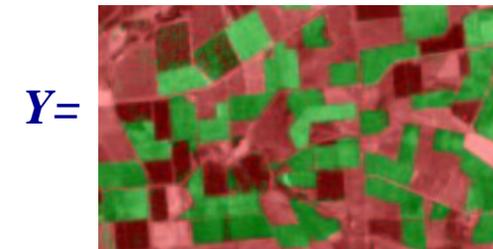


Machine Learning: Fitting Predictive Models to Data

- Observe two correlated “processes”
 - Process that produces data samples $\mathbf{x} \in X$
 - Process that provides labels $y \in Y$
- Find a predictor function $f : X \rightarrow Y$
 - Given a sample \mathbf{x} from the data process, use $\hat{y} = f(\mathbf{x})$ to predict the associated label y .
- Maybe you want to understand something about the processes. . . or maybe you just want to make successful predictions in the future
- It is traditional to make assumptions about the processes
 - linear, gaussian, deterministic, simple, *etc.*
 - . . . but it is not necessary.
- Obtain a training set of data: $(\mathbf{x}_i, y_i), i = 1 \dots m$
 - Assume: the training data are representative of the processes



The machine learning problem(s)

- “Processes” described by a probability distribution P over the space $X \times Y$
 - X =data; in general, quite arbitrary, but usually X is \mathcal{R}^d
 - Y =labels
 - Binary classification: Y is $\{-1, 1\}$
 - K -ary classification: Y is $\{1, \dots, K\}$
 - Regression: Y is usually \mathcal{R}
- $P(X, Y)$ is not known, but...
 - what is available is a data set $(x_i, y_i), i = 1 \dots m$
 - which is assumed to be randomly sampled from $P(x, y)$
 - usually assume independent samples: IID
- Infer properties of the distribution, conditioned on $x \in X$
 - Infer distribution itself: estimate $f(y, x) = P(y|x)$
 - Regression: estimate $f(x) = E(Y|x) = \int_Y yP(y|x) dy$
 - Classification: estimate $f(x) = \operatorname{argmax}_y P(y|x)$

Other machine learning problems

■ Transductive learning

- Given one (small) data set $(\mathbf{x}_i, \mathbf{y}_i)$, $i = 1 \dots m$, sampled from the distribution $P(\mathbf{x}, \mathbf{y})$, and a second (generally much larger) data set without labels \mathbf{x}_j , $j = 1 \dots M$, build a model for regression, classification, etc.

■ Unsupervised learning

- Given only data without labels \mathbf{x}_i , $i = 1 \dots m$, infer properties of $P(\mathbf{x})$
 - eg, estimate of number of modes in a multimodal model.

■ Anomaly detection

- Given a set of data without labels \mathbf{x}_i , $i = 1 \dots m$ which is presumed “normal,” and a second set of unlabelled data \mathbf{x}_j , $j = 1 \dots M$, identify the (presumed rare) elements of the second data set which are not normal.

■ Confidence machines

- Build a model that not only guesses the data point’s classification, but which also produces an estimate of the probability that that point is correctly classified. (“sixty percent chance of rain”)

Classifier performance

- *Generalization* error expresses performance on “future” data

$$E[f] = \int I(y \neq f(\mathbf{x}))P(\mathbf{x}, y)dx dy$$

- where $I(\cdot)$ is the indicator function;
it is one if its argument is true, zero if false.
- Optimal solution $f^* = \operatorname{argmin}_f E[f]$ given by $f^*(\mathbf{x}) = \operatorname{argmax}_y P(y|\mathbf{x})$
 - produces minimum error (aka “Bayes error”); not necessarily zero
 - requires knowledge of $P(\mathbf{x}, y)$

- *Empirical* or *Training* error is based on in-sample performance

$$E_t = \sum_i I(y_i \neq f(\mathbf{x}_i))$$

- Can be evaluated over training data: $(\mathbf{x}_i, y_i), i = 1 \dots m$
- Choosing $f(\mathbf{x})$ to minimize E_t will not necessarily minimize E
- If $E_t[f]$ is estimated with the same data used to fit the predictor f , then $E_t[f]$ will be a biased estimator of E^* .

- Overfit: try “too hard” to fit training data and generalization suffers.

The Probably Approximately Correct (PAC) Approach

- Valiant (1984): Produce a model of the data that...
 - ... with probability $1 - \delta$
 - ... is accurate to within ϵ
 - ... using computation that is polynomial in $1/\delta$ and $1/\epsilon$.
- Why not probability one?
 - The finite training set is randomly sampled from distribution $P(X, Y)$. With some probability $\delta \ll 1$, the random sample will be unrepresentative of the distribution.
- Avoids the Bayesian Nightmare
 - Probability is $P(\text{data}|\text{distribution})$, Not $P(\text{distribution}|\text{data})$
- No Free Lunch Theorems (Wolpert 1992, 1995, 1996)
 - No reason to expect one learning algorithm to be better than another
 - unless you make assumptions about the “uniformity of nature”
 - prior assumptions about $P(\text{distribution})$.

