{i,c} \in \Delta_{|c|}} -\frac{\lambda}{2}\|w(\mu)\|^2 + \frac{1}{n}\sum_i \mathcal{H}_i(\mu) \tag{36}$$

## 2.6 SDCA

As we have seen in the introduction about structured prediction, the separation of the feature vectors is equivalent to the factorisation of the primal probabilities. We can get the primal marginals with a marginalization oracle, such as message passing on a junction tree.

**Algorithm 1** SDCA for CRF

Let $\forall i, c, \mu_{i,c}^{(0)} := \frac{1}{|c|}$ and $w^{(0)} := \frac{1}{\lambda n} B\mu^{(0)}$
Let $\forall i g_i = 1$ (optional)
**for** $k = 0 \dots K$ **do**
  Pick $i$ at random in $\{1, \dots, n\}$ (optionally, proportional to $g_i$)
  Compute $\forall c, \nabla_{i,c}(y_c) := p(y_c|x; w^{(k)})$ (marginalization oracle)
  Let $g_i = \mathcal{D}(\mu_i || \nabla_i)$ (optional)
  Let $d_i = \nabla_i - \mu_i^{(k)}$ (ascent direction)
  Let $v_i = \frac{1}{\lambda} B_i d_i$ (primal direction)
  Solve $\gamma^* = \arg\max_{\gamma \in [0,1]} \mathcal{H}_i(\mu_i^{(k)} + \gamma d_i) - \frac{\lambda n}{2}\|w^{(k)} + \frac{\gamma}{n} v_i\|^2$ (Line Search)
  Update $\mu_i^{(k+1)} := \mu_i^{(k)} + \gamma^* d_i$
  Update $w^{(k+1)} := w^{(k)} + \frac{\gamma^*}{n} v_i$

**Complexity:** ascent direction in $O(\sum_c |\mathcal{Y}_c|)$ and primal direction in $O(d * \sum_c |\mathcal{Y}_c|)$. The line search is not too expensive. Each function or gradient evaluation is $O(\sum_c |\mathcal{Y}_c|)$.

Note that the computation of $w^{(0)}$ can usually be hand made very efficiently.

Every 10 pass, do a full pas over the data to compute the true duality gap, which gives a stopping criterion.

**Line Search:** we optimize over $\gamma \in [0, 1]$ because we want a convex combination of $\mu_i$ and $\nabla_i$. It gives us a guarantee that we stay in the simplex without any further checks. Plus $\gamma$ a priori has no incentives to be outside of $[0, 1]$, since $\mu_i$ and $\nabla_i$ are converging towards each other. The marginals over the separation $\mu_{i,s}$ and $d_{i,s}$ are computed once and for all at the beginning of the line search.

$$\begin{aligned}
f(\gamma) &= \mathcal{H}_i(\mu_i^{(k)} + \gamma d_i) - \frac{\lambda n}{2}\|w^{(k)} + \frac{\gamma}{n} v_i\|^2 \\
&= \sum_c H_{|c|}(\mu_c + \gamma d_{i,c}) - \sum_s H_{|s|}(\mu_s + \gamma d_{i,s}) - \frac{\lambda n}{2}\|w^{(k)}\|^2 - \gamma\lambda\langle w^{(k)}, v_i\rangle - \gamma^2 \frac{\lambda}{2n}\|v_i\|^2 \\
f'(\gamma) &= -\sum_c \langle d_{i,c}, \log(\mu_c + \gamma d_{i,c})\rangle + \sum_s \langle d_{i,s}, \log(\mu_s + \gamma d_{i,s})\rangle - \lambda\langle w^{(k)}, v_i\rangle - \gamma\frac{\lambda}{n}\|v_i\|^2 \\
f''(\gamma) &= -\sum_c \sum_{y_c} \frac{d_{i,c}(y_c)^2}{\mu_c(y_c) + \gamma d_{i,c}(y_c)} + \sum_s \sum_{y_s} \frac{d_{i,s}(y_s)^2}{\mu_s(y_s) + \gamma d_{i,s}(y_s)} - \frac{\lambda}{n}\|v_i\|^2
\end{aligned}$$

# References


[1] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-Coordinate Frank-Wolfe Optimization for Structural SVMs. *arXiv:1207.4747 [cs, math, stat]*, July 2012. arXiv: 1207.4747.

[2] Anton Osokin, Jean-Baptiste Alayrac, Isabella Lukasewitz, Puneet Dokania, and Simon Lacoste-Julien. Minding the Gaps for Block Frank-Wolfe Optimization of Structured SVMs. In *PMLR*, pages 593–602, June 2016.

[3] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, Cambridge ; New York, 2nd ed edition, 1992.

[4] Shai Shalev-Shwartz and Tong Zhang. Accelerated Proximal Stochastic Dual Coordinate Ascent for Regularized Loss Minimization. *arXiv:1309.2375 [cs, stat]*, September 2013. arXiv: 1309.2375.