Fisher Vector image representation

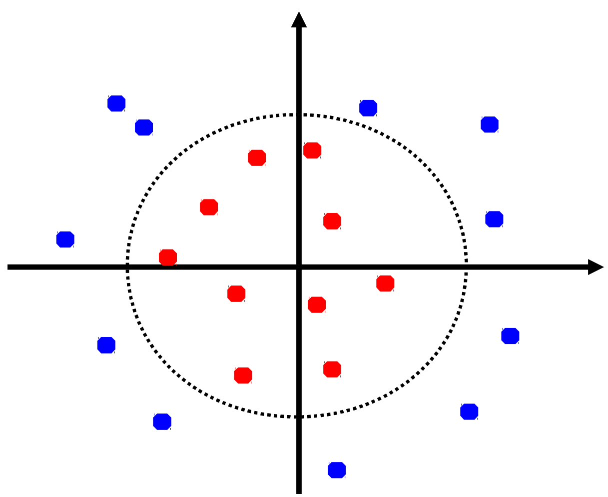
Machine Learning and Object Recognition 2017-2018

Jakob Verbeek

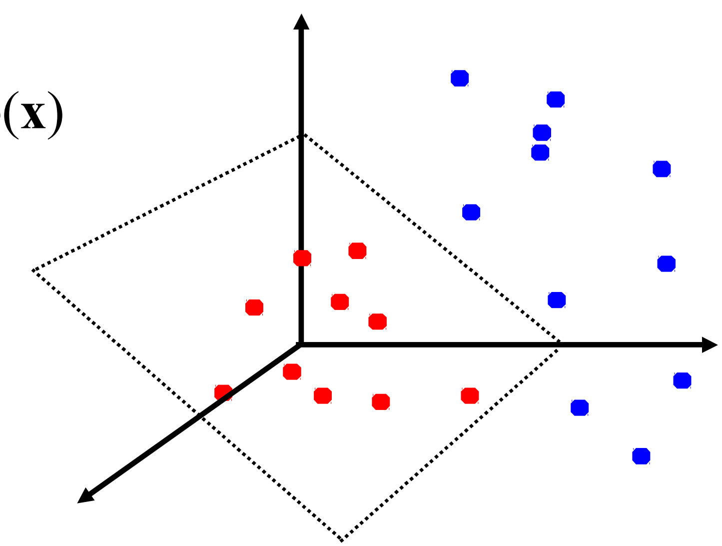
A brief recap on kernel methods

- A way to achieve non-linear classification by using a kernel that computes inner products of data after non-linear transformation.
  - Given the transformation, we can derive the kernel function.
- Conversely, if a kernel is positive definite, it is known to compute a dot-product in a (not necessarily finite dimensional) feature space.
  - Given the kernel, we can determine the feature mapping function.

$$k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$$



$\Phi\text{:}\ \mathbf{x} \rightarrow \varphi(\mathbf{x})$



A brief recap on kernel methods

- So far, we considered starting with data in a vector space, and mapping it into another vector space to facilitate linear classification
- Kernels can also be used to represent non-vectorial data, and to make them amenable to linear classification (or other linear data analysis) techniques
- For example, suppose we want to classify sets of points in a vector space, where the size of each set may vary

$$X = \{x_1, x_2, \ldots, x_N\} \quad \text{with} \quad x_i \in R^d$$

- We can, for example, define a representation of sets by concatenating the mean and variance of the set in each dimension

$$\phi(X) = \begin{pmatrix} \text{mean}(X) \\ \text{var}(X) \end{pmatrix}$$

  - Fixed size representation of sets in 2d dimensions
  - Use kernel to compare different sets:

$$k(X_1, X_2) = \langle \phi(X_1), \phi(X_2) \rangle$$

Fisher kernels

- Motivated by the need to represent variably sized objects in a vector space, such as sequences, sets, trees, graphs, etc., such that they become amenable to be used with linear classifiers, and other data analysis tools

- A generic method to define kernels over arbitrary data types based on statistical model of the items we want to represent

$$p(x;\theta), \quad x \in X, \quad \theta \in R^D$$

- Parameters and/or structure of the model p(x) estimated from data
  - Typically in unsupervised manner
- Automatic data-driven configuration of kernel instead of manual design
  - Kernel typically used for supervised task

[Jaakkola & Haussler, “Exploiting generative models in discriminative classifiers”,In Advances in Neural Information Processing Systems 11, 1998.]

