Mixing Strategies in Data Compression

Christopher Mattern

Fakultät für Informatik und Automatisierung
Technische Universität Ilmenau
Ilmenau, Germany
christopher.mattern@tu-ilmenau.de

Abstract

We propose geometric weighting as a novel method to combine multiple models in data compression. Our results reveal the rationale behind PAQ-weighting and generalize it to a non-binary alphabet. Based on a similar technique we present a new, generic linear mixture technique. All novel mixture techniques rely on given weight vectors. We consider the problem of finding optimal weights and show that the weight optimization leads to a strictly convex (and thus, good-natured) optimization problem. Finally, an experimental evaluation compares the two presented mixture techniques for a binary alphabet. The results indicate that geometric weighting is superior to linear weighting.

1 Introduction

1.1 Background

The combination of multiple models is a central aspect of many modern data compression algorithms, such as Prediction by Partial Matching (PPM) [2, 8, 9], Context Tree Weighting (CTW) [10, 11] or "Pack" (PAQ) [5, 8]. All of these algorithms belong to the class of statistical data compression algorithms, which share a common structure: The compressor consists of a *model* and a *coder*; and it processes the data (a string $x^n \in \mathcal{X}^n$ for some alphabet $\mathcal{X}, |\mathcal{X}| \geq 2$) sequentially. In the k-th step, $1 \leq k \leq n$, the model estimates the probability distribution $P(\,\cdot \mid x^{k-1})$ of the next symbol based on the already processed sequence $x^{k-1} = x_1 x_2 \ldots x_{k-1}$. The task of the coder is to map a symbol $x \in \mathcal{X}$ to a codeword of a length close to $-\log P(x \mid x^{k-1})$ bits (throughout this paper log is to the base two). For decompression the coder maps the encoding, given $P(\,\cdot \mid x^{k-1})$, to x. Arithmetic Coding (AC) closely approximates the ideal code length and is known to be asymptotically optimal [3]. Therefore, the prediction accuracy of the model is crucial for compression.

Mixture models or mixtures combine multiple models into a single model suitable for encoding. Let us now consider a simple example, which gives two reasons for our interest in mixtures. First, assume that we have $m > 1$ models available. Model $i, 1 \leq i \leq m$, maps an arbitrary x^n to a prediction $P_i(x^n)$ (a probability distribution), where

$$P_i(x^n) = \prod_{k=1}^{n} P_i(x_k \mid x^{k-1}) = \prod_{k=1}^{n} \frac{P_i(x^k)}{P_i(x^{k-1})} \tag{1}$$

and $P_i(x^k) > 0,\ 1 \leq i \leq m,\ k \geq 0$. When we compress x^n with a single model i, we need to encode the choice of i in $-\log W(i)$ bits (where $W(i)$ is the prior probability of

selecting model i) and we need to store the encoded string, which adds $-\log P_i(x^n)$ bits. If we knew x^n in advance, we could select

$$i = \arg \min_{1 \leq j \leq m} \left[-\log(W(j)) - \log(P_j(x^n)) \right]. \tag{2}$$

Surprisingly (as previously observed in e.g., [7]), a simple linear mixture $P(x^n) := \sum_{j=1}^{m} W(j) P_j(x^n)$ will never do worse than (2), since

$$\begin{aligned} -\log(W(i)) - \log(P_i(x^n)) &= -\log(W(i) P_i(x^n)) \\ &\geq -\log \sum_{j=1}^{m} \left(W(j) P_j(x^n) \right), \end{aligned}$$

where i is the model that minimizes (2). Such a mixture makes it possible to combine the advantages of different models without cumulating their disadvantages. Secondly, the sequential processing allows us to refine the mixture adaptively (in favor of the locally more accurate models).

1.2 Previous Work

Most of the major statistical compression techniques (PPM, CTW and PAQ) are based on mixtures. In PPM the concept of "escape" symbols is related to the computation of a recursively defined mixture distribution. The escape probability plays the role of a weight in a linear mixture. In [2] Bunton gave a very comprehensive (at that time) synopsis on that topic. Previously, several different methods for the estimation of escape probabilities had been proposed, e.g., PPMA, PPMB, PPMC, PPMD, PPMP, PPMX [8], PPMII [9]. CTW relies on the efficient combination of exponentially many (depending on a "tree depth" parameter) models for tree sources. However, the structure of PPM and CTW restrict the type of models they combine (order-N models for PPM and models for tree sources for CTW). Recently, some of the techniques of CTW led to β-weighting [4], as a linear general-purpose weighting method. We are interested in general-purpose mixture techniques, which combine arbitrary (and eventually totally different) models. The practical success of this approach was initiated by Mahoney with PAQ (see [8] for details). PAQ combines a large amount of totally different models (e.g., models for text, for images, etc.). As a minor part earlier work we successfully employed a simple linear mixture model for encoding Burrows-Wheeler-Transform (BWT) output and proposed a method for the parameter optimization on training data [6].