PAC and Boosting

- **Strong learning:** For any $\epsilon > 0$ and any $\delta > 0$, and access to an unlimited number of data samples, find a function $f(\mathbf{x})$ such that
 - With probability $1 - \delta$
 - solution is accurate to within ϵ ; ie, $\mathcal{P}(y \neq f(\mathbf{x})) < \epsilon$
 - In a number of steps that is polynomial in $1/\delta$ and $1/\epsilon$ and the number of data samples employed.
- **Weak learning:** Replace “any $\epsilon > 0$ ” with “some $\epsilon < 1/2$ ”
- **Boosting**
 - Kearns and Valiant posed the question: could a “weak” PAC learner be “boosted” to a “strong” learner
 - Schapire constructed first provably polynomial-time boosting algorithm The strength of weak learnability. *Machine Learning* 5:197–227, 1990
 - Many improvements followed. . .
- **Note:** no assumptions made about underlying distribution $P(X, Y)$
 - *Any* weak learner can be boosted to a strong learner

AdaBoost

- **Given data:** $(\mathbf{x}_i, y_i), i = 1 \dots m$; where $\mathbf{x}_i \in \mathbf{X}, y_i \in \{-1, 1\}$
- **And a family of “weak learners”** $h \in \mathcal{H}$; where $h : \mathbf{X} \rightarrow \{-1, 1\}$
 - **More general variant:** $h : \mathbf{X} \rightarrow \mathcal{R}$
- **Start with initial distribution of weights:** $w_i^0 = 1/m$
- **For $t = 1, \dots, T$**
 - **Find weak learner h_t that minimizes weighted error:**
$$\epsilon_t = \sum_i w_i^t I(y_i \neq h_t(\mathbf{x}_i)) / \sum_i w_i^t$$
 - **Assign weight α_t to the learner**
 - Larger weights assigned to learners with smaller errors
 - Usually, but not necessarily: $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$
 - **Adaptively reweight the data points:** $w_i^{t+1} = w_i^t \exp(-\alpha_t y_i h_t(\mathbf{x}_i))$
 - Note that the data points which were badly estimated this time will be more heavily weighted the next time.
- **Final estimator:** $\hat{y} = H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$

PAC and Support Vector Machines

■ Linear decision function: $f(\mathbf{x}) = \beta^T \mathbf{x} + \beta_o$

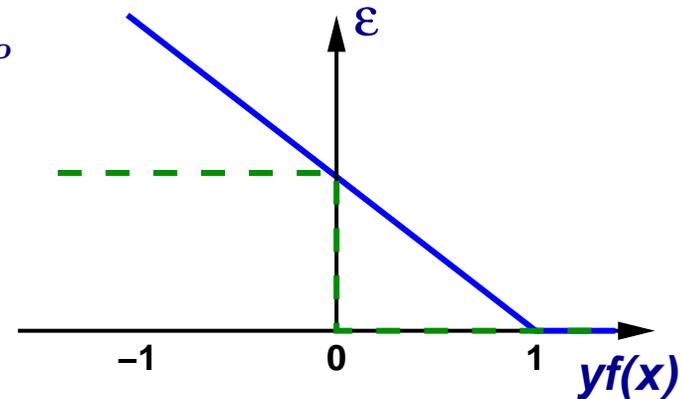
■ Classifier: $\hat{y} = \text{sign}(f(\mathbf{x}))$

■ SVM loss function

$$L(\beta, \beta_o) = \frac{1}{2} \beta^T \beta + C \sum_i \varepsilon_i$$

• where $\varepsilon_i = \max(0, 1 - y_i f(\mathbf{x}_i))$

• convex • margin



■ Optimization with bounded effort: quadratic programming (QP) problem

• Solution has the form $f(\mathbf{x}) = \left(\sum_i \alpha_i y_i \mathbf{x}_i^T \right) \mathbf{x} + \beta_o = \sum_i \alpha_i y_i (\mathbf{x}_i^T \mathbf{x}) + \beta_o$

■ Bounded generalization error

• based on Vapnik's amazing theorem: with probability $1 - \delta$,

$$E \leq E_t + \sqrt{\frac{\nu(\log(2m/\nu) + 1) - \log(\delta/4)}{m}}$$

• where ν is the VC dimension: $\nu \sim |\beta|^2$

• Independent of distribution $P(\mathbf{X}, \mathbf{Y})$

• Valid for all m

Kernels and Support Vector Machines

- Enables SVM to learn highly nonlinear decision surfaces
- Map data to a (usually) higher-dimensional feature space $\phi : \mathbf{X} \rightarrow \mathcal{F}$

- For example: $\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \end{bmatrix}$

- Nonlinear $f(\mathbf{x}) = \beta\phi(\mathbf{x}) + \beta_o$ is linear in feature space \mathcal{F}
- Do SVM algorithm (QP problem) in feature space
 - But not directly, feature space is too high-dimensional
 - Observe that QP algorithm only involves dot products: $\phi(\mathbf{x})^T \phi(\mathbf{x}')$
 - Define kernel function: $\mathbf{K}(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$
- Solution has the form: $f(\mathbf{x}) = \sum_i \alpha_i y_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}) + \beta_o$
 - Often, solution is sparse: many values of α_i are zero
 - Data \mathbf{x}_i corresponding to nonzero α_i are “support vectors”
- Trick: choose mappings ϕ which lead to convenient kernels