Fisher kernels

- Given a generative data model $p(x;\theta),\ \ x\in X,\ \ \theta\in R^D$
- Data representation with gradient of the data log-likelihood, or “Fisher score”

$$g(x)=\nabla_\theta \ln p(x),$$
$$g(x)\in R^D$$

- Define a kernel over X by taking the scaled inner product between the Fisher score vectors:

$$k(x,y)=g(x)^T F^{-1} g(y)$$

- Where F is the Fisher information matrix F:

$$F=\boldsymbol{E}_{p(x)}\left[g(x)\,g(x)^T\right]$$

- F is positive definite since

$$\alpha^T F\alpha=\boldsymbol{E}_{p(x)}\left[\left(g(x)^T\alpha\right)^2\right]>0$$

Fisher vector

- Since F is positive definite we can decompose its inverse as

$$F^{-1} = L^T L$$

- Therefore, we can write the kernel as

$$k(x_i, x_j) = g(x_i)^T F^{-1} g(x_j) = \phi(x_i)^T \phi(x_j)$$

  - Where phi is known as the **Fisher vector**

$$\phi(x_i) = L g(x_i)$$

- From this explicit finite-dimensional data embedding it follows immediately that the Fisher kernel is a positive-semidefinite

- Since F is covariance of Fisher score, normalization by L makes the Fisher vector have unit covariance matrix under p(x)

Normalization with inverse Fisher information matrix

- Gradient of log-likelihood w.r.t. parameters $g(x) = \nabla_\theta \ln p(x)$
- Fisher information matrix $F_\theta = \int g(x) g(x)^T p(x) dx$
- Normalized Fisher kernel $k(x_1, x_2) = g(x_1)^T F_\theta^{-1} g(x_2)$
  - Renders Fisher kernel invariant for parametrization

- Consider different parametrization given by some invertible function $\lambda = f(\theta)$
- Jacobian matrix relating the parametrizations $[J]_{ij} = \frac{\partial \theta_j}{\partial \lambda_i}$
- Gradient of log-likelihood w.r.t. new parameters, via chainrule

$$h(x) = \nabla_\lambda \ln p(x) = J \nabla_\theta \ln p(x) = J g(x)$$

- Fisher information matrix $F_\lambda = \int h(x) h(x)^T p(x) dx = J F_\theta J^T$
- Normalized Fisher kernel

$$\begin{aligned} h(x_1)^T F_\lambda^{-1} h(x_2) &= g(x_1)^T J^T (J F_\theta J^T)^{-1} J g(x_2) \\ &= g(x_1)^T J^T J^{-T} F_\theta^{-1} J^{-1} J g(x_2) \\ &= g(x_1)^T F_\theta^{-1} g(x_2) \end{aligned}$$

Fisher kernels – relation to generative classification

- Suppose we make use of generative model for classification via Bayes' rule
  - Where x is the data to be classified, and y is the discrete class label

$$p(y|x) = p(x|y)\,p(y)/p(x),$$
$$p(x) = \sum_{k=1}^{K} p(y=k)\,p(x|y=k)$$

and

$$p(x|y) = p(x;\theta_y),$$
$$p(y=k) = \pi_k = \frac{\exp(\alpha_k)}{\sum_{k'=1}^{K} \exp(\alpha_{k'})}$$

- Classification with the Fisher kernel obtained using the marginal distribution p(x) is at least as powerful as classification with Bayes' rule
- This becomes useful when the class conditional models are poorly estimated, either due to bias or variance type of errors
- In practice often used without class-conditional models, but direct generative model for the marginal distribution on X