1.3 Our Contribution

In Section 3 we propose geometric weighting as a novel non-linear mixture technique. We obtain the geometric mixture as the solution of a divergence minimization problem. In addition we show that PAQ-mixing is a special case of geometric weighting for a binary alphabet. Since geometric weighting depends on a set of weights, we examine the problem of weight optimization and propose a corresponding optimization method. In Section 4 we focus on linear mixtures. In a fashion analogous to Section 3 we describe a new generic linear mixture and investigate the problem of weight optimization. Finally, we compare the behavior of the implementations (for a binary alphabet) of the two proposed mixture techniques and of β-weighting in Section 5. Results indicate that geometric weighting is superior to the other mixture methods.

2 Preliminaries

First, we fix some notation. Let $\mathcal{X}$ denote an alphabet of cardinality $1 < |\mathcal{X}| < \infty$ and let $x_i^j = x_i x_{i+1} \dots x_j$ be a sequence of length $n = j - i + 1$ over $\mathcal{X}$. For short we may write x^n for x_1^n. Abbreviations such as $(a_i)_{1 \le i \le n}$ expand to $(a_1\ a_2\ \dots\ a_n)$ and denote row vectors. Boldface letters indicate matrices or vectors, "^{T}" denotes the transpose operator, $\mathbf{1}_m := (1\ 1\ \dots\ 1)^T \in \mathbb{R}^m$ and $\Omega_m := \{\boldsymbol{v} \in \mathbb{R}^m \mid \boldsymbol{v} \ge \mathbf{0},\ \boldsymbol{v}^T \mathbf{1}_m = 1\}$. We use log to denote the logarithm with base two, ln denotes the natural logarithm.

Suppose that we want to compress a string $x^n \in \mathcal{X}^n$ sequentially. In every step $1 \le k \le n$ a *model* $M : \cup_{k \ge 0} \mathcal{X}^k \to \mathcal{P}$ maps the already known prefix x^{k-1} of x^n to a *model distribution* $P(\,\cdot \mid x^{k-1}),\ P \in \mathcal{P}$, where $\mathcal{P} := \{Q : \mathcal{X} \to (0,1) \mid \sum_{x \in \mathcal{X}} Q(x) = 1\}$. An encoder translates this into a code of length close to $-\log P(x \mid x^{k-1})$ bits for x. Now, if there are $m > 1$ *submodels* $M_1, M_2, \dots, M_m$ (or submodels $1, 2, \dots, m$, for short), we require a *mixture function* $f_k \colon \mathcal{X} \times \mathcal{P}^m \to (0,1)$ to map the m corresponding distributions $P_1, P_2, \dots, P_m$ to a single distribution $P(x) = f_k(x, P_1, P_2, \dots, P_m), P \in \mathcal{P}$, in step k; f_k may depend on x^{k-1}.

An approach in information theory is to suppose that x^n was generated by an unknown mechanism, which is called a *source*. W.l.o.g. we may assume that x^n was generated sequentially: In every step k the source draws x according to an arbitrary *source distribution* $P' \in \mathcal{S} := \{Q : \mathcal{X} \to [0,1] \mid \sum_{x \in \mathcal{X}} Q(x) = 1\}$ (i.e., the distribution P' may vary from step to step) and appends it to x^{k-1} to yield $x^k = x^{k-1}x$. When we encode x, using a model distribution $P \in \mathcal{P}$, we obtain an expected code length of

$$\sum_{x \in \mathcal{X}} P'(x) \log \frac{1}{P(x)} = \underbrace{\sum_{x \in \mathcal{X}} \left[P'(x) \log \frac{1}{P'(x)} \right]}_{H(P')} + \underbrace{\sum_{x \in \mathcal{X}} P'(x) \left[\log \left(\frac{1}{P(x)} \right) - \log \frac{1}{P'(x)} \right]}_{D(P' \| P)},$$

where $H(P')$ is the *source entropy* and $D(P' \parallel P)$ is the *KL-divergence* [3], which measures the redundancy of P relative to P'. Our aim is to find a P, that minimizes the code length. Since $H(P')$ is fixed (by the source), we want to minimize $D(P' \parallel P)$. We have $D(P' \parallel P) \ge 0$, which is zero iff $P = P'$, i.e., the best model distribution is the source distribution itself.

