

DEPARTMENT OF COMPUTER SCIENCE  
SERIES OF PUBLICATIONS C  
REPORT C-2002-68



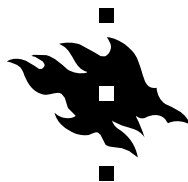
---

Microscopy image analysis of bread  
using machine learning methods

---



J. T. Lindgren and Juho Rousu



UNIVERSITY OF HELSINKI  
FINLAND

# Microscopy image analysis of bread using machine learning methods

J. T. Lindgren and Juho Rousu

Department of Computer Science  
P.O. Box 26, FIN-00014 University of Helsinki, Finland  
{jtlindgr,Juho.Rousu}@cs.Helsinki.FI

Technical report, Series of Publications C, Report C-2002-68  
Helsinki, December 2002, 13 pages

## Abstract

We studied the link between the bread microstructure and sensory properties using machine learning methods. In the first phase, we extracted starch gelatinization degree and protein mesh properties from microscopy images. Color histograms and Haralick features were found to be suitable feature extraction techniques. In both cases, the prediction error of an induced model tree was lower than the standard deviation between independent predictions given by domain experts. Combining models by boosting gave small gains in accuracy.

In the second phase, we used various analysis-originating attributes to predict bread sensory properties. Best single attribute predictors and best pairs are reported.

## Computing Reviews (1998) Categories and Subject Descriptors:

I.2.6 Learning  
I.4.7 Feature Measurement

## General Terms:

Algorithms, Experimentation

## Additional Key Words and Phrases:

visual learning, model trees, boosting, support vector machines

## Distribution:

Karin Autio,  
Tessa Kuuva,  
Liisa Lähteenmäki,  
Kaisa Poutanen,  
Katariina Roininen, and  
Marjatta Salmenkallio-Marttila, VTT Biotechnology.  
J. T. Lindgren,  
Juho Rousu, and  
Esko Ukkonen, University of Helsinki, Department of Computer Science.

## 1 Introduction

In food engineering, the sensory properties—how food looks, feels and tastes—are key criteria to consider. In order to improve these in a controlled manner, one needs to have knowledge about the effect of food microstructure to sensory characteristics.

Linking microstructure properties to sensory characteristics has the practical difficulty that analysis of microscopy images is often cumbersome and requires expertise. Although a domain expert can give semi-quantitative assessments of the key properties of the sample just by looking at the image, quantitative analysis requires the use of image-analysis software of some kind. These applications work well if the properties to be extracted are simple such as the area of color or the size-distribution of well-formed particles. When the property to be analyzed is more complex, the feature extraction becomes prohibitively laborious. Machine learning methods might aid in tasks of this kind. By learning from examples a model mimicking the domain expert, one can ease the work burden.

We studied the link between the bread microstructure and sensory properties using machine learning methods in a two-phase study. In the first phase, we extracted starch gelatinization degree and protein mesh properties from 510 microscopy images, each independently classified by domain experts. We tested feature extraction methods including color histograms, edge detection, Fourier transform, Haralick features and other techniques. To learn a model several machine learning methods were tried out, including model trees, support vector machines and boosting algorithms.

In the second phase, the starch gelatinization and protein mesh predictors were combined to other instrumental analyses, including rheological measurements, to predict sensory properties of bread.

## 2 Materials and methods

### 2.1 Data

Our dataset originates from 12 microscope images of bread baked with 6 different baking recipes, two images per recipe. Of these, domain experts have selected and cropped 510 small  $150 \times 150$  full-color image patches to use in model induction. Figure 1 shows two patches. For each example, the domain experts have provided two sets of numeric labels in range  $[1, 5]$ . For starch and protein problems, predictions of two and three experts were given, respectively. For each image, we calculate the mean prediction of the experts, and use that as actual instance label.

### 2.2 Model validation

To avoid presenting accuracy results on data that has been available to the learning algorithm during training, we allocated 100 random instances as a separate test set, and used 410 instances for training and tuning. We did the tuning by performing 10-fold cross-validations on the training set, and chose

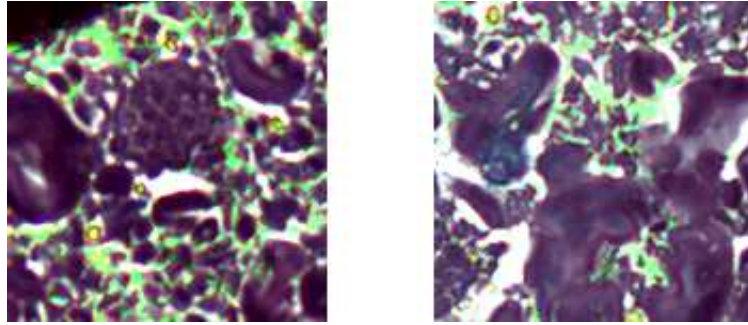


Figure 1: Two image patches (#211 and #98). The one on the left should have smaller level of starch gelatinization.

those models as best which had the smallest cross-validation error, with small consideration towards model complexity.

