Fisher Vector image representation

Machine Learning and Object Recognition 2016-2017

Jakob Verbeek

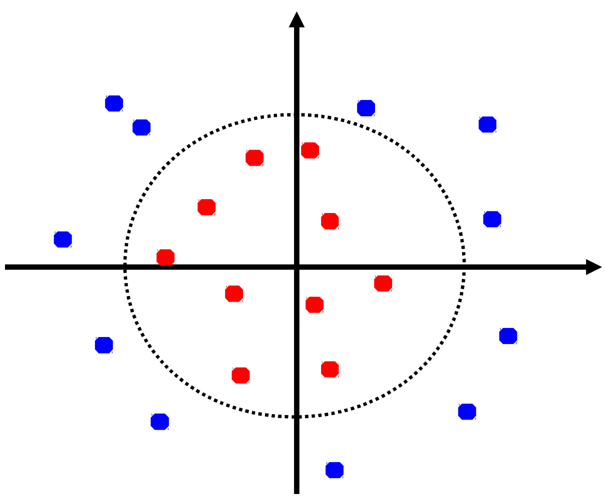
Course website:

http://thoth.inrialpes.fr/~verbeek/MLOR.16.17

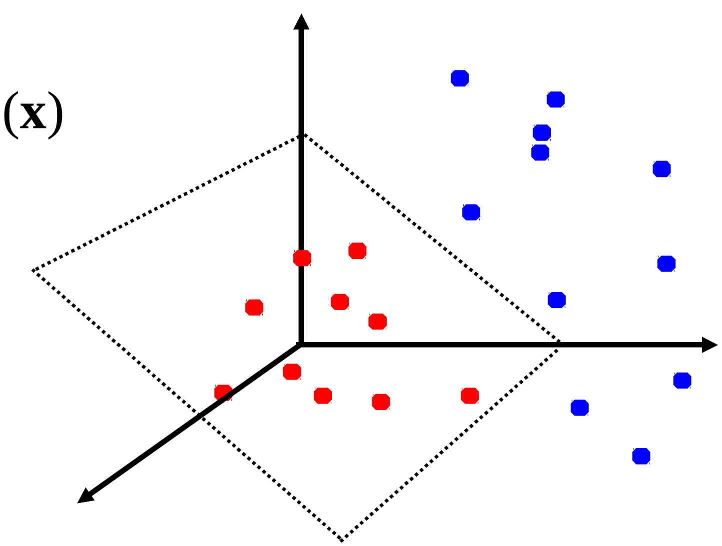
A brief recap on kernel methods

- 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.

$$k(x_{1,} x_2) = \langle \phi(x_1), \phi(x_2) \rangle$$



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



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 define a representation of sets by concatenating the mean and variance of the set in each dimension

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

  - 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 $p(x;\theta),\ \ x\in X\ ,\ \ \theta\in R^D$
- Represent data x with the 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 kernels

- 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**

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

- It follows that the Fisher kernel is a positive-semidefinite

$$\alpha^T K \alpha = \| \sum_i \alpha_i \phi(x_i) \|_2^2 = \| L \sum_i \alpha_i g(x_i) \|_2^2 \geq 0$$

  - where

$$[K]_{ij} = k(x_i, x_j)$$

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

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

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 kernels: example with Gaussian data model

- Let lambda be the inverse variance, i.e. precision, parameter

$$p(x) = N(x;\mu,\lambda) = \sqrt{\lambda/(2\pi)}\exp\left[-\frac{1}{2}\lambda(x-\mu)^2\right]$$

$$\ln p(x) = \frac{1}{2}\ln\lambda - \frac{1}{2}\ln(2\pi) - \frac{1}{2}\lambda(x-\mu)^2$$

$$\theta = (\mu,\lambda)^T$$

- The partial derivatives and Fisher information matrix are found to be

$$\frac{\partial \ln p(x)}{\partial \mu} = \lambda(x-\mu) \qquad \frac{\partial \ln p(x)}{\partial \lambda} = \frac{1}{2}\left[\lambda^{-1} - (x-\mu)^2\right]$$

$$F = \begin{pmatrix} \lambda & 0 \\ 0 & \frac{1}{2}\lambda^{-2} \end{pmatrix}$$

- The Fisher vector is then

$$\phi(x) = \begin{pmatrix} (x-\mu)/\sigma \\ \left(\sigma^2 - (x-\mu)^2\right)/\left(\sigma^2\sqrt{2}\right) \end{pmatrix}$$

Fisher kernels: example with Gaussian data model

- Now suppose an i.i.d. data model over a set of data points

$$p(x) = N(x;\mu,\lambda) = \sqrt{\lambda/(2\pi)}\exp\left[-\frac{1}{2}\lambda(x-\mu)^2\right]$$

$$p(X) = p(x_{1,\ldots},x_N) = \prod_{i=1}^{N} p(x_i)$$

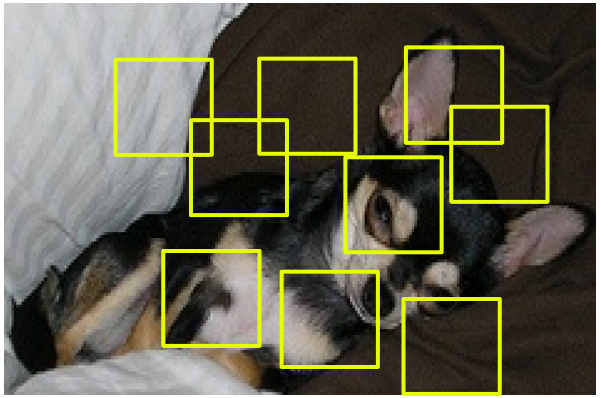
- Then the Fisher vector is given by the sum of Fisher vectors of the points
  - Encodes the discrepancy in the first and second order moment of the data w.r.t. those of the model

$$\phi(X) = \sum_{i=1}^{N} \phi(x_i) = N\begin{pmatrix} (\hat{\mu}-\mu)/\sigma \\ (\sigma^2-\hat{\sigma}^2)/(\sigma^2\sqrt{2}) \end{pmatrix}$$