3 Geometric Mixtures

This section contains the major part of our work: We derive geometric weighting as a novel method for combining multiple models. Now suppose that we have m model distributions $P_1, P_2, \dots, P_m$ available in step k. Since the source distribution P' is *unknown* (if it exists at all) we try to identify an approximate source distribution $P \in \mathcal{S} \cap \mathcal{P}$, which we can use as a model distribution. It should be "close" (in the divergence-sense) to good models and "far away" from bad models. The terms good and bad refer to short and long code lengths (due to past observations and/or prior knowledge). We assume that we are given a set of non-negative weights $w_i, 1 \le i \le m$, $\sum_{i=1}^m w_i > 0$ (in Section 3.2 we discuss a method of weight estimation), which quantify how well model i fits the unknown source distribution. Summarizing, we are looking for the distribution

$$P := \arg\min_{Q \in \mathcal{P}} \sum_{i=1}^{m} w_i D(Q \parallel P_i). \tag{3}$$

3.1 Divergence Minimization

In order to solve (3) we adopt the method of Lagrangian multipliers. First, we set $Q(x \mid x^{k-1}) = \theta_x$ and $\boldsymbol{\theta}^T = (\theta_x)_{x\in\mathcal{X}}$ to omit the implicit dependence on k and to simplify the equations. Now we rewrite (3) to yield

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \sum_{i=1}^{m} w_i \sum_{x\in\mathcal{X}} \theta_x \left[\log(\theta_x) - \log(P_i(x \mid x^{k-1}))\right], \\ &\text{s.t.} \sum_{x\in\mathcal{X}} \theta_x = 1 \text{ and } \theta_x > 0, x \in \mathcal{X} \end{aligned} \tag{4}$$

and formulate its Lagrangian

$$\begin{aligned} L(\boldsymbol{\theta}, \lambda, \boldsymbol{\mu}) = &\sum_{i=1}^{m} w_i \sum_{x\in\mathcal{X}} \theta_x \left[\log(\theta_x) - \log(P_i(x \mid x^{k-1}))\right] \\ &- \lambda \left(1 - \sum_{x\in\mathcal{X}} \theta_x\right) - \sum_{x\in\mathcal{X}} (\mu_x \theta_x). \end{aligned}$$

The variable λ and the vector $\boldsymbol{\mu} = (\mu_x)_{x\in\mathcal{X}}$ denote the Lagrange multipliers. A local minimum $\boldsymbol{\theta}^*, \lambda^*, \boldsymbol{\mu}^*$ satisfies the Karush-Kuhn-Tucker (KKT) conditions (see, e.g. [1])

$$\frac{\partial L(\boldsymbol{\theta}^*, \lambda^*, \boldsymbol{\mu}^*)}{\partial \theta_x} = 0, \tag{5}$$

$$\theta_x^* > 0,\ \mu_x^* \geq 0,\ \theta_x^* \mu_x^* = 0 \tag{6}$$

for all $x \in \mathcal{X}$ and

$$\sum_{x\in\mathcal{X}} \theta_x^* = 1. \tag{7}$$

Due to (6) we obtain $\mu_x^* = 0$ for all $x \in \mathcal{X}$. Equation (5) can be transformed to

$$\left(\sum_{i=1}^{m} w_i\right) + \lambda + \log\left(\theta_x^{* \sum_{i=1}^{m} w_i}\right) = \log \prod_{i=1}^{m} P_i(x \mid x^{k-1})^{w_i}. \tag{8}$$

Now we fix a disjoint pair $x \neq x'$ of symbols from $\mathcal{X}$ and subtract the corresponding instances of (8), which results in

$$\theta_{x'}^* = \theta_x^* \prod_{i=1}^{m} \left[\frac{P_i(x' \mid x^{k-1})}{P_i(x \mid x^{k-1})}\right]^{w_i'}, \text{ where } w_i' := \frac{w_i}{\sum_{j=1}^{m} w_j}. \tag{9}$$

Again, we fix a single character x and substitute any other occurrence of $x' \neq x$ in (7) via (9). Thus we have

$$1 = \theta_x^* + \theta_x^* \sum_{x'\in\mathcal{X}\setminus\{x\}} \prod_{i=1}^{m} \left[\frac{P_i(x' \mid x^{k-1})}{P_i(x \mid x^{k-1})}\right]^{w_i'},$$

which we rewrite to yield

$$\theta_x^* = \frac{\prod_{i=1}^{m} P_i(x \mid x^{k-1})^{w_i'}}{\sum_{x'\in\mathcal{X}} \prod_{i=1}^{m} P_i(x' \mid x^{k-1})^{w_i'}}. \tag{10}$$

Finally, we reintroduce the dependencies on k and obtain the geometric mixture

$$P(x \mid x^{k-1}) = f_k(x, P_1, P_2, \ldots, P_m) := \frac{\prod_{i=1}^m P_i(x \mid x^{k-1})^{w_i/(\boldsymbol{w}^T \mathbf{1}_m)}}{\sum_{x' \in \mathcal{X}} \prod_{i=1}^m P_i(x' \mid x^{k-1})^{w_i/(\boldsymbol{w}^T \mathbf{1}_m)}}, \tag{11}$$

where $\boldsymbol{w}^T = (w_i)_{1 \le i \le m}$ is composed of the non-negative weights w_i. It remains to show that (10) minimizes (4). For this, we observe that the Hessian of (4) is

$$\boldsymbol{w}^T \mathbf{1}_m \cdot \operatorname{diag}\left((1/\theta_x)_{x \in \mathcal{X}}\right),$$

which is positive definite, since $\theta_x > 0$ for all $x \in \mathcal{X}$.