For second phase data, we have only 6 examples (one for each bread) and various characteristics (features) extracted from them by traditional image analysis techniques or other non-algorithmic conventional methods.

### 2.3 Preprocessing methods

It is well known that general machine learning algorithms do not perform well on raw image data. In our case, the raw data dimension is  $150 \times 150 \times 3 = 67500$  features, which is over a hundredfold times the size of our training set of 410 examples. In general, high image dimensionality in combination with much smaller number of training examples causes the feature space to be sparsely populated. This makes it easy for the learning algorithms to find hypotheses that are consistent with the training data, but fail totally on any provided test set. This is called *overfitting*. In addition, if the image label doesn't depend on orientation, scale or small shifts of the image content, very complex attribute interactions and large training sets are required to find and model these properties accurately.

To overcome this, we attempt to apply some well-known feature extraction techniques to reduce the problem dimension, and use the resulting attribute vectors as input for state-of-the-art learning methods.

#### 2.3.1 Color histograms

Histograms are one of the few established methods in image classification and content based image retrieval. They are simple discretized measures of the color distribution in an image.

We create a separate histogram for each color channel, dividing its intensity interval  $[0, 255]$  to 32, 64 or 256 bins. All bins are of equal width. Each bin is associated with an intensity interval, its value denoting the percentage of that intensity in the bins channel.<sup>1</sup> Thus, for example, for 32 bin histograms we get

<sup>1</sup>Using a popular method of histogram equalization prior to other extraction methods –

$3 * 32 = 96$  attributes – a considerable reduction of the raw dimension 67500. Histograms are invariant to scale, rotation and shifts.

### 2.3.2 Edge detection

Edge information has often been used to measure texture characteristics. We experiment with a simple statistical measure of images response to different edge detection algorithms. For R,G,B channels and grayscale images separately, we apply all the edge detection routines found in Matlab (*sobel*, *prewitt*, *log*, *zerocross*, *Roberts* and *canny*) and calculate the images response to each. This gives  $4 * 6 = 24$  attributes.

Intuitively, this ad-hoc statistic captures the roughness of the images, where we view the edge detection routines as differently biased experts. It is interesting to note that even these features allow better-than-guess prediction accuracies. The invariance properties towards scaling, rotation and shifts depend on the detection method.

### 2.3.3 Fourier transform

A common use of Fourier transforms is to find the frequency components of a signal buried in a noisy time domain signal. In the image analysis application, we take the columns  $\{1, \dots, 150\}$  as the 'time' and the color intensity as the 'signal', each color channel separately. The frequency histograms of the rows are summed to have an aggregate spectrum for the image for each color channel.

The Fourier transform then answers the question: which periods of color intensity alternations exist along the row. If the color alternates slowly, we will end up with high intensity in low frequency bands and otherwise we expect to have high intensity in the high frequency bands.

Fourier transform is not invariant to scale, but is robust to shifts. The described way of applying FFT is probably not very robust towards rotations.

### 2.3.4 Haralick texture features

In 1973 Haralick et al. [HSD73] presented a set of features that has later been successful in various experiments on texture classification. Using image intensity values as matrix indices, we compute a so-called co-occurrence matrix  $P$  for some orientation  $\theta$  and distance  $d$ . For two pixels having distance  $d$  and orientation  $\theta$ , with intensities  $i$  and  $j$  respectively, we increment  $P_{ij}$ . In total 14 statistical measures are computed on  $P$ , such as angular second momentum, contrast and entropy.

We compute Haralick features using typical orientations  $\theta \in \{0, 45, 90, 135\}$ . This gives  $14 * (4 + 2) = 84$  attributes for each channel. The additional set of  $(2 * 14)$  features comes from including maximums and averages of each measure over all the orientations. Here distance  $d \in \{1, 2\}$ , giving two separate datasets.

Haralick features are not invariant to scale, but should be robust to shifts and somewhat robust to orientations (through selections of  $\theta$ ).

---

often attempted for alleviating problems related to varying brightness conditions – generally made the results worse in our setting.