  - Where

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \hat{\sigma} = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2$$

Local descriptor based image representations



- Patch extraction and description stage
  - For example: SIFT, HOG, LBP, color, ...
  - Dense multi-scale grid, or interest points

$$X = \{x_{1,}, \ldots, x_N\}$$

- Coding stage: embed local descriptors, typically in higher dimensional space
  - For example: assignment to cluster indices

$$\phi(x_i)$$

- Pooling stage: aggregate per-patch embeddings
  - For example: sum pooling

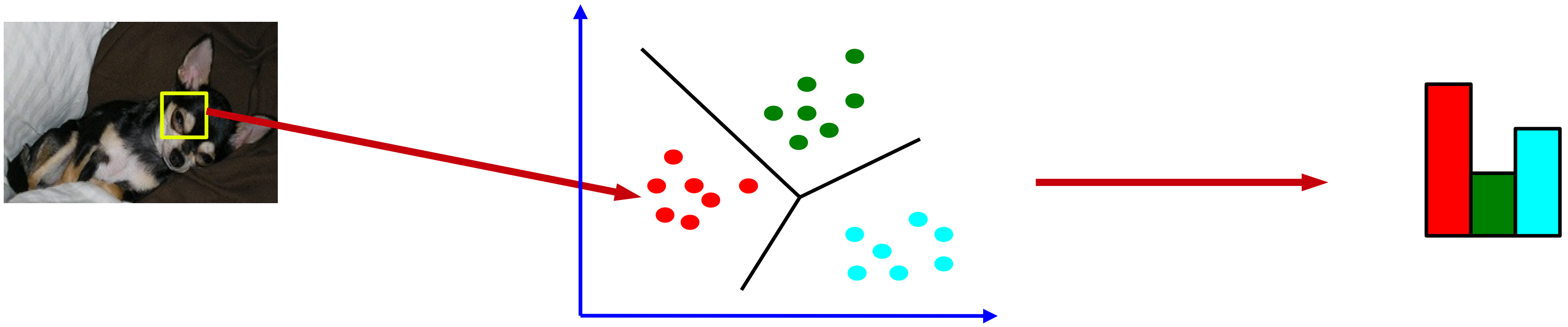
$$\Phi(X) = \sum_{i=1}^{N} \phi(x_i)$$

Bag-of-word image representation

- Extract local image descriptors, e.g. SIFT
  - Dense on multi-scale grid, or on interest points

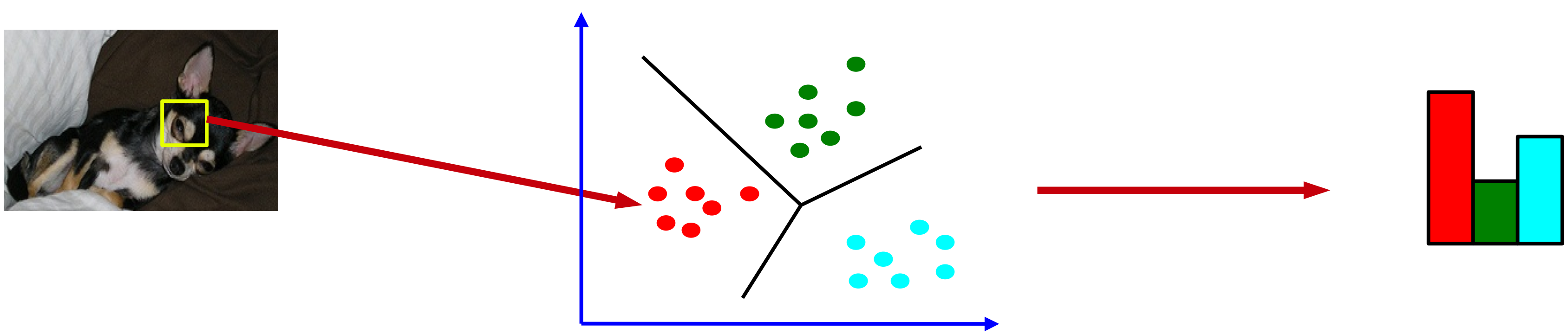
- Off-line: cluster local descriptors with k-means
  - Using random subset of patches from training images

- To represent training or test image
  - Assign SIFTs to cluster indices / visual words $\phi(x_i) = [0,\ldots,0,1,0,\ldots,0]$
  - Histogram of cluster counts aggregates all local feature information
    [Sivic & Zisserman, ICCV'03], [Csurka et al., ECCV'04] $h = \sum_i \phi(x_i)$