3.2 Weight Estimation and Convexity

The mixture function (11) requires m non-negative weights $\boldsymbol{w}^T = (w_i)_{1 \le i \le m}$, which we still need to obtain. In our situation the sequence x^n is known (and fixed) and the sequence probability is given as a function of $\boldsymbol{w}$ as

$$\prod_{k=1}^n f_k(x_k, P_1, P_2, \ldots, P_m) = \prod_{k=1}^n \frac{\prod_{i=1}^m P_i(x_k \mid x^{k-1})^{w_i/(\boldsymbol{w}^T \mathbf{1}_m)}}{\sum_{x' \in \mathcal{X}} \prod_{i=1}^m P_i(x' \mid x^{k-1})^{w_i/(\boldsymbol{w}^T \mathbf{1}_m)}}. \tag{12}$$

We now wish to find a weight vector $\boldsymbol{w}$, which maximizes (12) (a maximum-likelihood estimation). Since a maximization of the sequence probability is equivalent to a minimization of its code length, we may alternatively solve

$$\min_{\boldsymbol{w}} \sum_{k=1}^n \left(-\log \frac{\prod_{i=1}^m P_i(x_k \mid x^{k-1})^{w_i/(\boldsymbol{w}^T \mathbf{1}_m)}}{\sum_{x' \in \mathcal{X}} \prod_{i=1}^m P_i(x' \mid x^{k-1})^{w_i/(\boldsymbol{w}^T \mathbf{1}_m)}} \right). \tag{13}$$

We define $\boldsymbol{w}^*$ to be the minimizer of (13).

Now we want to show that the cost function of (13) is convex. Since the cost function is a sum, we analyze a slight modification of a single term $l(\boldsymbol{w}) := -\ln(g(\boldsymbol{w})/h(\boldsymbol{w}))$ (since $\log(x) \sim \ln(x)$). W.l.o.g. we may assume that $\boldsymbol{w} \in \Omega_m$ (due to (9)). In order to simplify the analysis of the Hessian of $l(\boldsymbol{w})$ we set

$$\begin{aligned}
g(\boldsymbol{w}) &:= \prod_{i=1}^m P_i(x_k \mid x^{k-1})^{w_i} = e^{\sum_{i=1}^m w_i \ln P_i(x_k \mid x^{k-1})} = e^{\boldsymbol{w}^T \boldsymbol{Q}(x_k)}, \\
h(\boldsymbol{w}) &:= \sum_{x \in \mathcal{X}} \prod_{i=1}^m P_i(x \mid x^{k-1})^{w_i} = \sum_{x \in \mathcal{X}} e^{\boldsymbol{w}^T \boldsymbol{Q}(x)}, \\
\boldsymbol{Q}(x)^T &:= (\ln P_i(x \mid x^{k-1}))_{1 \le i \le m}, \\
p_x &:= e^{\boldsymbol{w}^T \boldsymbol{Q}(x)} / \sum_{x' \in \mathcal{X}} e^{\boldsymbol{w}^T \boldsymbol{Q}(x')} = f_k(x, P_1, P_2, \ldots, P_m)
\end{aligned}$$

and we obtain

$$\begin{aligned}
\nabla g(\boldsymbol{w})/g(\boldsymbol{w}) &= \boldsymbol{Q}(x_k), & \nabla^2 g(\boldsymbol{w})/g(\boldsymbol{w}) &= \boldsymbol{Q}(x_k)\boldsymbol{Q}(x_k)^T, \\
\nabla h(\boldsymbol{w})/h(\boldsymbol{w}) &= \sum_{x \in \mathcal{X}} p_x \boldsymbol{Q}(x), & \nabla^2 h(\boldsymbol{w})/h(\boldsymbol{w}) &= \sum_{x \in \mathcal{X}} p_x \boldsymbol{Q}(x)\boldsymbol{Q}(x)^T.
\end{aligned}$$