### 2.3.5 Other methods

We also attempted several other approaches for feature extraction, in various degrees of complexity. These methods included Independent Component Analysis (ICA) [HKO01], Zernike moments [KH90], various uses of filters, and finally augmenting several featuresets with their basic statistics. In our studies, the application of these rather general ideas didn't help the learning methods to induce more accurate models.

## 2.4 Machine learning algorithms

We experiment with two slightly different schools of methods (Table 1), those of native regression model learners (AddM5P, AddSt, LinR), and methods of learning binary classifiers (AdaC4.5, LbSt20, SMO), via transformation from regression to classification problem (see below). The model families used by the algorithms were restricted to two regression models, model trees and linear regression, and to two classification models, decision trees and support vector machines. Four of the selected algorithms (AddM5P, AddSt, AdaC4.5, LbSt20) belong to the family of boosting algorithms [FS96, FHT98] that iteratively construct a carefully weighted linear combination of the base models.

Transforming a numeric prediction problem to multiple binary classifications is a problem in itself. Here we apply a simple mechanism of Frank and Hall [FH01]. First, the numeric class interval is discretized to a chosen number of bins. For each numeric value  $v$  denoting the separator between two bins, a new classifier is trained using examples having label value less than  $v$  as negative examples and the rest as positive. The final prediction is the most likely bin. The key property of this transformation is its ability to preserve order in the discretized class.

### 2.4.1 Tree-based approaches

C4.5 [Qui93] learns decision (classification) trees by recursively partitioning the input space so that class entropy in each partition is minimized. The split-points are chosen greedily. Each leaf of the tree (denoting a subset of the feature space) predicts the majority label of training examples ending up in that leaf (region).

An M5P model tree [Qui92] is a decision tree extended for regression. The model tree has linear regression models at its leaves. This time the feature space partitioning attempts to minimize the error of the linear predictors on the training data.

Decision stumps are one-level decision trees, here generated by C4.5. Decision stumps can be used both for regression and classification.

### 2.4.2 Support vector machines

Support Vector Machines (SVMs) [Vap95] try to separate two classes of examples by a hyperplane in a chosen feature space. The feature space is induced by choosing a *kernel function* that decides the dimensionality of the feature space. The feature space may be very high-dimensional, depending on the kernel. In

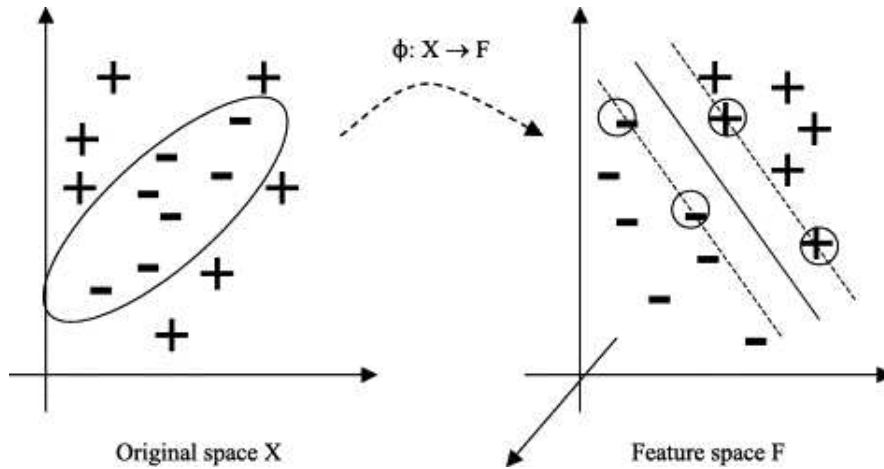


Figure 2: SVMs first map the data into a potentially high-dimensional feature space and search there for a separating hyperplane with a high margin. Support vectors (encircled) are points lying closest to the hyperplane.

this study, however, we used the linear kernel, meaning that the dimensionality of the feature space was the same as the number of variables in the data.

The hyperplane to separate the classes is usually chosen so that its distance to examples of the two classes is maximized. That distance is called the *margin*. As the selected hyperplane is defined by the nearest examples to it, its natural to call those examples *support vectors*. These points have to be retained for classification.<sup>2</sup> Figure 2 illustrates the SVM idea, where  $\Phi$  is a mapping from the input space  $X$  to a chosen feature space  $F$ .

In this study we used Sequential Minimal Optimization (SMO) [Pla99] to learn SVMs for binary classification.