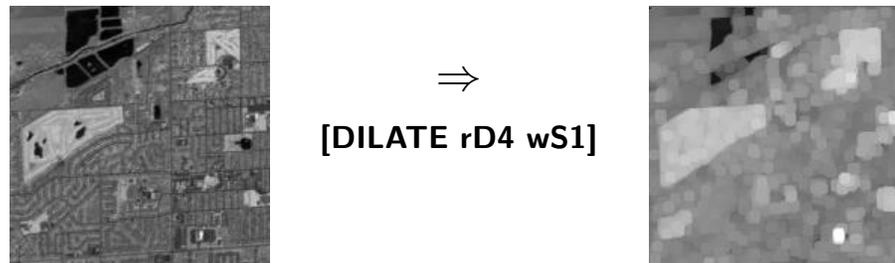
Using Spatial Information in Multispectral Classification

- Ignore spatial properties entirely, and use purely spectral methods
 - max likelihood, min distance, Fisher discriminant, SVM, *etc.*
 - k-means and EM for unsupervised classification
- Process local spatial features into scratch planes
And treat scratch planes as extra spectral channels
 - Texture features (Gabor, Laws, *etc.*)
 - Morphological operators (erode, dilate, *etc.*)
 - Adaptive/Online feature selection (Genie, Afreet, Grafting, . . .)
- Use contiguity property as part of the cost function
 - Markov Random Field methods
 - Bayesian interpretation, MCMC computation, Gibbs sampler, texture synthesis, simulated annealing, various nightmares. . .
 - Direct minimization of the cost function
 - contig-SVM for supervised
 - contig-k-means for unsupervised

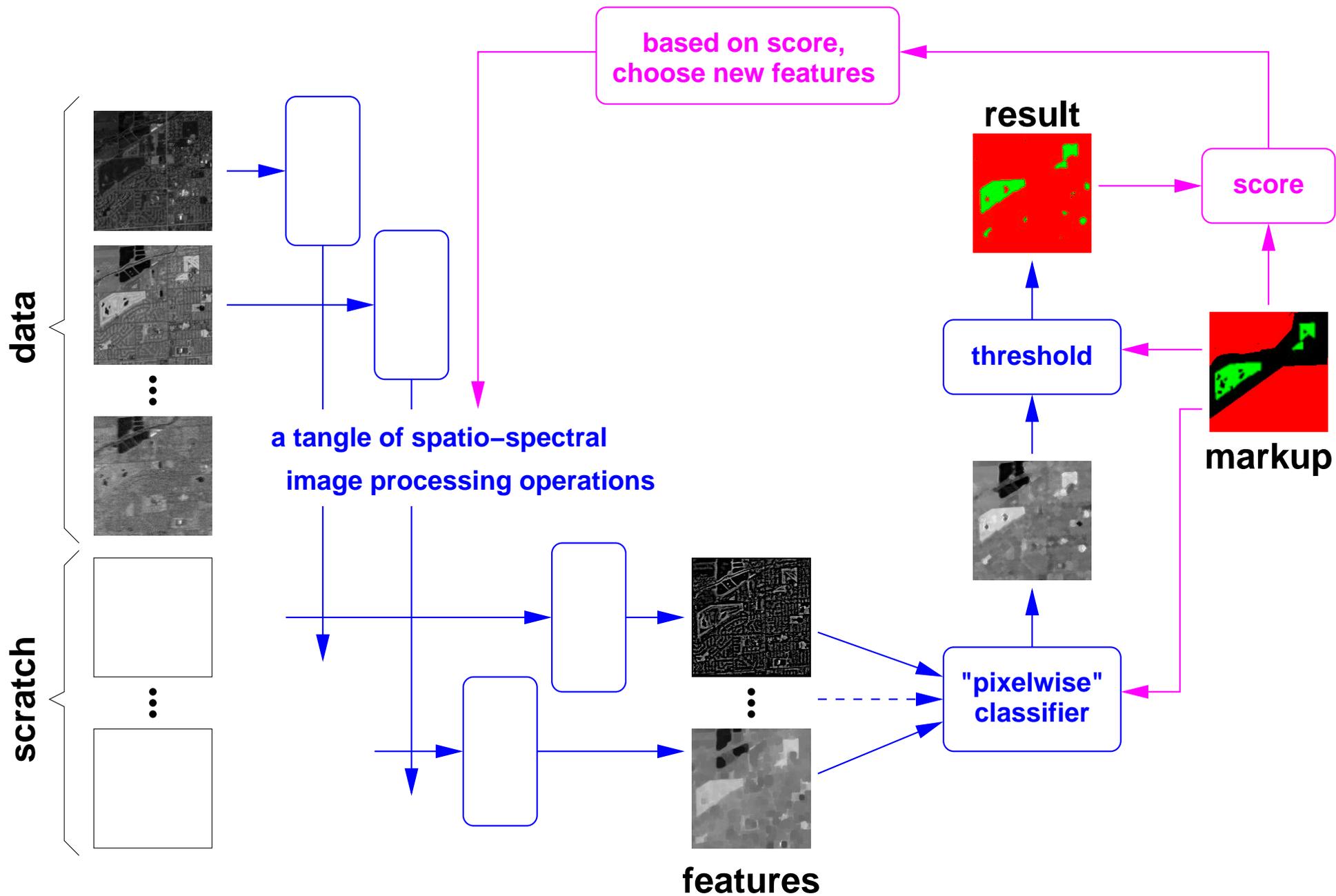
“Online” feature selection

■ Iterate:

- Spatio-spectral features created by applying short “pipelines” of elementary image-processing operators to the data
 - various convolutions: smooth, Sobel gradient, Gabor filter, ...
 - morphological operators: erode, dilate, open, close, ...