The Hessian of $l(\boldsymbol{w})$ is positive definite, since for $\boldsymbol{v} \neq 0, \boldsymbol{v} \in \mathbb{R}^m$

$$\begin{aligned}
\boldsymbol{v}^T \nabla^2 l(\boldsymbol{w}) \boldsymbol{v} &= \boldsymbol{v}^T \left(\frac{\nabla g(\boldsymbol{w}) \nabla g(\boldsymbol{w})^T}{g(\boldsymbol{w})^2} - \frac{\nabla^2 g(\boldsymbol{w})}{g(\boldsymbol{w})} + \frac{\nabla^2 h(\boldsymbol{w})}{h(\boldsymbol{w})} - \frac{\nabla h(\boldsymbol{w}) \nabla h(\boldsymbol{w})^T}{h(\boldsymbol{w})^2} \right) \boldsymbol{v} \\
&= \left(\frac{\boldsymbol{v}^T \nabla g(\boldsymbol{w})}{g(\boldsymbol{w})} \right)^2 - \boldsymbol{v}^T \frac{\nabla^2 g(\boldsymbol{w})}{g(\boldsymbol{w})} \boldsymbol{v} + \boldsymbol{v}^T \frac{\nabla^2 h(\boldsymbol{w})}{h(\boldsymbol{w})} \boldsymbol{v} - \left(\frac{\boldsymbol{v}^T \nabla h(\boldsymbol{w})}{h(\boldsymbol{w})} \right)^2 \\
&= \left(\boldsymbol{v}^T \boldsymbol{Q}(x_k) \right)^2 - \left(\boldsymbol{v}^T \boldsymbol{Q}(x_k) \right)^2 + \sum_{x \in \mathcal{X}} \left(p_x \left(\boldsymbol{v}^T \boldsymbol{Q}(x) \right)^2 \right) - \left(\sum_{x \in \mathcal{X}} p_x \boldsymbol{v}^T \boldsymbol{Q}(x) \right)^2 \\
&= \sum_{x \in \mathcal{X}} \left(p_x \left(\boldsymbol{v}^T \boldsymbol{Q}(x) \right)^2 \right) - \left(\sum_{x \in \mathcal{X}} p_x \boldsymbol{v}^T \boldsymbol{Q}(x) \right)^2 \\
&> 0
\end{aligned}$$

holds, where the last line is due to Jensen's inequality (since $\sum_{x \in \mathcal{X}} p_x = 1$). It follows that the problem (13) is strictly convex and there exists a single global minimizer $\boldsymbol{w}^* \in \Omega_m$.

We solve the problem (13) with an optimization method tailored to a natural requirement in statistical compression: The sequence to be compressed is processed only once. Since the cost function is convex, the optimization algorithm does not need strong global search capabilities. A possible method-of-choice is an instance of iterative gradient descent [1]. In the k-th step we use the estimates $\boldsymbol{w}(k)$ in place of $\boldsymbol{w}^*$ (in (11)). Initially we set $\boldsymbol{w}(0) = 1/m \cdot \mathbf{1}_m$. In each step k we adjust the weight vector $\boldsymbol{w}(k-1)$ after we observe x_k via a step towards the direction of steepest descent, i.e.,

$$-\alpha_k \nabla_{\boldsymbol{w}} \left(-\log f_k(x_k, P_1, P_2, \ldots, P_m) \right) \tag{19}$$

where $\alpha_k > 0$ is the step size in the k-th step. The choice of α_k is crucial for the convergence of $\boldsymbol{w}(k)$ to $\boldsymbol{w}^*$ [1] (see Sections 3.3 and 5). In the case of a geometric mixture function we have

$$\boldsymbol{w}(k) := \max \left\{ \varepsilon \mathbf{1}_m,\ \boldsymbol{w}(k-1) + \alpha_k \frac{(\boldsymbol{Q}(x_k) - q_{x_k} \mathbf{1}_m) - \sum_{x \in \mathcal{X}} p_x \left(\boldsymbol{Q}(x) - q_x \mathbf{1}_m \right)}{\boldsymbol{w}^T \mathbf{1}_m} \right\},$$

where $q_x := (\boldsymbol{w}^T \boldsymbol{Q}(x))/(\boldsymbol{w}^T \mathbf{1}_m)$. As an implementation detail $\varepsilon > 0$ is a small constant to bound the weights away from zero and to avoid a division by zero in (11).

3.3 PAQ Mixtures or Geometric Mixtures for a Binary Alphabet

Before we examine the details of "the" PAQ mixture method, we need to clarify that there exist multiple PAQ mixture mechanisms [8]. We focus on the latest instance, which was introduced in 2005 as a part of PAQ7. PAQ computes mixtures for a binary alphabet and works with the probability of one-bits. The mixture is defined as follows

$$f_k(1, P_1, P_2, \ldots, P_m) := \mathrm{sq} \left(\sum_{i=1}^{m} w_i(k-1)\, \mathrm{st}(P_i(1 \mid x^{k-1})) \right), \tag{20}$$

$$w_i(k) := w_i(k-1) + \alpha (x_k - f_k(1, P_1, P_2, \ldots, P_m))\, \mathrm{st}(P_i(1)), \tag{21}$$

where x_k is the bit we observed in step k and

$$\mathrm{st}(x) := \ln \frac{x}{1-x}, \quad \mathrm{sq}(x) := \frac{1}{1+e^{-x}}. \tag{22}$$