Application of FV for bag-of-words image-representation

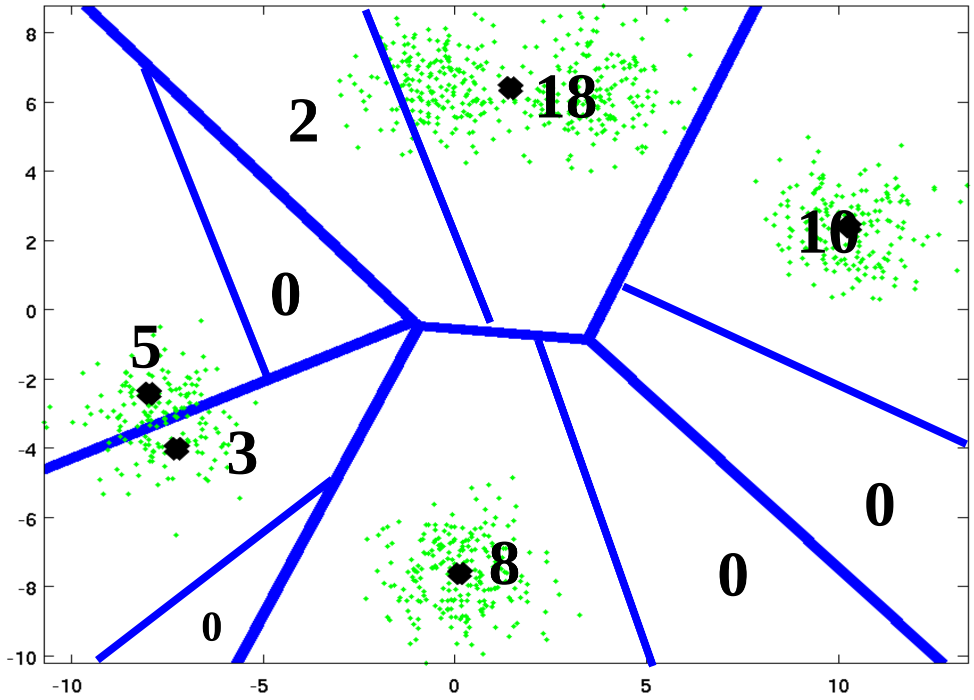
- Bag of word (BoW) representation
  - Map every descriptor to a cluster / visual word index $w_i \in \{1, \ldots, K\}$
- Model visual word indices with i.i.d. multinomial $p(w_i = k) = \frac{\exp \alpha_k}{\sum_{k'} \exp \alpha_{k'}} = \pi_k$
  - Likelihood of N i.i.d. indices: $p(w_{1:N}) = \prod_{i=1}^{N} p(w_i)$
  - Fisher vector given by gradient $\frac{\partial \ln p(w_{1:N})}{\partial \alpha_k} = \sum_{i=1}^{N} \frac{\partial \ln p(w_i)}{\partial \alpha_k} = h_k - N \pi_k$
    - i.e. BoW histogram + constant



Fisher vector GMM representation: Motivation

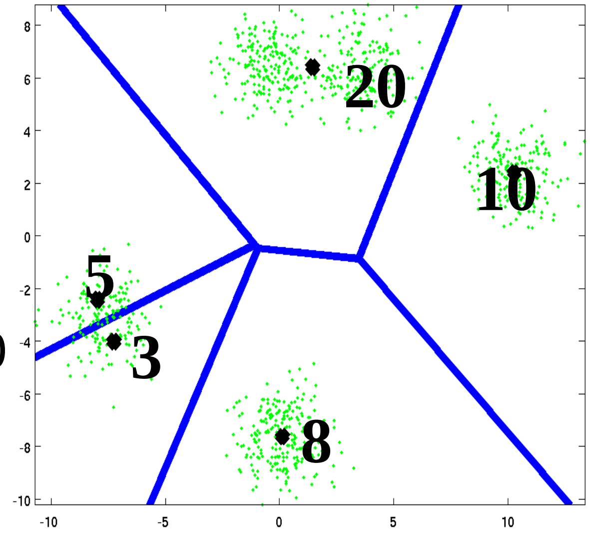
- Suppose we want to refine a given visual vocabulary to obtain a richer image representation

- 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 GMM representation: Motivation

- Feature vector quantization is computationally expensive
- To extract visual word histogram for a new image
  - Compute distance of each local descriptor to each k-means center
  - run-time O(NKD) : linear in
    - N: nr. of feature vectors ~ 10^4 per image
    - K: nr. of clusters ~ 10^3 for recognition
    - D: nr. of dimensions ~ 10^2 (SIFT)