### 2.4.3 Boosting

A practical recent development is that of boosting [FS96], which has been able to enhance prediction accuracy of various learning algorithms on real-life datasets. Basically, boosting works by iteratively training new models, on each training round emphasizing such examples that the previous model had problems with, or that lie close to the decision surface. A linear combination of the built classifiers is used as a final classifier. The term 'boosting' is justified by theoretical results stating that a weak learning algorithm, capable of only slightly better predictions than random guessing, can be transformed into a strong algorithm by combining the weak models in a particular way.

In this study we examine if AdaBoostM1 [FS96], Additive Regression [Fri99] or LogitBoost [FHT98] can enhance the previously listed learning methods on our problems.

<sup>2</sup>In this respect, SVMs may be thought of as distant relatives to nearest neighbour classifiers, which also store training examples for prediction.

AddM5P	Additive Regression using M5P, shrinkage 0.5
AddSt	Additive Regression using Decision Stumps, shrinkage 0.5
AdaC4.5	AdaBoostM1 using C4.5 and 10 iterations
LbSt20	LogitBoost using Decision Stumps and 20 iterations
LinR	Linear Regression with attribute selection
SMO	An SVM with linear kernel and $C = 1$ .
edges	Edge statistics featureset.
fft	Fast Fourier Transform featureset.
h1	Haralick featureset with $d = 1$ .
h1h2	Haralick featureset with $d \in \{1, 2\}$ .
h1hg	Haralick featureset with $d = 1$ and the same calculated on grayscale image.
hg	Haralick featureset calculated on grayscale image.
hstX	X-bin color histogram featureset.
hst64-X	A combination featureset.
hsvhst32	32-bin color histogram featureset calculated in HSV space.
miscfeat	A small set of primitive statistic features.
zernike	Zernike moments featureset.

Table 1: The machine learning algorithms (top) and the feature extraction methods (bottom) used in the tests.

### 3 Results

In the following, we report the model accuracy by its Root Mean Squared Error (RMSE). The accuracies were measured using 10-fold cross-validation on our train/tune set of 410 examples. Statistical significance was measured by paired  $t$ -test at confidence level 0.05 where reported. Entries *Geom* in the tables denote geometric means of the respective rows and columns. Based on this information we choose representative models for both of our problems and report their accuracy on a separate test set of 100 instances as row *Final* in the tables.

Used abbreviations are explained in Table 1. It should be noted that all parameters of the algorithms were selected only roughly. Small experiments on parameter tuning did not hint at possibility to get significantly more accurate classifiers by more extensive tuning.

#### 3.1 Prediction of starch gelatinization

Table 2 presents the results for the starch gelatinization problem. It can be seen that already M5P – as arguably the simplest of these methods after linear regression – achieves a competitive error rate of 0.35 on the even more simple 32 bin histogram featureset *hst32*. The various boosting methods were not able to significantly improve on this error rate, and SMO mostly did worse<sup>3</sup>.

A natural question regards the quality of the achieved error rates. A trivial baseline model is one that always predicts the training set mean. Using this method, an RMSE accuracy of 0.66 can be achieved on the starch problem.

<sup>3</sup>Our additional experiments did show that this was mostly due to lack of parameter and kernel tuning for SMO, but even after reasonable amount of tuning we were not able to make SMO perform better than the other algorithms, and did not pursue that direction further.

Table 2: Achieved RMSE rates on the starch problem.

Featureset	LinR	M5P	AddSt	LbSt20	AdaC4.5	SMO	AddM5P	Geom
edges	0.42	0.40 *	0.45 v	0.46 v	0.46 v	0.50 v	0.39 *	0.44
fft	0.65	0.69	0.53 *	0.56 *	0.73 v	0.55 *	0.63	0.62
h1	<b>0.37</b>	0.35	0.36	0.37	<b>0.36</b>	0.40	0.34	<b>0.36</b>
h1h2	0.38	0.35	0.35	0.37	<b>0.36</b>	<b>0.38</b>	0.34 *	<b>0.36</b>
h1hg	<b>0.37</b>	0.36	0.35	0.37	<b>0.36</b>	0.40	0.35 *	0.37
hg	<b>0.37</b>	0.36	0.37	0.39	0.38	0.42 v	0.35 *	0.38
hst256	0.65	0.43 *	0.36 *	0.38 *	0.39 *	0.39 *	0.39 *	0.42
hst32	0.40	0.35 *	0.35 *	0.37	0.38 *	0.41	0.36 *	0.37
hst64-misc	0.48	0.39 *	0.35 *	0.37 *	0.38 *	0.41 *	0.38 *	0.40
hst64	0.49	0.40	0.35 *	0.39 *	0.38 *	0.43 *	0.38 *	0.40
hst64-edges	0.48	0.36 *	<b>0.33</b> *	0.38 *	0.37 *	0.39 *	0.36 *	0.38
hst64-h1h2	0.49	<b>0.34</b> *	0.34 *	<b>0.36</b> *	<b>0.36</b> *	0.39 *	<b>0.33</b> *	0.37
hsvhst32	0.39	0.37	0.34 *	<b>0.36</b>	0.37 *	0.41	0.35 *	0.37
miscfeat	0.44	0.44	0.43	0.47	0.44	0.51 v	0.43	0.45
zernike	0.53	0.55	0.50 *	0.51 *	0.51	0.62 v	0.52	0.53
Geom	0.45	0.40	<b>0.38</b>	0.40	0.41	0.44	0.39	
Final	-	0.35	-	-	-	-	-	