Let $\boldsymbol{w}^T = (w_i)_{1 \le i \le m}$ be the weight vector in step k where we assume that $\boldsymbol{w} \in \Omega_m$. Now we rewrite (20) (due to (22)) to yield

$$\begin{aligned} f_k(1, P_1, P_2, \ldots, P_m) &= \left[1 + \exp\left(-\sum_{i=1}^{m} w_i \ln \frac{P_i(1 \mid x^{k-1})}{1 - P_i(1 \mid x^{k-1})}\right)\right]^{-1} \\ &= \left[1 + \prod_{i=1}^{m} \left(\frac{1 - P_i(1 \mid x^{k-1})}{P_i(1 \mid x^{k-1})}\right)^{w_i}\right]^{-1} \\ &= \frac{\prod_{i=1}^{m} P_i(1 \mid x^{k-1})^{w_i}}{\prod_{i=1}^{m} P_i(0 \mid x^{k-1})^{w_i} + \prod_{i=1}^{m} P_i(1 \mid x^{k-1})^{w_i}}, \end{aligned}$$

which matches (11). It is easy to check (via substituting (20) into (19)), that (21) is an instance of iterative gradient descent, where $\alpha_k = \alpha$ is constant in any step and the max-operation is omitted. When α is sufficiently small, the sequence $(\boldsymbol{w}(k))_{k \ge 1}$ converges to some $\boldsymbol{w}_\alpha$ rather than the optimal solution $\boldsymbol{w}^*$. In turn, $\lim_{\alpha \to 0} \boldsymbol{w}_\alpha = \boldsymbol{w}^*$ [1]. A (small) constant step size α thus needs to be determined experimentally.

4 Linear Mixtures

Let us return to the setting of Section 1.1. Instead of encoding x^n with model i and transmitting our choice in $-\log W(i)$ bits, we will not do worse using the mixture distribution

$$P(x^n) := \sum_{i=1}^{m} W(i) P_i(x^n).$$

Since we want to process x^n sequentially we use the distribution (cf. (1))

$$\begin{aligned} \frac{P(x^{k-1}x)}{P(x^{k-1})} &= \frac{\sum_{i=1}^{m} P_i(x^{k-1}x) W(i)}{P(x^{k-1})} \\ &= \sum_{i=1}^{m} \frac{P_i(x^{k-1}) W(i)}{P(x^{k-1})} P_i(x \mid x^{k-1}) \\ &= \sum_{i=1}^{m} W(i \mid x^{k-1}) P_i(x \mid x^{k-1}) \end{aligned} \tag{23}$$

in step k. There is an obvious interpretation for the mixture (23). Suppose that there are m sources and a probabilistic switching mechanism, which selects source i with probability $W(i \mid x^{k-1})$ in step k (we interpret this as the posterior probability of i given x^{k-1}). When a source is selected, it appends a character x (with probability $P_i(x \mid x^{k-1})$) to the sequence x^{k-1} to yield $x^k = x^{k-1}x$. We denote such a source as a *switching source*.

4.1 β-Weighting

We can modify the probability assignment of (23) to yield a linear mixture technique called *β-weighting*, which has its roots in the CTW compression technique and was proposed in [4]. β-weighting is defined by

$$\begin{aligned} f_k(x, P_1, P_2, \ldots, P_m) &:= \sum_{i=1}^{m} \beta_i(k) P_i(x \mid x^{k-1}), \\ \beta_i(k) &:= W(i \mid x^{k-1}) = W(i) \frac{P_i(x^{k-1})}{P(x^{k-1})}. \end{aligned}$$

After the character x_k is known, we can compare $\beta_i(k)$ and $\beta_i(k-1)$ and observe, that

$$\beta_i(k) = \beta_i(k-1)\frac{P_i(x_k \mid x^{k-1})}{f_k(x_k, P_1, P_2, \ldots, P_m)} \text{ and } \beta_i(0) = W(i). \tag{24}$$

4.2 Generic Linear Weighting

With the method of Lagrangian multipliers (see Section 3.1) we can show that (in step k)

$$P := \arg\min_{Q\in\mathcal{P}} \sum_{i=1}^{m} w_i D(P_i \parallel Q), \text{ where } w_i \geq 0, 1 \leq i \leq m, \text{ and } \sum_{i=1}^{m} w_i > 0, \tag{25}$$

yields the linear mixture

$$P(x \mid x^{k-1}) = f_k(x, P_1, P_2, \ldots, P_m) := \sum_{i=1}^{m} w_i' P_i(x \mid x^{k-1}), \text{ where } w_i' := \frac{w_i}{\sum_{i=1}^{m} w_i}.$$

In the setting of the previous section the normalized weights w_i' correspond to the switching probabilities $W(i \mid x^{k-1})$. Thus, the cost function in (25) would be proportional to the expected redundancy of a switching source in step k.