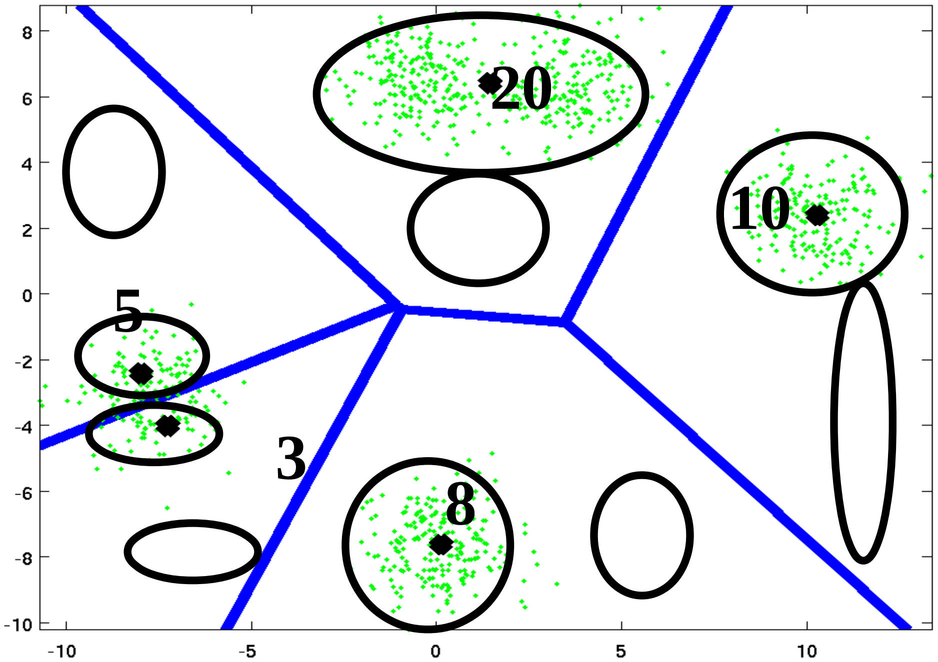
- So in total in the order of 10^9 multiplications per image to obtain a histogram of size 1000

- Can this be done more efficiently ?!
  - Yes, extract more than just a visual word histogram from a given clustering

Fisher vector representation in a nutshell

- Instead, the Fisher Vector for GMM 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

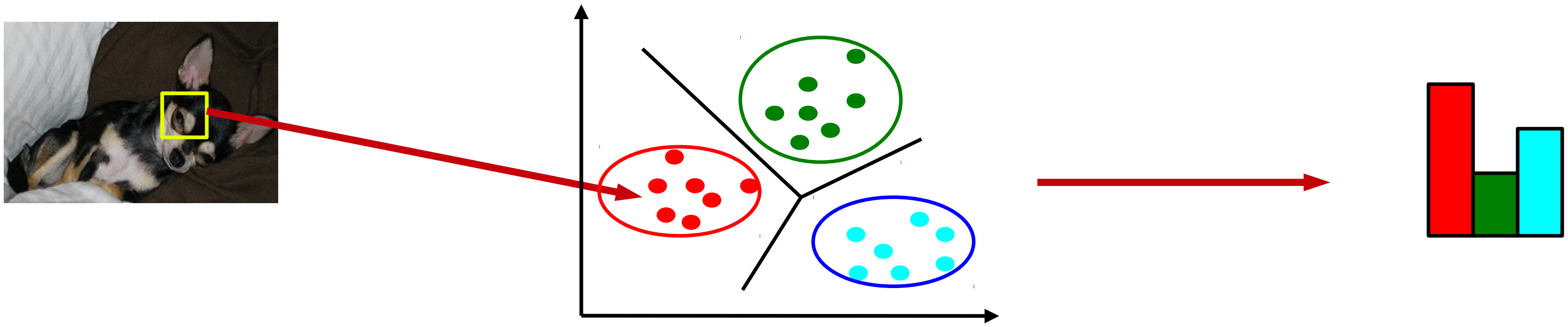
- Gaussian mixture models for local image descriptors

  [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

- Gaussian mixture models for local image descriptors

  [Perronnin & Dance, CVPR 2007]
  - State-of-the-art feature pooling for image/video classification/retrieval

- Representation: gradient of 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

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

Application of FV for Gaussian mixture model of local features

- Fisher vector components give the difference between the data mean predicted by the model and observed in the data, and similar for variance.

- For the gradient w.r.t. the mean

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

  - where $n_k=\sum_{n=1}^{N} p(k|x_n)$ $\hat{\mu}_k=n_k^{-1}\sum_{n=1}^{N} p(k|x_n)x_n$