- spectral features: band ratio, band difference, clip, threshold, ...
- Fixed number of “scratch planes” hold currently employed features
 - Features are *always* spatially local
- Pixel-by-pixel classifier applied to scratch (and, optionally, data) planes
 - Pixels treated as independent samples
 - Fisher discriminant, support vector machine, *etc.*
- Based on performance, go back and choose new features



Contiguity Property for Images

- In remote sensing, what is of interest on the ground is usually much larger than a pixel
 - especially terrain categories: lakes, forests, beaches, vegetation canopies, agricultural fields, golf courses. . .
 - few exceptions: sub-pixel beacons, narrow roads



- Pixels are not IID samples!
 - Two adjacent pixels are more likely to come from the same class than are two pixels chosen at random from the image.
 - This is an almost universal property of images
 - How to quantify, how to exploit?

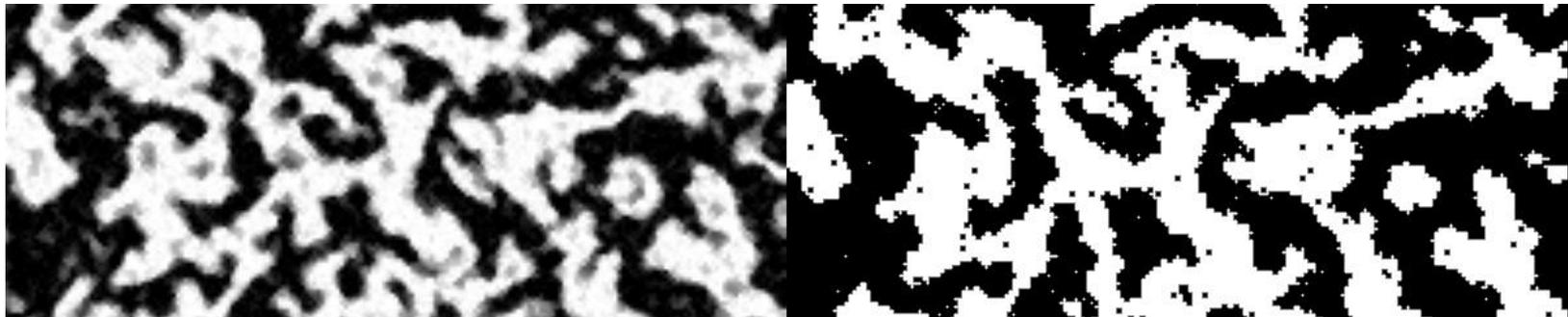
Markov Random Fields

- “If there was something called a noncausal 2D autoregressive model, MRF would be called that.” – Murat Dondar
- Bayesian Image Analysis
 - Prior: $P(I)$ is distribution over all possible images, not having seen the data
 - Data: D is an observed image
 - Think I =idealized, D =dirty
 - Likelihood: $P(D|I)$ is probability of observing data D , given that the true image is I
 - Bayes Rule: $P(I|D) = P(D|I)P(I)/P(D)$
 - Maximum a posteriori: $I_{\text{MAP}} = \text{argmax}_I P(D|I)P(I)$
- MRF is a prior that encourages adjacent pixels to be alike
 - Product of pixel probabilities: $P(I) = \prod_{ij} P_{ij}(x_{ij})$
 - Pixel probabilities only depend on neighbors (Markov property)