(\* / v) statistically significant gain/loss w.r.t Linear Regression.

Table 3: Achieved RMSE rates on the protein problem.

Featureset	LinR	M5P	AddSt	LbSt20	AdaC4.5	SMO	AddM5P	Geom
edges	0.39	0.43 v	0.45 v	0.49 v	0.38	0.46 v	0.36 *	0.42
fft	0.84	0.79	0.67 *	0.69 *	0.84	0.75 *	0.72 *	0.75
h1	0.35	0.35	0.33	<b>0.35</b> v	0.33	0.39 v	0.32 *	0.34
h1h2	<b>0.33</b>	<b>0.32</b>	<b>0.32</b>	0.37 v	0.33	0.37 v	<b>0.29</b> *	<b>0.33</b>
hg	0.32	0.35	0.33	0.35 v	0.33	0.39 v	0.32	0.37
hst256	0.78	0.38 *	0.36 *	0.40 *	0.37 *	0.42 *	0.36 *	0.42
hst32	0.40	0.42	0.36 *	0.42	0.35 *	0.39	0.35 *	0.38
hst64-edges	0.54	0.34 *	<b>0.32</b> *	0.37 *	<b>0.31</b> *	0.34 *	0.32 *	0.36
hst64	0.54	0.40 *	0.37 *	0.39 *	0.34 *	0.42 *	0.37 *	0.40
hst64-h1h2	0.55	0.33 *	0.33 *	0.36 *	<b>0.31</b> *	0.36 *	0.31 *	0.36
hst64-stats	0.60	0.38 *	0.36 *	0.41 *	0.33 *	0.41 *	0.35 *	0.40
hsvhst32	0.40	0.33 *	0.33 *	0.38	0.34 *	<b>0.34</b> *	0.33 *	0.35
miscfeat	0.47	0.42 *	0.46	0.49	0.53 v	0.53 v	0.43 *	0.47
zernike	0.60	0.60	0.59	0.62	0.59	0.69 v	0.59	0.61
Geom	0.49	0.40	0.39	0.43	0.39	0.44	<b>0.38</b>	
Final	-	-	-	-	-	-	0.27	

(\* / v) statistically significant gain/loss w.r.t Linear Regression.

We also calculated another simple heuristic baseline by measuring the *expert error* against the training labels, that is, the deviation of a single expert from the mean prediction of the experts on the sample. For the starch problem, expert RMSE accuracy by this measure is 0.38, the mean standard deviation of their predictions being 0.41. This implies that in some sense our built model is competitive with a domain expert.

After parameter tuning and crossvalidation, we selected M5P with 32 bin histogram features as the representative combination for this problem. The last row of table 2 denotes the RMSE 0.35 this combination achieved on the separate test set of 100 instances.

### 3.2 Prediction of protein mesh properties

Table 3 presents results on the protein problem. Here M5P performs quite well with RMSE of 0.32 on Haralick featureset *h1h2*. This time boosting methods are slightly more helpful. Especially Additive Regression with M5P achieves the overall best error rate of 0.29. In comparison, the baseline method predicting the training set mean achieves error of 0.78, and the error of an average expert (in the sense described earlier) is 0.33, or 0.35 as standard deviation. Thus our models seem to remain competitive also in this problem.

Again, last row of table 3 denotes the RMSE 0.27 of the chosen model on our separate test set. For the protein problem we selected Additive Regression using M5P, run on combined Haralick featureset *h1h2*, as the method of choice.

### 3.3 Convergence rates

An important practical question regards the learning rate of the used algorithms on the used featureset. That is, it would be interesting to know whether more accurate models could be built by increasing training set size.