- Similar for the gradient w.r.t. the variance

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

  - where $\hat{\sigma}_k^2=n_k^{-1}\sum_{n=1}^{N} p(k|x_n)(x_n-\mu_k)^2$

Image representation using Fisher kernels

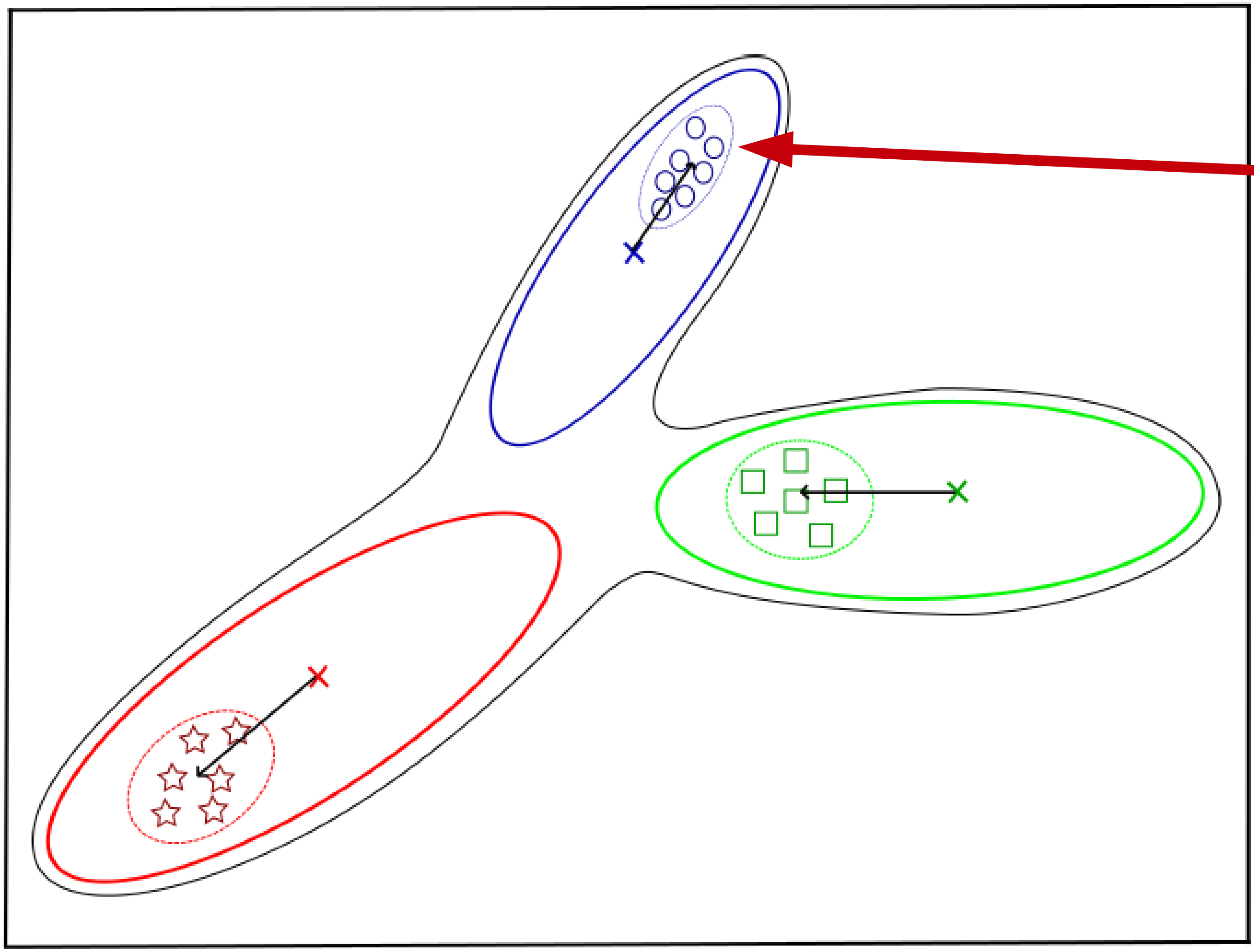
- Data representation

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

- 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

BoW and FV from a function approximation viewpoint

- Let us consider uni-dimensional descriptors: vocabulary quantizes real line
- For both BoW and FV the representation of an image is obtained by sum-pooling the representations of descriptors.
  - Ensemble of descriptors sampled in an image $X = \{x_{1,}, \dots, x_N\}$
  - Representation of single descriptor
    - One-of-k encoding for BoW $\phi(x_i) = [0, \dots, 0, 1, 0, \dots, 0]$
    - For FV concatenate per-visual word gradients of form

$$\phi(x_i) = \left(\dots, p(k|x_i) \left[1 \quad \frac{(x_i - \mu_k)}{\sigma_k} \quad \frac{(x_i - \mu_k)^2 - \sigma_k^2}{\sigma_k^2} \right], \dots \right)$$

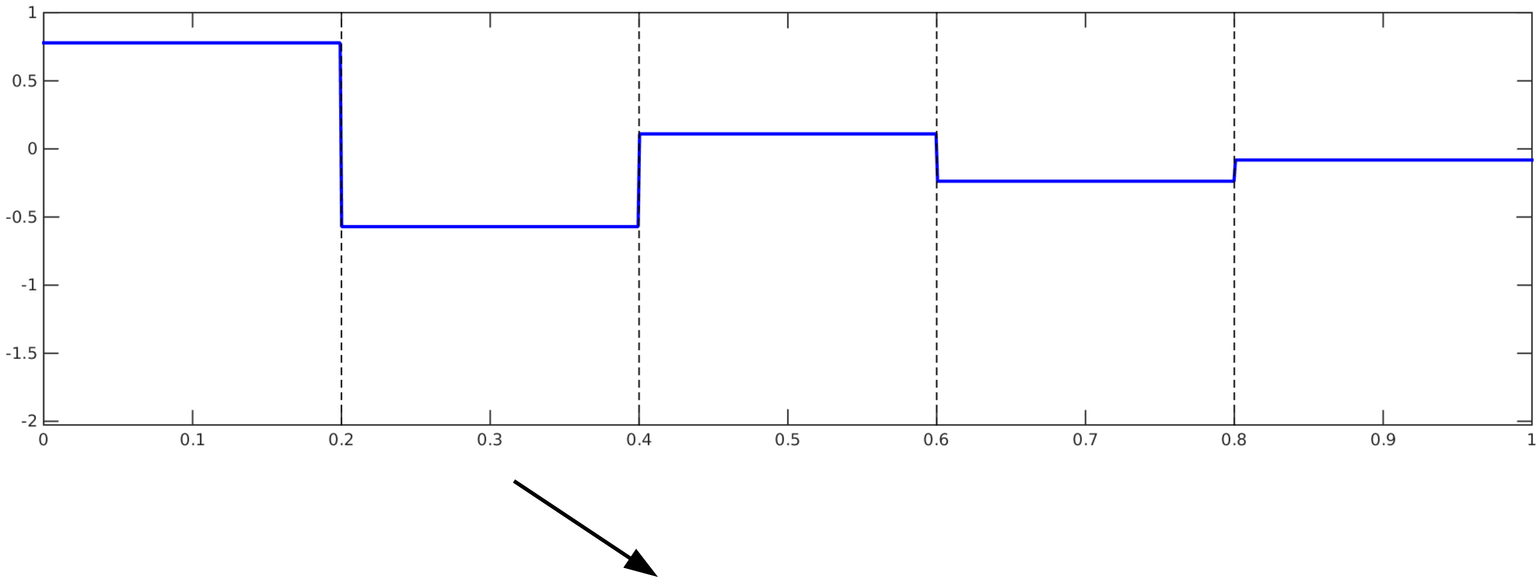
- Linear function of sum-pooled descriptor encodings is a sum of linear functions of individual descriptor encodings:

$$\Phi(X) = \sum_{i=1}^{N} \phi(x_i)$$

$$w^T \Phi(X) = \sum_{i=1}^{N} w^T \phi(x_i)$$

From a function approximation viewpoint

- Consider the score of a single descriptor for BoW
  - If assigned to k-th visual word then $w^T \phi(x_i) = w_k$
  - Thus: constant score for all descriptors assigned to a visual word



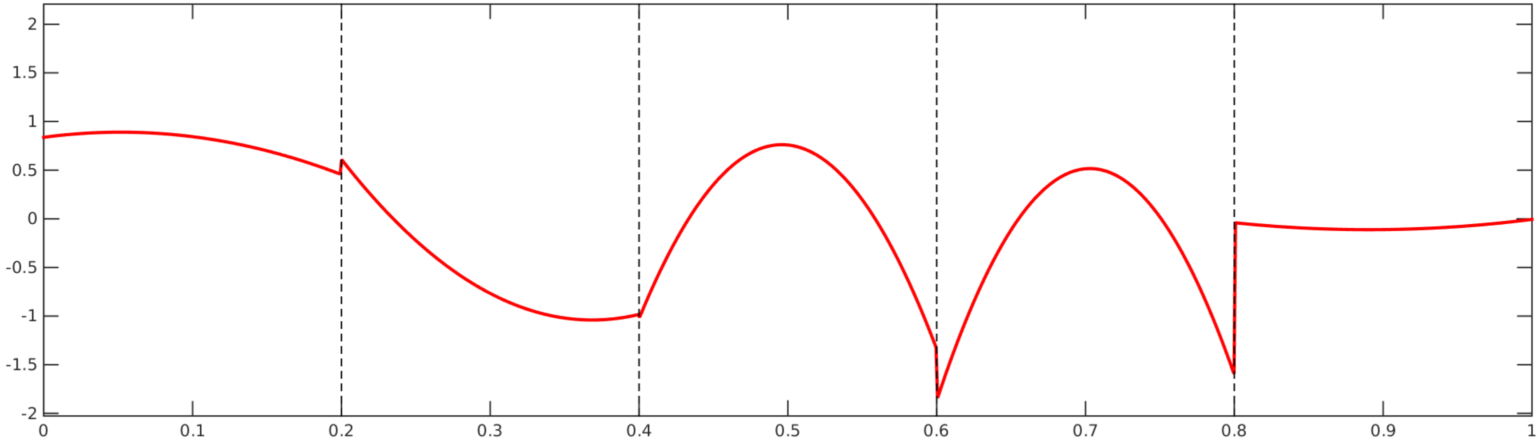
Each cell corresponds to a visual word

From a function approximation viewpoint

- Consider the same for FV, and assume soft-assignment is “hard”
  - Thus: assume for one value of k we have $p(k|x_i) \approx 1$
  - If assigned to the k-th visual word:

$$w^T \phi(x_i) = w_k^T \begin{bmatrix} 1 & \frac{(x_i - \mu_k)}{\sigma_k} & \frac{(x_i - \mu_k)^2 - \sigma_k^2}{\sigma_k^2} \end{bmatrix}$$