Fisher kernels – relation to generative classification

- Consider the Fisher score vector with respect to the marginal distribution on X

$$\nabla_\theta \ln p(x) = \frac{1}{p(x)} \nabla_\theta \sum_{k=1}^{K} p(x, y=k)$$

$$= \frac{1}{p(x)} \sum_{k=1}^{K} p(x, y=k) \nabla_\theta \ln p(x, y=k)$$

$$= \sum_{k=1}^{K} p(y=k|x) \left[\nabla_\theta \ln p(y=k) + \nabla_\theta \ln p(x|y=k) \right]$$

- In particular for the alpha that model the class prior probabilities we have

$$\frac{\partial \ln p(x)}{\partial \alpha_k} = p(y=k|x) - \pi_k$$

Fisher kernels – relation to generative classification

$$\frac{\partial \ln p(x)}{\partial \alpha_k} = p(y=k|x) - \pi_k$$

$$g(x) = \nabla_\theta \ln p(x) = \left(\frac{\partial \ln p(x)}{\partial \alpha_1}, \ldots, \frac{\partial \ln p(x)}{\partial \alpha_K}, \ldots \right)$$

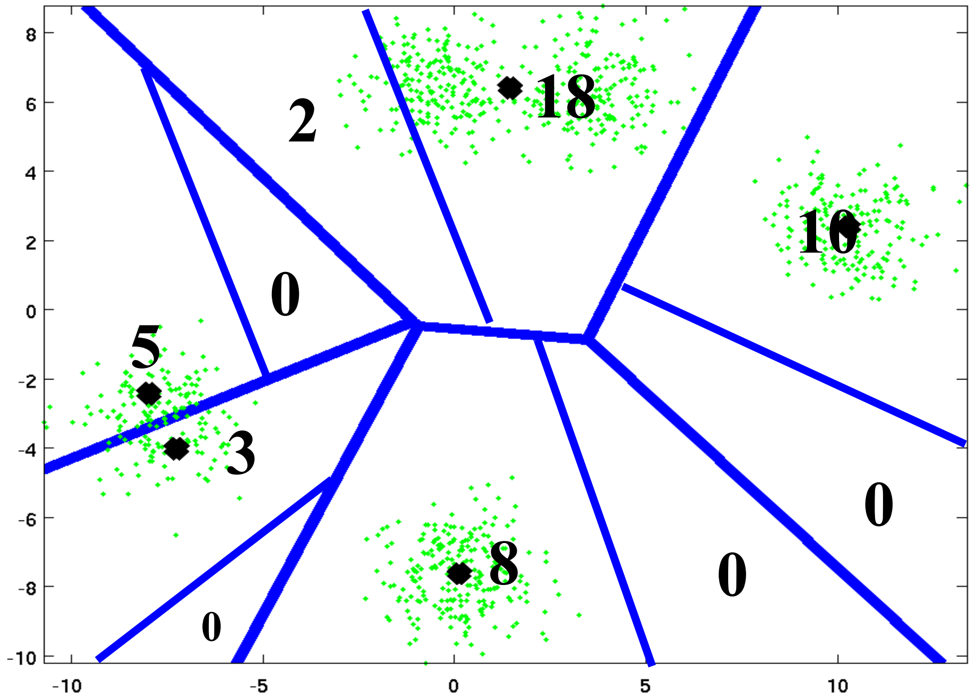
- Consider discriminative multi-class classifier.
- Let the weight vector for the k-th class to be zero, except for the position that corresponds to the alpha of the k-th class where it is one. And let the bias term for the k-th class be equal to the prior probability of that class
- Then $f_k(x) = w_k^T g(x) + b_k = p(y=k|x)$

  and thus $\operatorname{argmax}_k \ f_k(x) = \operatorname{argmax}_k \ p(y=k|x)$
- Thus the Fisher kernel based classifier can implement classification via Bayes' rule, and generalizes it to other classification functions

Fisher vector GMM image representation: Motivation

- Suppose we want to refine a given visual vocabulary to obtain a richer image representation