$$P_{ij}(x_{ij}) = P(x_{ij} | x_{i \pm \Delta i, j \pm \Delta j})$$

MRF's for texture synthesis

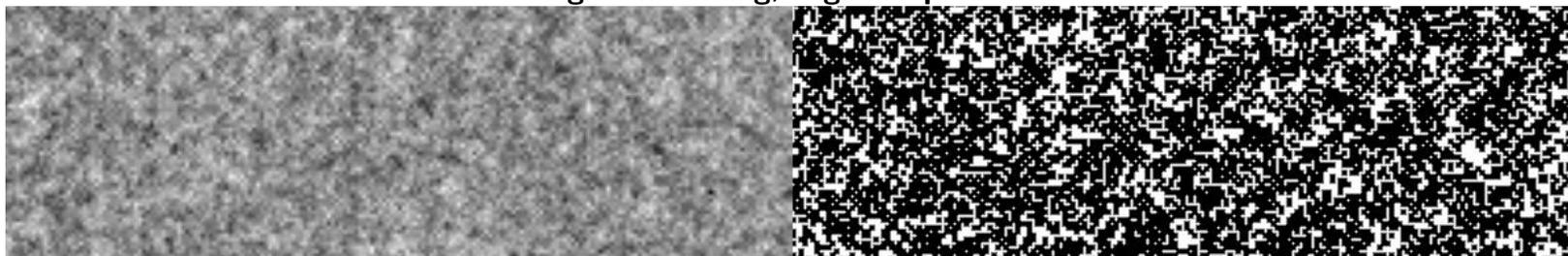
- No data; generate realizations of idealized images consistent with $P(I)$
- Algorithm begins with idealized image: $x \in \{-1, 1\}$
 - Smooth image: $x'_i = \sum_{\Delta i} w_{\Delta i} x_{i+\Delta i}$
 - Resample image probabilistically: $P(x = \pm 1) = \frac{e^{\pm\beta x'}}{e^{\beta x'} + e^{-\beta x'}}$
 - Iterate
- Characteristic properties of image “texture” a tradeoff
 - Strength and nature of smoothing: $w_{\Delta i}$
 - “Temperature” ($1/\beta$) of probabilistic resampling



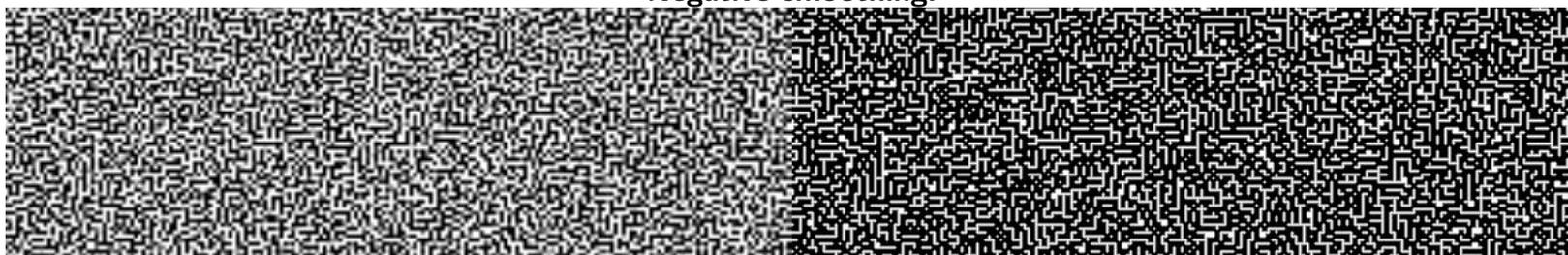
Left panel: Average over many trials;

Right panel: single realization.

Light smoothing, High temperature



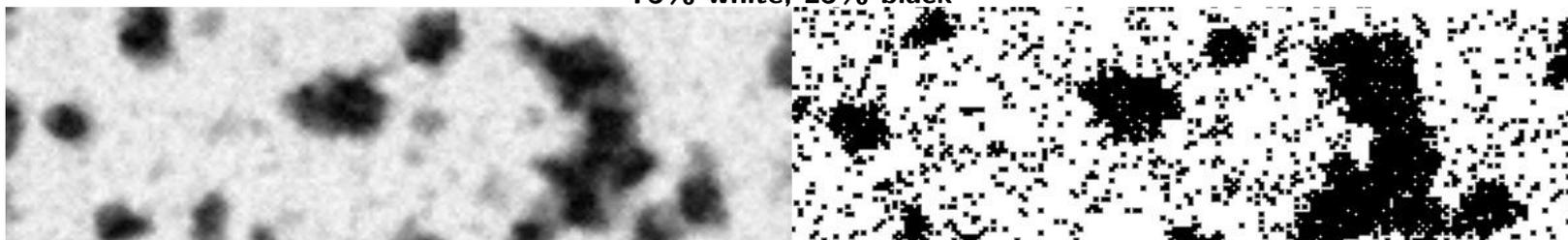
Negative smoothing.



Low temperature ($\beta = 5$)



75% white, 25% black



Markov Random Fields (cont'd)

■ Prior is MRF: $P(I) = \prod_{ij} P_{ij}(\mathbf{x}_{ij}) = \prod_{ij} P(\mathbf{x}_{ij} | \mathbf{x}_{i\pm\Delta i, j\pm\Delta j})$

■ Likelihood model: $P(D|I) = \prod_{ij} P(d_{ij} | \mathbf{x}_{ij})$

• Let $V_{ij}(\mathbf{x}_{ij}) = -\log P_{ij}(\mathbf{x}_{ij}) - \log P(d_{ij} | \mathbf{x}_{ij})$, so then

$$P(D|I)P(I) \propto \exp\left(-\sum_{ij} V_{ij}(d_{ij}, \mathbf{x}_{ij})\right) = \exp\left(-\sum_{ij} V(d_{ij}, \mathbf{x}_{ij}, \mathbf{x}_{i\pm\Delta i, j\pm\Delta j})\right)$$

■ Invoke thermodynamic analogy

• Call it a Gibbs distribution

• Introduce a temperature T

$$[P(D|I)P(I)]^{1/T} \propto \exp\left(-\sum_{ij} V(d_{ij}, \mathbf{x}_{ij}, \mathbf{x}_{i\pm\Delta i, j\pm\Delta j})/T\right)$$

• and note that as $T \rightarrow 0$, we have

$$[P(D|I)P(I)]^{1/T} \rightarrow \delta(I = I_{\text{MAP}})$$

■ Use Markov Chain Monte Carlo (MCMC) to explore space of images I distributed according to $\exp(-V(D, I)/T)$.