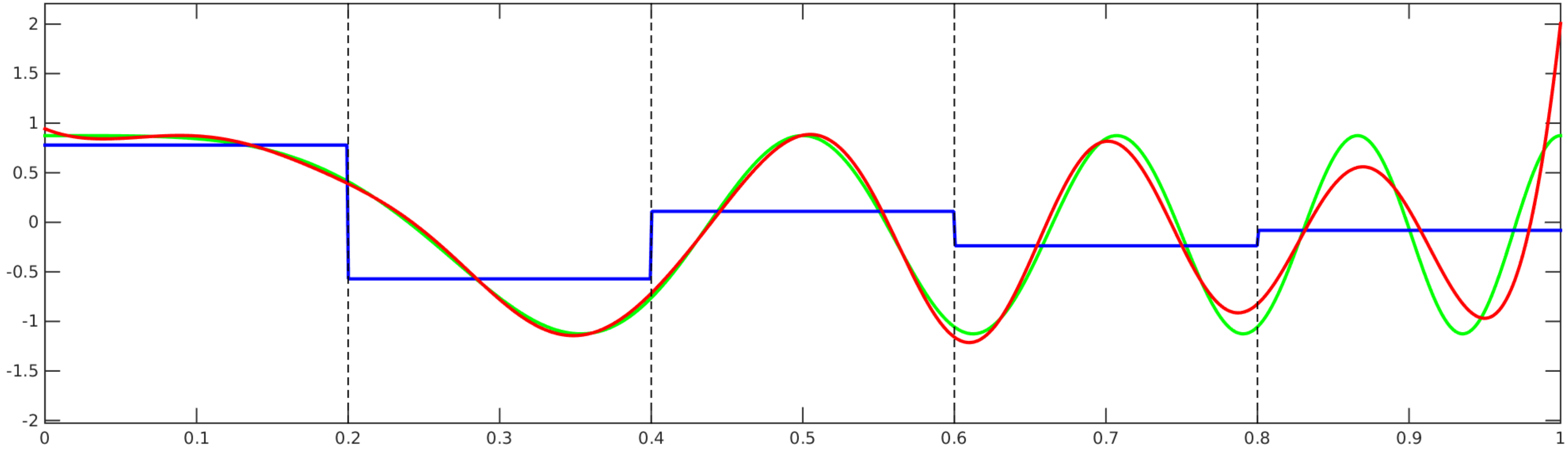
    - Note that w_k is no longer a scalar but a vector
  - Thus: score is a second-order polynomial of the descriptor x, for descriptors assigned to a given visual word.



From a function approximation viewpoint

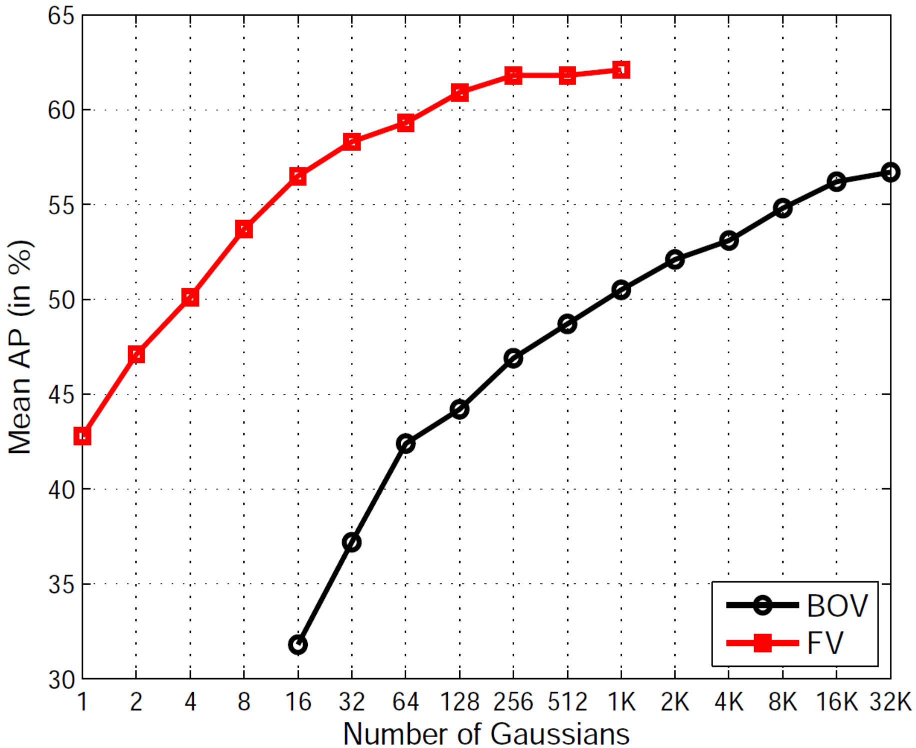
- Consider that we want to approximate a true classification function (green) based on either BoW (blue) or FV (red) representation
  - Weights for BoW and FV representation fitted by least squares to optimally match the target function

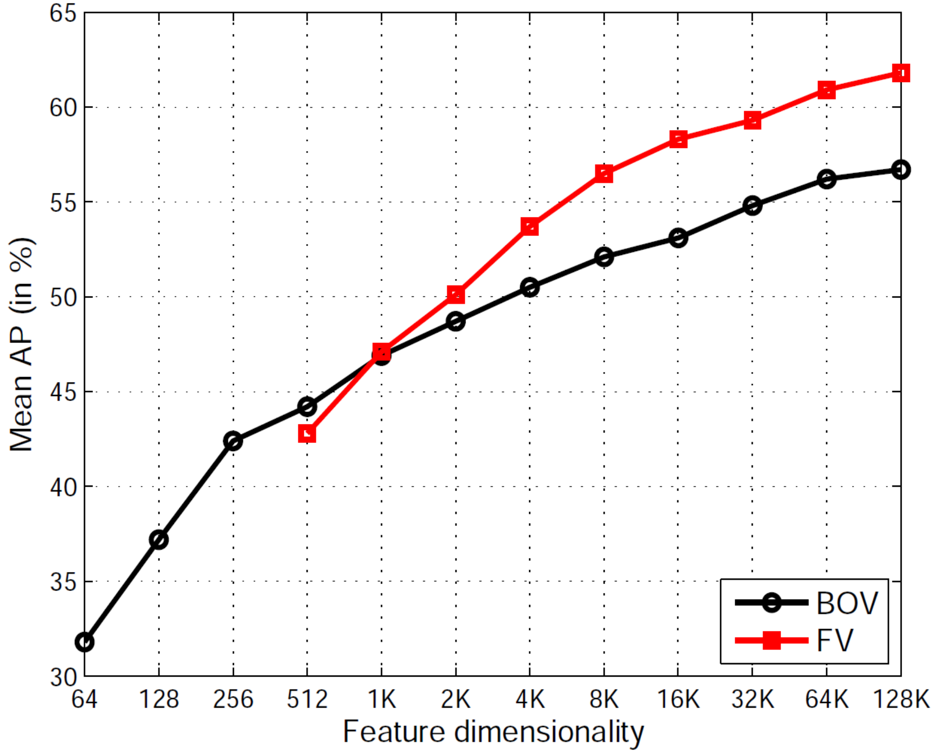
- Better approximation with FV
  - Local second order approximation, instead of local zero-order
  - Smooth transition from one visual word to the next