We give intuition of this by learning rate plots, which are calculated as follows. The models are induced in rounds. On each round, the training set size is incremented by 10 and a new model is built. The model is then evaluated on our fixed test set of 100 images. This procedure iterates until all 410 examples are used in the training set. To get a smoother plot, the results are averaged over 10 such runs, each time permuting the order in which examples are added.

Figure 3 presents the learning rate behaviour of M5P-based methods on the starch problem, when histograms and Haralick features are used, together with a similar curve for linear regression. The plot shows that the expert prediction level is reached using the Haralick featureset already on 150 examples, whereas the more simple 32 bin histogram set requires over 350 examples. As the form of these figures is inversely exponential, it seems unlikely that additional training examples could significantly enhance prediction accuracy on the starch problem.

Learning rates for the protein problem are given in figure 4. Here again the Haralick featureset seems better, but now 300 examples are required to reach the expert level. Interestingly the error rate starts diminishing again when we approach 410 examples. Whether this is an artifact of our fixed random choice of test set, is not known.

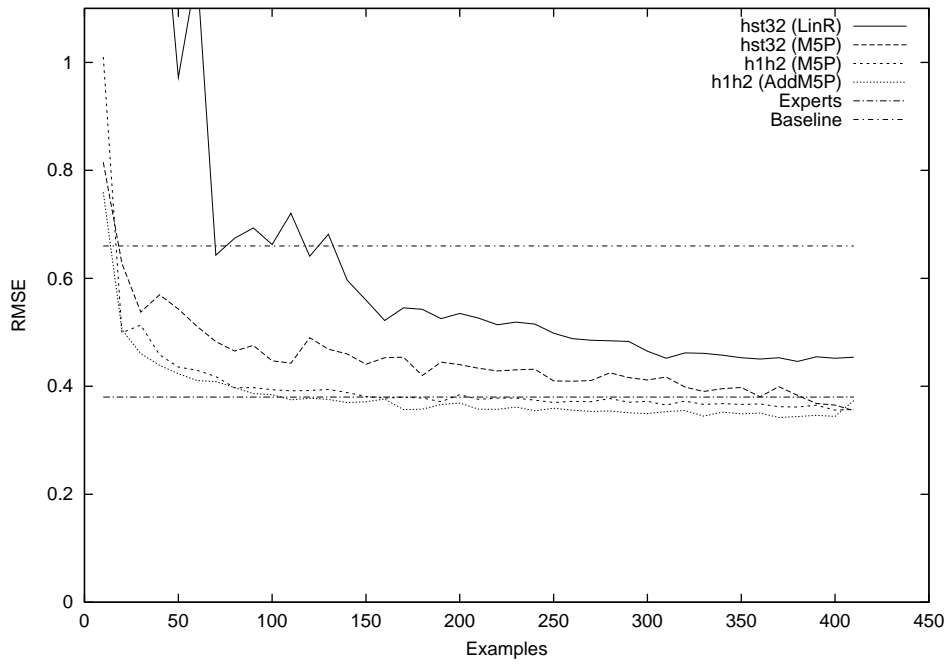


Figure 3: Learning rates on the starch problem.