• Anneal $T \rightarrow 0$, and converge $I \rightarrow I_{\text{MAP}}$.

Markov Random Fields (an alternative viewpoint)

■ A lot of hullabaloo: $I_{\text{MAP}} = \operatorname{argmin}_I V(D, I)$.

■ Take a closer look at $V(D, I)$; typically

$$V(D, I) = \sum_{ij} V_{\text{DATA}}(d_{ij}, x_{ij}) + V_{\text{CONTIG}}(x_{ij}, x_{i\pm\Delta i, j\pm\Delta j})$$

- V_{DATA} is penalty for label to disagree with data
- V_{CONTIG} is penalty for neighboring pixels to have different labels

■ You can ask the practitioner to adopt a Bayesian point of view,

- produce an MRF that is appropriate for the imagery at hand,
- produce a model for likelihood of data given a label, and
- spend hours of computer time on huge MCMC runs. . .

■ Or you can ask for two cost functions:

- $V_{\text{DATA}}(d, x)$ expresses the cost of label x when the data is d
- V_{CONTIG} expresses the cost for neighboring pixels to have different labels

Contiguity and Support Vector Machines

- Add a simple V_{CONTIG} term to the SVM loss function
 - Quadratic term does not affect mathematical structure of loss function, so SVM algorithm can still be used
 - Provably bounded computation to find optimum
 - Inherit SVM's resistance to overfitting(?)
 - Contiguity term does not require labelled data
- For linear SVM, effect of contiguity can be expressed as
 - Linear preprocessing of data: $\mathbf{z} = \mathbf{D}_\lambda^{-1/2} \mathbf{x}$, or
 - Modified linear kernel: $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{D}_\lambda^{-1} \mathbf{x}'$
 - Suggests interpretation of contiguity property in terms of an invariance-preserving kernel, as advocated by Schölkopf and Smola, *Learning with Kernels*, Chapter 11.
 - Solution should be “approximately” invariant to the operation of moving to a neighboring pixel
- Method can also be employed for nonlinear kernels

Contiguity and Linear Support Vector Machines

■ Decision function: $f(\mathbf{x}) = \beta^T \mathbf{x} + \beta_o$

■ Ordinary SVM loss function

$$L(\beta, \beta_o) = \frac{1}{2} \beta^T \beta + C \sum_i \varepsilon_i$$

• where $\varepsilon_i = \max(0, 1 - y_i f(\mathbf{x}_i))$

■ Contiguity cost

$$V_{\text{CONTIG}}(\mathbf{x}_i, \mathbf{x}_{i \pm \Delta i}) = \sum_{\Delta i} |\mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_{i \pm \Delta i})|^2 = \sum_{\Delta i} \beta^T (\mathbf{x}_i - \mathbf{x}_{i \pm \Delta i})(\mathbf{x}_i - \mathbf{x}_{i \pm \Delta i})^T \beta$$

■ Contiguity matrix

$$D = \frac{2}{NN_{\Delta}} \sum_{i=1}^N \sum_{\Delta i} (\mathbf{x}_i - \mathbf{x}_{i \pm \Delta i})(\mathbf{x}_i - \mathbf{x}_{i \pm \Delta i})^T$$

• where N_{Δ} is number of neighbors (we use $N_{\Delta} = 8$)

• weighted neighborhoods would be a straightforward generalization

■ Contig-SVM loss function

$$L(\beta, \beta_o) = \frac{1}{2} \beta^T \beta + C \sum_i \varepsilon_i + \frac{1}{2} \kappa \beta^T D \beta$$

Linear contig-SVM (cont'd)

■ Loss function

$$L(\beta, \beta_o) = \frac{1}{2}\beta^T\beta + C \sum_i \varepsilon_i + \underbrace{\frac{1}{2}\kappa\beta^T D\beta}$$

- Let $\lambda = \kappa/(\kappa + 1)$ and $C^* = (\kappa + 1)C$, and
- Let $D_\lambda = \lambda D + (1 - \lambda)I$ be “regularized” contiguity matrix

■ Rewrite loss function

$$L(\beta, \beta_o) = \frac{1}{2}\beta^T D_\lambda \beta + C^* \sum_i \varepsilon_i$$

■ Regularized contiguity matrix is positive definite: write $D_\lambda = U\Lambda_\lambda U^T$

- let $\beta^* = D_\lambda^{1/2}\beta = U\Lambda_\lambda^{1/2}U^T\beta$, and
- let $\mathbf{z} = \phi(\mathbf{x}) = D_\lambda^{-1/2}\mathbf{x} = U\Lambda_\lambda^{-1/2}U^T\mathbf{x}$