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*
  - Renders FV invariant for re-parametrization
  - Linear projection, analytical approximation for MoG gives diagonal matrix
    [Jaakkola, Haussler, NIPS 1999], [Sanchez, Perronnin, Mensink, Verbeek IJCV'13]

$$F = E[g(x)g(x)^T]$$
$$f(x) = F^{-1/2} g(x)$$

- Power-normalization, applied independently per dimension
  - Renders Fisher vector less sparse
    [Perronnin, Sanchez, Mensink, ECCV'10]
  - Corrects for poor independence assumption on local descriptors
    [Cinbis, Verbeek, Schmid, PAMI'15]

$$f(x) \leftarrow sign(f(x)) |f(x)|^{\rho}$$
$$0 < \rho < 1$$

- L2-normalization
  - Makes representation invariant to number of local features
  - Among other Lp norms the most effective with linear classifier
    [Sanchez, Perronnin, Mensink, Verbeek IJCV'13]

$$f(x) \leftarrow \frac{f(x)}{\sqrt{f(x)^T f(x)}}$$

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

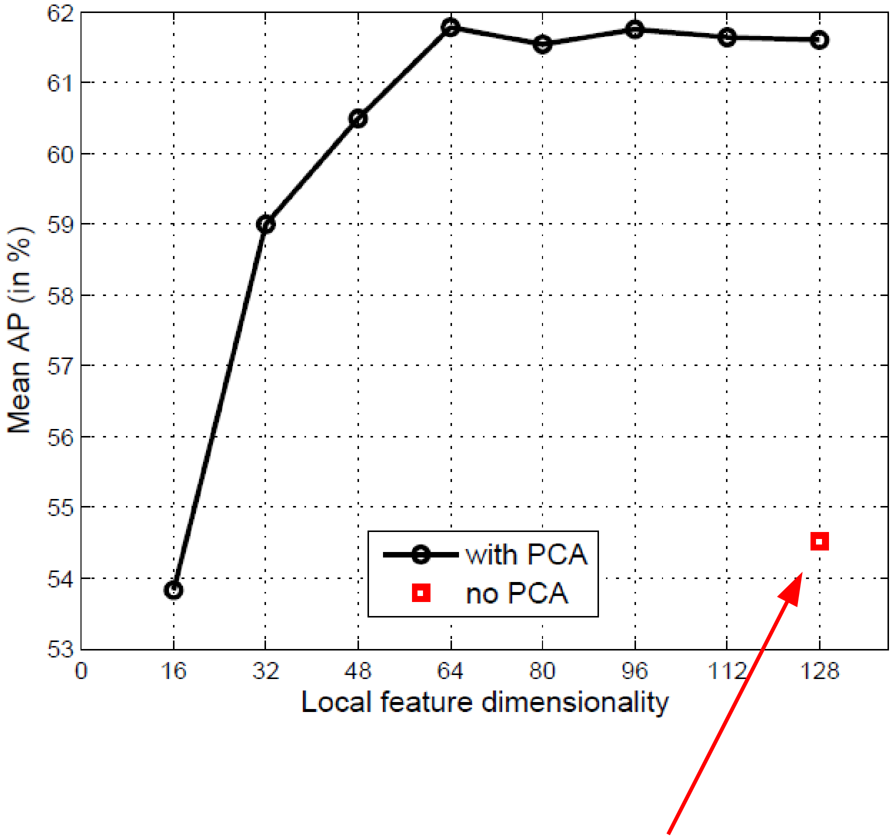
- Dimensions might be correlated

- 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

- Both representations based on a visual vocabulary obtained by means of clustering local descriptors

- Bag-of-words image representation
  - Off-line: fit k-means clustering to local descriptors
  - Representation: histogram of visual word counts, K dimensions

- Fisher vector image representation
  - Off-line: fit GMM model to local descriptors
  - Representation: gradient of log-likelihood, 2KD dimensions

Summary of Fisher vector image representation

- Computational cost similar:
  - Both compare N descriptors to K clusters (visual words)
- Memory usage:
  - Fisher vector has size 2KD for K clusters and D dim. descriptors
  - Bag-of-word has size K for K clusters
- 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

- A recent overview article on Fisher Vector representation
  - Image Classification with the Fisher Vector: Theory and Practice

    Sanchez, Perronnin, Mensink, Verbeek

    International Journal of Computer Vision, 2013