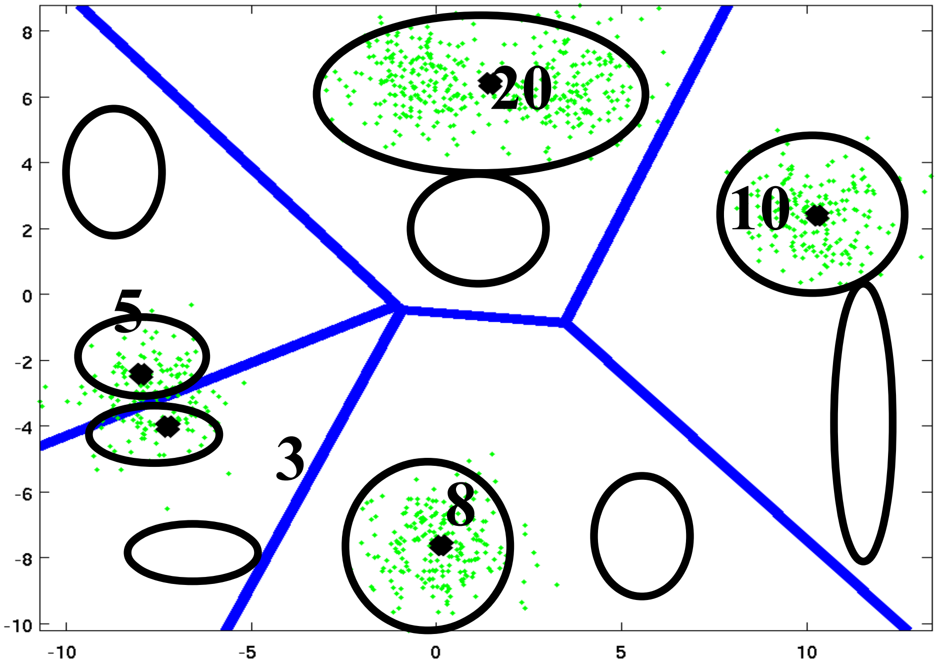
- Bag-of-word histogram stores # patches assigned to each word
  - Need more words to refine the representation
  - But this directly increases the computational cost
  - And leads to many empty bins: redundancy



Fisher vector representation in a nutshell

- Fisher Vector derived from Gaussian mixture also records the mean and variance of the points per dimension in each cell
  - More information for same # visual words
  - Does not increase computational time significantly
  - Leads to high-dimensional feature vectors
- Even when the counts are the same,

  the position and variance of the points in the cell can vary



Application of FV for Gaussian mixture model of local features

- Gaussian mixture models for local image descriptors

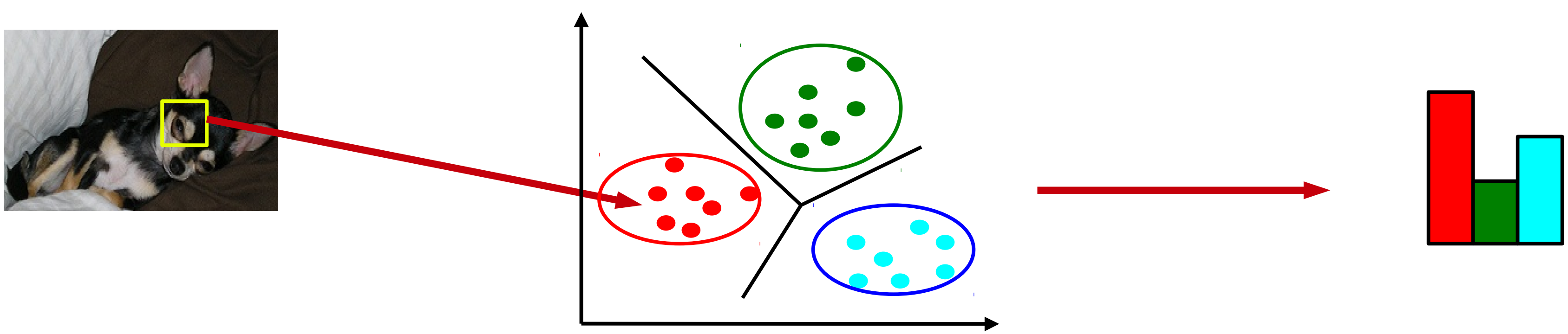
  [Perronnin & Dance, CVPR 2007]