■ Then loss function becomes

$$L(\beta^*, \beta_o) = \frac{1}{2}\beta^{*T}\beta + C^* \sum_i \varepsilon_i$$

- ### ■ Solution: $f(\mathbf{x}) = \left(\sum_i \alpha_i y_i \mathbf{x}_i^T D_\lambda^{-1} \right) \mathbf{x} + \beta_o = \sum_i \alpha_i y_i \left(\mathbf{x}_i^T D_\lambda^{-1} \mathbf{x} \right) + \beta_o$

Kernelized contig-SVM

- Map $\mathbf{x} \rightarrow \phi(\mathbf{x})$
- Contiguity matrix in mapped space:

$$\mathbf{D}^\phi = \frac{2}{NN_\Delta} \sum_{i=1}^N \sum_{\Delta i} (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_{i+\Delta i}))(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_{i+\Delta i}))^T$$

- Want use contiguity enhanced kernel

$$\mathbf{K}_{\text{CONTIG}}(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \mathbf{D}_\lambda^{\phi-1} \phi(\mathbf{x}')$$

- But need to evaluate $\mathbf{K}_{\text{CONTIG}}(\mathbf{x}, \mathbf{x}')$ without evaluating $\phi(\mathbf{x})$ directly.
 - Can't evaluate $\phi(\mathbf{x})$, but can evaluate $\mathbf{K}(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$
 - this is just the ordinary (non-contig-enhanced) kernel
 - For convenience, also write $\mathbf{K}_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$
 - Need to express $\mathbf{K}_{\text{CONTIG}}(\mathbf{x}, \mathbf{x}')$ in terms of $\mathbf{K}(\mathbf{x}, \mathbf{x}')$
-
- Question: can we preprocess the data and then just use the ordinary kernel?
 - Answer: I don't think so

Simpler (but untested) idea: contig-LDA

■ Linear Discriminant Analysis (e.g., Fisher discriminant)

- Data \mathbf{x}_i and associated class label y_i
- Class means: $\mathbf{x}_y = \sum_i \mathbf{1}_{y_i=y} \mathbf{x}_i / \sum_i \mathbf{1}_{y_i=y}$
- Within class covariance: $\mathbf{S}_w = \sum_i (\mathbf{x}_i - \mathbf{x}_{y_i})(\mathbf{x}_i - \mathbf{x}_{y_i})^T$
- Between class covariance: $\mathbf{S}_b = \sum_{y,y'} (\mathbf{x}_y - \mathbf{x}_{y'})(\mathbf{x}_y - \mathbf{x}_{y'})^T$
- Discriminant functions: $f(\mathbf{x}) = \beta^T \mathbf{x}$
 - Want small within-class $\beta^T \mathbf{S}_w \beta$ and large between-class: $\beta^T \mathbf{S}_b \beta$.
 - e.g., max $\beta^T \mathbf{S}_b \beta$ subject to $\beta^T \mathbf{S}_w \beta = 1$.
 - Discriminant(s) given by largest eigenvector(s) of $\mathbf{S}_w^{-1} \mathbf{S}_b$

■ Now add a contiguity term

- Want small $\sum_i \sum_{\Delta i} |f(\mathbf{x}_i) - f(\mathbf{x}_{i+\Delta i})|^2$; that is: small $\beta^T \mathbf{D} \beta$
- Suggests choosing eigenvectors of $((1 - \lambda) \mathbf{S}_w + \lambda \mathbf{D})^{-1} \mathbf{S}_b$

Contiguity and Unsupervised Classification (clustering)

■ k-means (Lloyd, 1958)

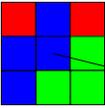
- Class labels assigned to pixels: $\mathbf{x}_{ij} \in \{1, \dots, K\}$
- Class means $\mathbf{d}_k = \sum_{ij} \mathbf{d}_{ij} \delta(\mathbf{x}_{ij}, k) / \sum_{ij} \delta(\mathbf{x}_{ij}, k)$
- Minimize in-class variance: $V_{\text{DATA}}(\mathbf{d}_{ij}, \mathbf{x}_{ij}) = (\mathbf{d}_{ij} - \mathbf{d}_{\mathbf{x}_{ij}})^2$

■ contig-k-means (Theiler and Gisler, 1997)

- Introduce a penalty term for “discontiguity” among the labels

$$V_{\text{CONTIG}}(\mathbf{x}_{ij}, \mathbf{x}_{i\pm\Delta i, j\pm\Delta j}) = \kappa \sum_{\Delta i, \Delta j} (1 - \delta(\mathbf{x}_{ij}, \mathbf{x}_{i\pm\Delta i, j\pm\Delta j}))$$