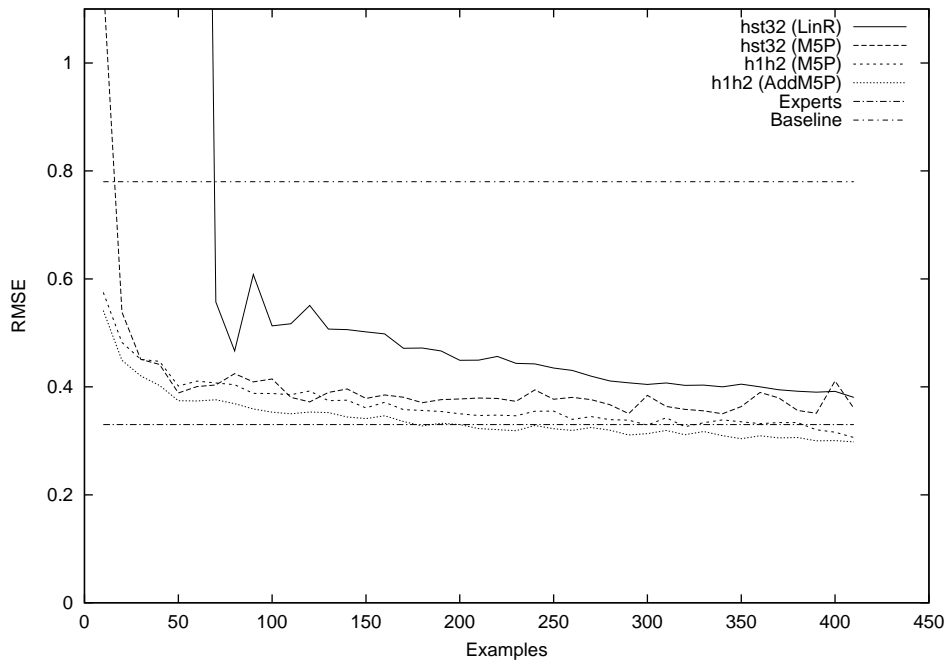


Figure 4: Learning rates on the protein problem.

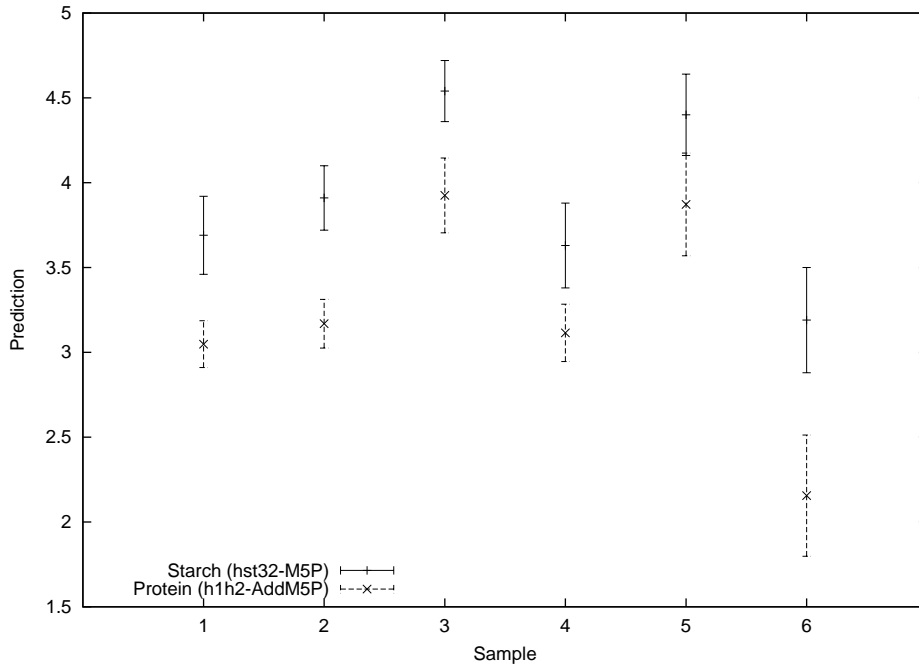


Figure 5: Prediction means and standard deviations, grouped by sample.

The two convergence plots show exhibit the same overall behaviour: the error rate very quickly drops to a level 0.1 above the expert error rate. After that, the convergence slows down considerably, suggesting that the expert error rate is almost—but not quite—a lower bound to the generalization error.

### 3.4 Some notes on the dataset

The baseline rates achieved by predicting the training set mean shows that the example labels are concentrated among few typical values on both problems. Figure 5 represents the training set means, as predicted by our chosen models, with standard deviations as vertical error bars. The image patches from each sample are quite homogenous.

The previous results would imply that our models generalize well, but actually this might not be the case. The reason is that cross-validation or evaluating on a disjoint test set doesn't really measure our generalization capability, as even though our training and test sets are disjoint, they both originate from the same  $6 \times 2$  bread samples. To reliably evaluate the created models, the test set would have to be independent from the training set, the images originating from a separate, fresh baking process. It also seems clear that 12 bread images can not be a representable set of the whole imaginable *bread space*, and it is not likely that the models could generalize to such bread types that were not represented during the training phase.

Another reason to expand the set of biological samples is the fact that the natural uncertainty related to sample handling and the measurement errors can be more reliably taken into account.

Characteristic	Best single pred.	Best pred. pair
S-Size	I-Chewiness	<b>I-Chewiness, VL-Prot</b>
S-Aerated (S)	ReoMax	<b>I-Gumminess, VL-Starch</b>
S-Aerated (U)	ReoMax	<b>VL-Starch, ReoMax</b>
S-Springy	VL-Starch	<b>SpecVol, ReoMax</b>
S-Crumbly	ReoMax	<b>I-Gumminess, VL-Prot</b>
S-Dense	ReoMax	<b>I-Chewiness, VL-Prot</b>
S-Soft	I-Chewiness	<b>SpecVol, VL-Prot</b>
S-Wet	VL-Starch	<b>SpecVol, VL-Prot</b>
S-Hardness	SpecVol	<b>SpecVol, ReoMax</b>
S-Crispness	<b>VL-Prot</b>	SpecVol, ReoMax

Table 4: Best predictors for sensory characteristics. The better of the single-variable and two-variable predictors is depicted by boldface font.