It is important to understand the difference between (3) and (25). In (3) P_i plays the role of a model distribution and we seek an approximate source distribution, which we can use as a model distribution. On the other hand, in (25) P_i plays the role of a source distribution and we seek a model distribution, which matches our *assumptions* on the specific source structure (namely, a switching source). We belief that the assumptions of (3) are inferior to those of (25), hence the geometric mixture is more general.

In analogy to Section 3.2 we look for a weight vector $\boldsymbol{w}^*$, which minimizes the code length of the sequence x^n we want to compress, i.e.,

$$\boldsymbol{w}^* := \arg\min_{\boldsymbol{w}} \sum_{k=1}^{n} -\log \frac{\sum_{i=1}^{m} w_i P_i(x_k \mid x^{k-1})}{\sum_{i=1}^{m} w_i}. \tag{26}$$

First we analyse the convexity properties of (26). W.l.o.g. we assume that $\boldsymbol{w}^T = (w_i)_{1\leq i\leq m}$ is an element of Ω_m. The convexity properties of (26) follow from the analysis of a single term of the sum, which is proportional to

$$l(\boldsymbol{w}) := -\ln \frac{\boldsymbol{w}^T \boldsymbol{P}(x_k)}{\boldsymbol{w}^T \mathbf{1}_m} \overset{\boldsymbol{w}\in\Omega_m}{=} \ln \frac{1}{\boldsymbol{w}^T \boldsymbol{P}(x_k)}, \text{ where } \boldsymbol{P}(x_k)^T := (P_i(x_k \mid x^{k-1}))_{1\leq i\leq m}.$$

The Hessian of $l(\boldsymbol{w})$ is positive definite, since

$$\boldsymbol{v}^T \nabla^2 l(\boldsymbol{w}) \boldsymbol{v} = \boldsymbol{v}^T \frac{\boldsymbol{P}(x_k)\boldsymbol{P}(x_k)^T}{(\boldsymbol{w}^T \boldsymbol{P}(x_k))^2} \boldsymbol{v} = \left(\frac{\boldsymbol{v}^T \boldsymbol{P}(x_k)}{\boldsymbol{w}^T \boldsymbol{P}(x_k)}\right)^2 > 0$$

holds for $\boldsymbol{v} \neq \mathbf{0},\ \boldsymbol{v} \in \mathbb{R}^m$. We conclude that the problem (26) is strictly convex. Thus, there exists a single global minimizer $\boldsymbol{w}^* \in \Omega_m$. As in Section 3.2 we can obtain a weight update rule via iterative gradient descent

$$\boldsymbol{w}(k) := \max\left\{\varepsilon \mathbf{1}_m,\ \boldsymbol{w}(k-1) + \alpha_k \frac{\boldsymbol{P}(x_k) - f_k(x_k, P_1, P_2, \ldots, P_m)\cdot \mathbf{1}_m}{f_k(x_k, P_1, P_2, \ldots, P_m)\cdot \boldsymbol{w}^T \mathbf{1}_m}\right\}, \tag{27}$$

where $\boldsymbol{w}(0)^T := 1/m \cdot \mathbf{1}_m$ and ε is a small positive constant. It is interesting to note, that when we replace α_k with the matrix $\mathrm{diag}(\boldsymbol{w}(k-1))$ and omit the max-operation, (27) turns into β-weighting (cf. (24)) and $\boldsymbol{w}(k) \in \Omega_m, k \geq 0$.

5 Experiments

In this section we compare the performance of a geometric mixture (`GEO`), a generic linear mixture (`LIN`) and β-weighting (`BETA`) on the files of the well-known Calgary Corpus. We have implemented the weighting techniques for a binary alphabet. To process non-binary symbols (here, bytes) we employ an alphabet decomposition. Every symbol $x_k \in \mathcal{X}$ is processed in $N = \lceil \log |\mathcal{X}| \rceil$ intermediate steps, for details see, e.g., [6]. To ensure a fair comparison, the set of models is the same for any mixture method: There are seven finite-order context models (the probability estimations are conditioned on order-0 to order-6 contexts). The eighth model is a *match model*. In step k it searches the longest matching substring x_{k-L}^{k-1} of length $L \geq 7$ in x^{k-2}. In the case of a match it predicts the symbol (here, each bit in the N intermediate steps), which succeeds the matching substring with probability $1 - 1/L$, otherwise each symbol receives the probability $1/|\mathcal{X}|$.