  - State-of-the-art feature pooling for image/video classification/retrieval

- Offline: Train k-component GMM on collection of local features

$$p(x) = \sum_{k=1}^{K} \pi_k N(x; \mu_k, \sigma_k)$$

- Each mixture component corresponds to a visual word
  - Parameters of each component: mean, variance, mixing weight
  - We use diagonal covariance matrix for simplicity
    - Coordinates assumed independent, per Gaussian



Application of FV for Gaussian mixture model of local features

- Representation: gradient of data log-likelihood

- For the means and variances we have:

$$F^{-1/2}\nabla_{\mu_k}\ln p(x_{1:N})=\frac{1}{\sqrt{\pi_k}}\sum_{n=1}^{N}p(k|x_n)\frac{(x_n-\mu_k)}{\sigma_k}$$

$$F^{-1/2}\nabla_{\sigma_k}\ln p(x_{1:N})=\frac{1}{\sqrt{2\pi_k}}\sum_{n=1}^{N}p(k|x_n)\left\{\frac{(x_n-\mu_k)^2}{\sigma_k^2}-1\right\}$$

- Soft-assignments given by component posteriors

$$p(k|x_n)=\frac{\pi_k N(x_n;\mu_k,\sigma_k)}{p(x_n)}$$

Image representation using Fisher kernels

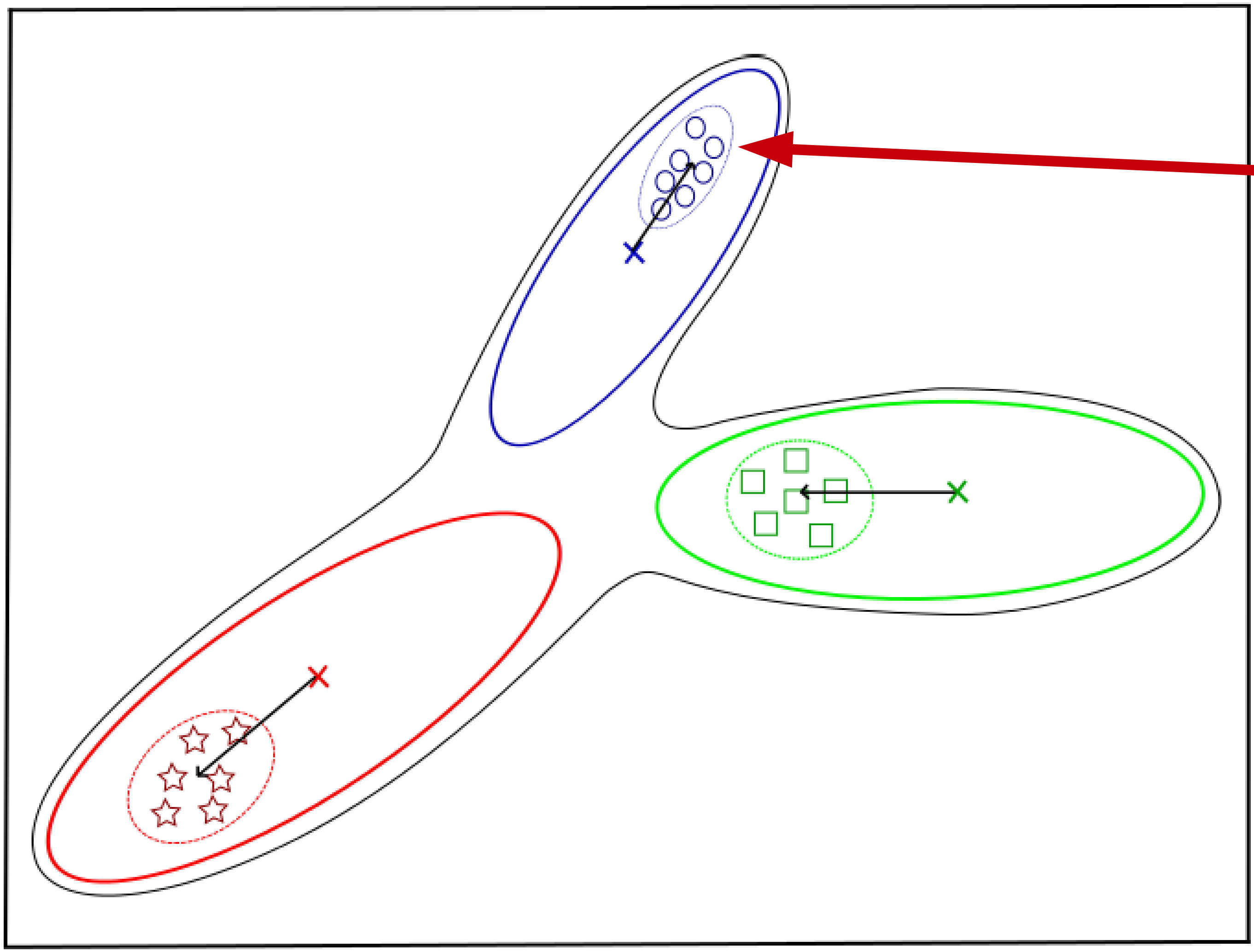
- Data representation

$$G(X,\Theta)=F^{-1/2}\left(\frac{\partial L}{\partial \alpha_1}, \ \ldots \ , \frac{\partial L}{\partial \alpha_K} \ , \ \nabla_{\mu_1} L, \ \ldots \ , \nabla_{\mu_K} L \ , \ \nabla_{\sigma_1} L, \ \ldots \ , \ \nabla_{\sigma_K} L\right)^T$$

- In total K(1+2D) dimensional representation, since for each visual word / Gaussian we have
  - Mixing weight (1 scalar)
  - Mean (D dimensions)
  - Variances (D dimensions, since single variance per dimension)

- Gradient with respect to mixing weights often dropped in practice since it adds little discriminative information for classification.
  - Results in 2KD dimensional image descriptor

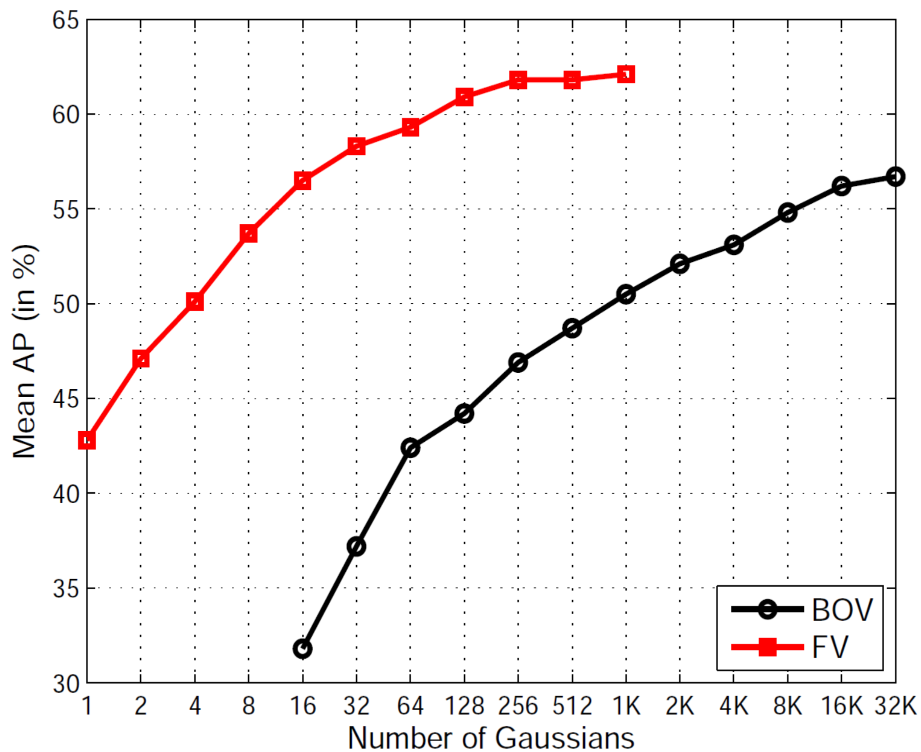
Illustration of gradient w.r.t. means of Gaussians