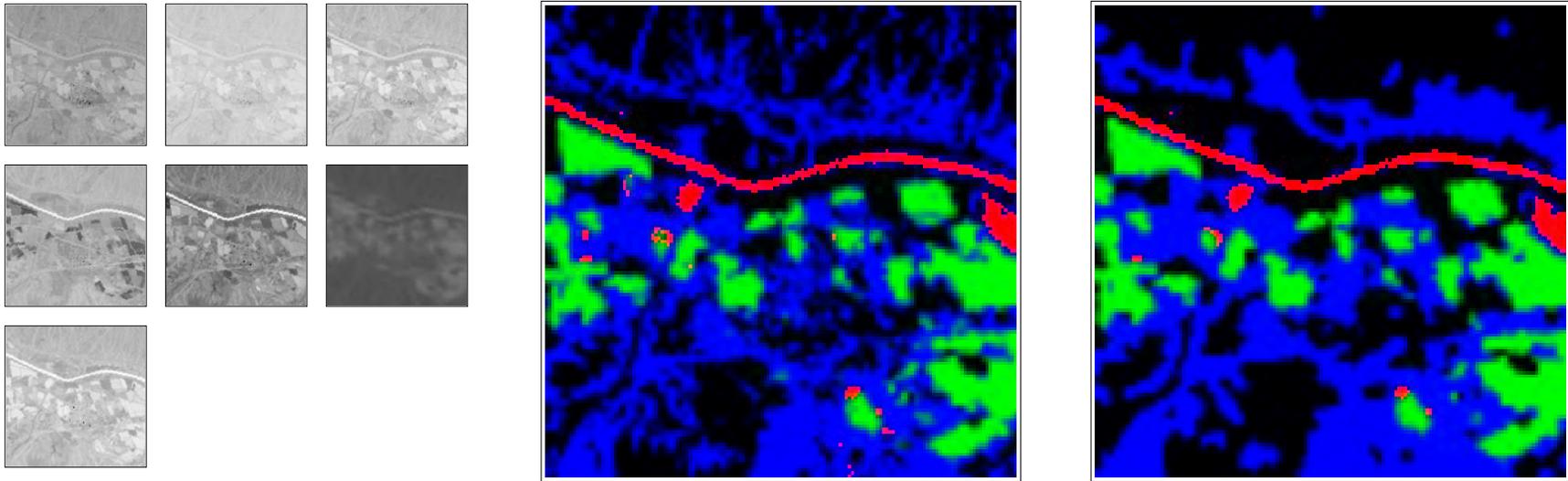
- where κ is the κ knob that adjusts the relative importance of contiguity compared to in-class variance

- For example:  $V_{\text{CONTIG}}(\mathbf{x}_{ij}) = 5\kappa$

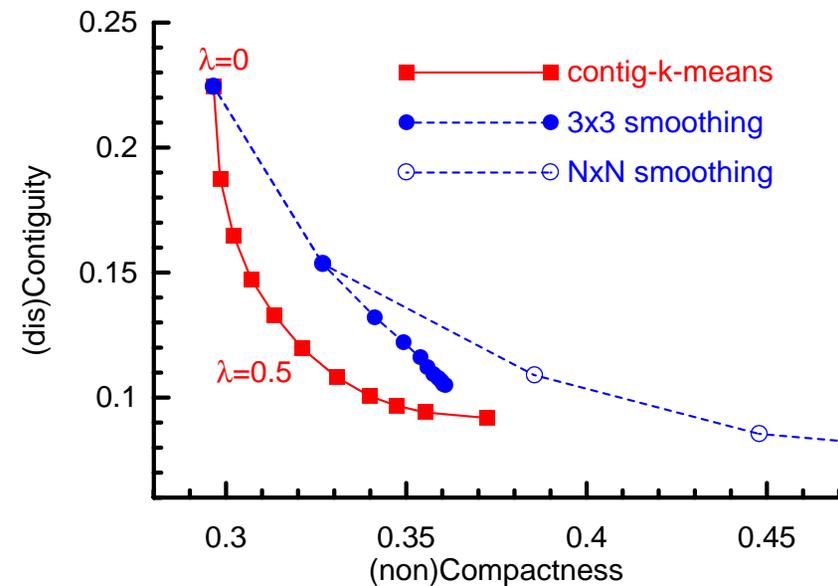
- Produce an iterative k-means-like algorithm to minimize

$$V(D, I) = \sum_{ij} V_{\text{DATA}}(\mathbf{d}_{ij}, \mathbf{x}_{ij}) + V_{\text{CONTIG}}(\mathbf{x}_{ij}, \mathbf{x}_{i\pm\Delta i, j\pm\Delta j})$$

Contiguity vs. Compactness



- Multispectral Landsat-TM imagery
- Seven-band 151×151 -pixel image of Hatch, New Mexico
- Segmentation into $K = 4$ classes
 - from 56 bits/pixel to 2 bits/pixel
- Can substantially enhance contiguity at minimal cost to in-class variance



Conclude

- Many remote sensing problems can benefit from machine learning tools
 - In many of these problems, Image Analysis plays a central role



- But images are not collections of independent pixels
- Focus-of-attention problems
- Use of Spatial features
 - Select from a large pool of potential features
 - Online schemes for “growing” successful features
- Exploitation of Contiguity property
 - Markov Random Field approaches
 - Direct use of discontiguity penalty function: $V_{\text{CONTIG}} \sim |f(\mathbf{x}_i) - f(\mathbf{x}_{i+\Delta i})|^p$
 - Convexity is a virtue: $p \geq 1$
 - Robustness is a virtue: $p \leq 1$
 - Preserving mathematical structure is a virtue: $p = 2$