For each mixture technique we select a weight vector $\boldsymbol{w}$ based on an order-1 context and on the match length L (determined by the match model in every step k). Initially any weight vector is initialized to $1/m \cdot \mathbf{1}_m$. After a weight update we ensure that $\boldsymbol{w} \geq \varepsilon \cdot \mathbf{1}_m$ (we set $\varepsilon = 2^{-30}$) and $\boldsymbol{w}^T \mathbf{1}_m = 1$. For β-weighting we can confirm the observation made in [4]: The weights must be bounded considerably away from zero, i.e., $\beta_i \geq \varepsilon$ (we set $\varepsilon = 2^{-8}$). A weight update based on iterative gradient descent requires a step size α_k. We set $\alpha_k = 1/16$ (`GEO`) or $\alpha_k = 1/32$ (`LIN`), respectively. The step size (for `GEO` and `LIN`) and ε (for `BETA`) were determined experimentally for maximum compression. We did not notice significant changes in compression, when the step size was sufficiently small (in the scale of 10^{-2}).

Table 1 summarizes our experimental results. `GEO` outperforms `LIN` and `BETA` in almost every case, expect for the file *obj1*, where the compression is roughly 2% worse than `LIN` and `BETA`. On average `LIN` compresses about 2% and `BETA` compresses about 3.6% worse than `GEO`, respectively. When we compare `LIN` and `BETA` we see that `BETA` produces worse compression in every case, 1.5% on average. Summarizing we may say that `GEO` works better than `LIN`. In our experiments `BETA` is inferior to the other weighting techniques.

6 Conclusion

In this paper we introduced geometric weighting as a new technique for computing mixtures in statistical data compression. In addition we introduced a new generic linear weighting strategy. We explain which assumptions the weighting techniques are based on. Furthermore, our results reveal that PAQ is an instance of geometric weighting for a binary alphabet. All of the presented mixture techniques rely on weight vectors. It turns out that in any of the two cases the weight estimation is a good-natured problem since it is strictly convex. An experimental study indicates that geometric weighting is superior to linear weighting (for a binary alphabet).

For future research it would be interesting to obtain statements about the situations where geometric weighting outperforms linear weighting (and vice-versa). Another topic is how to select a fixed number of submodels for maximum compression. This leads to the optimization of model and mixture parameters (and to the question, whether or not, the optimization problem remains convex). Such a question is very natural, since we wish to maximize the compression with limited resources (CPU and RAM). Combining multiple models in data compression is highly successful in practice, but more research in this area is needed.

Acknowledgment. The author would like to thank Martin Dietzfelbinger, Michael Rink, Martin Aumueller and the anonymous reviewers for helpful comments and corrections.

Table 1: Compression rates in bpc on the Calgary Corpus for geometric- (`GEO`), generic linear- (`LIN`) and β-weighting (`BETA`), best results are typeset boldface.

File	`GEO`	`LIN`	`BETA`
bib	**1.816**	1.890	1.907
book1	**2.212**	2.304	2.313
book2	**1.864**	1.943	1.965
geo	**4.407**	4.423	4.501
news	**2.286**	2.347	2.412
obj1	3.672	**3.603**	3.610
obj2	**2.224**	2.240	2.298
paper1	**2.274**	2.327	2.343
paper2	**2.220**	2.288	2.310
pic	**0.813**	0.871	0.922
progc	**2.276**	2.327	2.361
progl	**1.558**	1.607	1.651
progp	**1.610**	1.638	1.669
trans	**1.384**	1.430	1.453
Average	**2.187**	2.231	2.265

References

[1] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 2nd edition, 1999.

[2] Suzanne Bunton. *On-Line Stochastic Processes in Data Compression.* PhD thesis, University of Washington, 1996.

[3] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley-Interscience, 2nd edition, 2006.

[4] Manfred Kufleitner, Edgar Binder, and Alexander Fries. Combining Models in Data Compression. In *Proc. Symposium on Information Theory in the Benelux*, volume 30, pages 135–142, 2009.

[5] Matthew Mahoney. Adaptive Weighing of Context Models for Lossless Data Compression. Technical report, Florida Tech., Melbourne, USA, 2005.

[6] Christopher Mattern. Combining Non-stationary Prediction, Optimization and Mixing for Data Compression. In *Proc. First International Conference on Data Compression, Communications and Processing*, volume 1, pages 29–37, 2011.

[7] Neri Merhav and Meir Feder. Universal prediction. *IEEE Transactions on Information Theory*, 44:2124–2147, 1998.

[8] David Salomon and Giovanni Motta. *Handbook of Data Compression.* Springer, 1st edition, 2010.

[9] Dimitry Shkarin. PPM: one step to practicality. In *Proc. Data Compression Conference*, volume 12, pages 202–211, 2002.

[10] F. Willems. The context-tree weighting method: extensions. *IEEE Transactions on Information Theory*, 44:792 –798, 1998.

[11] F. Willems, Yuri M. Shtarkov, and T. J. Tjalkens. The context-tree weighting method: basic properties. *IEEE Transactions on Information Theory*, 41:653–664, 1995.