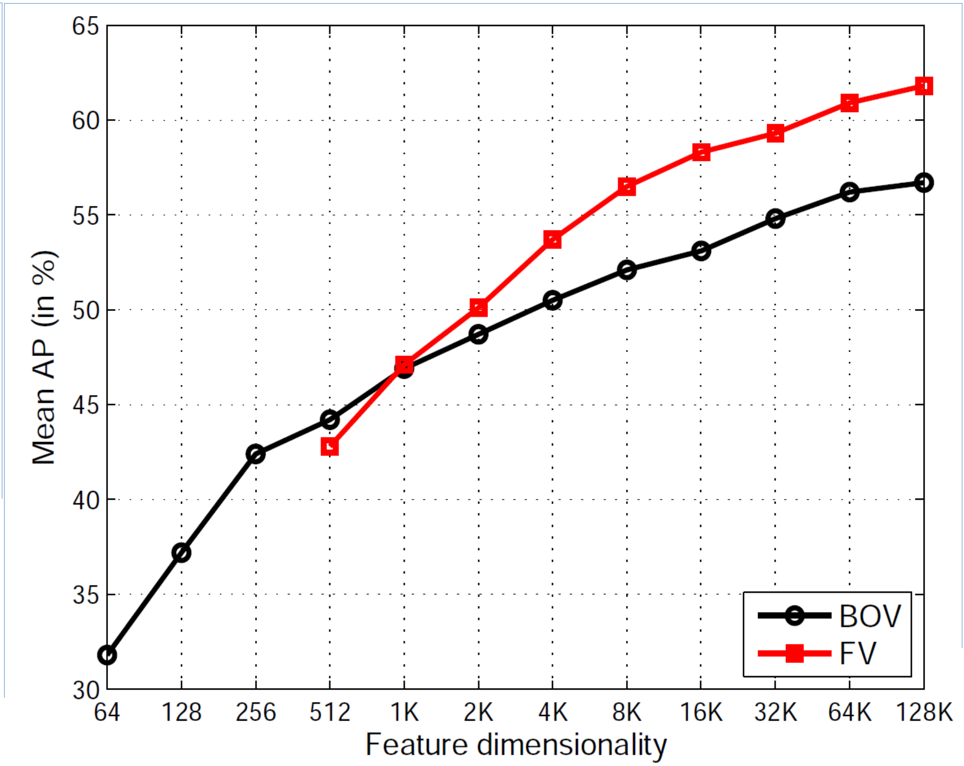


New Data Points

Fisher vectors: classification performance VOC'07

- Fisher vector representation yields better performance for a given number of Gaussians / visual words than Bag-of-words.
- For a fixed dimensionality Fisher vectors perform better, and are more efficient to compute





Normalization of the Fisher vector

- Inverse Fisher information matrix *F* $F = E[g(x)g(x)^T]$
  - Renders FV invariant for re-parametrization $f(x) = F^{-1/2} g(x)$
  - Linear projection, analytical approximation for MoG gives diagonal matrix [Jaakkola, Haussler, NIPS 1999], [Sanchez, Perronnin, Mensink, Verbeek IJCV'13]

- Power-normalization, applied independently per dimension
  - Renders Fisher vector less sparse (typically rho=0.5) $f(x) \leftarrow sign(f(x))\, |f(x)|^{\rho}$ $0 < \rho < 1$
    [Perronnin, Sanchez, Mensink, ECCV'10]
  - Corrects for poor independence assumption on local descriptors
    [Cinbis, Verbeek, Schmid, PAMI'15]

- L2-normalization $f(x) \leftarrow \dfrac{f(x)}{\sqrt{f(x)^T f(x)}}$
  - Makes representation invariant to number of local features
  - Among other Lp norms the most effective with linear classifier
    [Sanchez, Perronnin, Mensink, Verbeek IJCV'13]