For the similar experiments in the future we propose *statistical experimental design* to be applied as to obtain a maximally informative and balanced dataset. We speculate that the total number of the 'small images' does not need to be raised significantly, instead, distributing the images more evenly in the bread space—via increased number of bread samples—should provide enough information to induce more reliable models than in the present study.

### 3.5 Prediction of sensory properties

In the second phase we examined how various bread sensory characteristics can be predicted by analysis-originating features. The predicted sensory characteristics are listed in the first column of table 4. As the set of predictors we used various instrumental analyses: *IA-Prot*, *IA-Starch*, *I-Hardness*, *I-Cohesiveness*, *I-Gumminess*, *I-Chewiness* and *Specific Volume* and *ReoMax* (effect of heating on storage modulus), obtained from the domain experts.

We augmented this set by the predictors *VL-Starch*, *VL-Prot*—corresponding to the starch gelatinization degree and protein network structure features, learned from the microscopy images ('VL' denotes 'Visual learning') as described in the previous section.

We used both single predictors and all pairs of two to predict the sensory characteristics by linear regression, measuring the accuracy by leave-one-out cross-validation. Due to the tiny size of our dataset in this case (only 6 examples) we refrain from giving numeric results, but instead report just which combinations were best in predicting which characteristic. These are noted in table 4. A two-variable model was usually the best combination, with the exception of the *S-Crispness*, which was best to be predicted with the *VL-Prot* feature alone.

The visual learning predictors frequently occur in the lists of best variables. In 8 cases of 10 either *VL-Starch* or *VL-Prot* among the best variable combination, *S-hardness* and *S-Springy* being the exceptions. In the latter case, moreover, *VL-Starch* still was the best single predictor.

If three or more predicting attributes were used, the error started to increase

due to overfitting. In any case, the reader is cautioned against drawing too far-reaching conclusions from the results.

## 4 Conclusion

Our results indicate that machine learning may be successfully used in constructing models of starch gelatinization and protein mesh properties using preprocessed and expert-labeled image data. The performance of the models matched to performance of domain experts in both tasks. However, the small number of biological samples ( $6 \times 2$ ) underlying the image data (510 images) limits the possibilities of making strong statements about the true predictive capability of the models.

Color histograms and Haralick texture features were found to be best preprocessing methods of those examined. Both perform well with the model tree inducer M5P working in regression domain. Slightly better accuracies were achieved by applying boosting.

Although the classification models (C4.5, decision stumps and SVM) performed almost as well as M5P in these tasks, the added complexity from the regression-to-classification conversion makes the native regression models the more appealing choice.

If future work on predicting starch gelatinization degree, protein mesh properties or some other attributes from visual images is considered, one of the significant challenges will be collecting representative datasets, with truly independent test sets. It would be interesting to see how the combinations we studied could perform in some more general setting.

In this study we did not rigorously experiment on feature selection (pruning the feature space). For example, not all histogram bins are required for the final prediction, as likewise not all statistics of the Haralick features are always relevant. Feature selection might be worthwhile if more simple models are desired.

The second task in this study, predicting the sensory properties of bread using the features from the visual learning and other instrumental analysis as predictors, showed that the information obtained from the microscopy images correlates with the sensory properties. Measuring the quality of the found dependencies would require a larger collection of data. The application of statistical experiment design is advisable in generation of the data.

## References

- [FH01] E. Frank and M. Hall. A simple approach to ordinal prediction. In *12th European Conference on Machine Learning*, pages 145–156. Springer-Verlag, 2001.
- [FHT98] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Dept. of Statistics, Stanford University, 1998.

- [Fri99] J. Friedman. Stochastic gradient boosting. Technical report, Dept. of Statistics, Stanford University, 1999.
- [FS96] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156, 1996.
- [HKO01] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley and Sons, 2001.
- [HSD73] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, 1973.
- [KH90] A. Khotanzad and Y. H. Hong. Invariant image recognition by zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):489–497, 1990.
- [Pla99] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, 1999.
- [Qui92] J. R. Quinlan. Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, 1992.
- [Qui93] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, 1993.
- [Vap95] V. N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, Heidelberg, DE, 1995.