Effect of power and L2 normalization in practice

- Classification results on the PASCAL VOC 2007 benchmark dataset.
- Regular dense sampling of local SIFT descriptors in the image
  - PCA projected to 64 dimensions to de-correlate and compress
- Using mixture of 256 Gaussians over the SIFT descriptors
  - FV dimensionality: 2*64*256 = 32 * 1024

Power Nomalization	L2 normalization	Performance (mAP)	Improvement over baseline
No	No	51.5	0
Yes	No	59.8	8.3
No	Yes	57.3	5.8
Yes	Yes	61.8	10.3

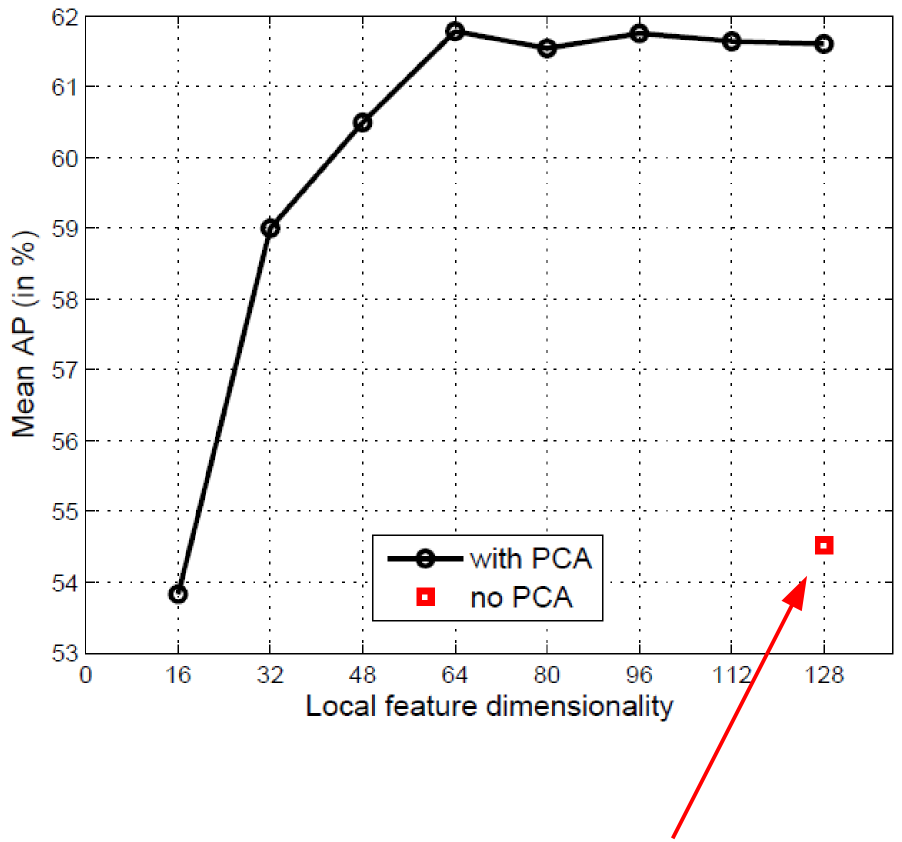
PCA dimension reduction of local descriptors

- We use diagonal covariance model in GMM for simplicity and efficiency
- But dimensions might be correlated

- Apply PCA projection to
  - De-correlate features
  - Reduce dimension of final FV

- FV with 256 Gaussians over local SIFT descriptors of dimension 128

Results on PASCAL VOC’07:



Bag-of-words vs. Fisher vector representation

- Bag-of-words image representation
  - k-means clustering
  - histogram of visual word counts, K dimensions
- Fisher vector image representation
  - GMM clustering
  - Local first and second order moments, 2KD dimensions
- For a given dimension of the representation
  - FV needs less clusters, and is faster to compute
  - FV gives better performance since it is a smoother function of the local descriptors.

- Review article on Fisher Vector image representation

  Image Classification with the Fisher Vector: Theory and Practice

  Sanchez, Perronnin, Mensink, Verbeek

  International Journal of Computer Vision, 2013