1 Learning Rates

Heuristics on how to choose the learning rate:

- constant: $\eta_t = 10^{-3}$ / decreasing: $\eta_t = \max\left\{10^{-2}, \frac{1}{t}\right\}$
- adaptive: $\eta_t = \arg\min_\eta \hat{R}(\mathbf{w}_t - \eta \mathbf{g}_t)$ (via 1D-pt. problem)
- bold-driver: $\eta_{t+1} := \hat{R}(\mathbf{w}_{t+1}) < \hat{R}(\mathbf{w}_t)\ ?\ \eta_t \cdot c_{\text{inc}} : \eta_{t-1} \cdot c_{\text{dec}}$

2 Regression

2.1 Ridge Regression

$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$ ($\lambda > 0$, chosen via CV)

1) Closed Form $\mathcal{O}(nd^2 + d^3)$ (setup + solve)

$\mathbf{w}^* = (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$ (always has a solution)

2) Gradient descent $\mathcal{O}(\text{iter.} \times nd)$

$\mathbf{g}_t = -2\mathbf{X}^\mathsf{T}(\mathbf{y} - \mathbf{X}\mathbf{w}_t) + 2\lambda \mathbf{w}_t$

Note: Now the scale of the data matters for λ! ($\to$ normalize data)

1) VS 2) Complexity, Optimality of Sol., CF possible (enoug data)?

Bayesian Interpretation (=Gaussian MAP) Implicit assumption: label y is linear in $\mathbf{x}$, with Gaussian noise with *constant* variance.

$Y \sim \mathcal{N}(\mathbf{w}^\mathsf{T}\mathbf{x}, \sigma^2)$, $y_i = \mathbf{w}^\mathsf{T}\mathbf{x}_i + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2)$

$P(Y = y \mid \mathbf{x} = \mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y; h(\mathbf{x}), \sigma^2)$, $h(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$, $\boldsymbol{\theta} = (\mathbf{w}, \sigma^2)$

weights prior: $\mathbf{w} \sim \mathcal{N}(0, \beta^2 \mathbf{I})$, $w_i \sim \mathcal{N}(0, \beta^2)$

Maximizing $P(\mathbf{w} \mid D)$ then leats to the connection $\lambda = \frac{\sigma^2}{\beta^2}$.

2.2 Kernelized Ridge Regression

Insight optimal $\mathbf{w}^*$ lies in the span of the data.

$\mathbf{w}^* = \mathbf{X}_\phi^\mathsf{T}\mathbf{z}^*$ $(\mathbf{K} = \mathbf{X}_\phi \mathbf{X}_\phi^\mathsf{T} \in \mathbb{R}^{n \times n})$

$\mathbf{z}^* = \arg\min_{\mathbf{z}} \|\mathbf{K}\mathbf{z} - \mathbf{y}\|_2^2 + \lambda \mathbf{z}^\mathsf{T}\mathbf{K}\mathbf{z}$

1) Closed form $\mathbf{z}^* = (\mathbf{X}_\phi \mathbf{X}_\phi^\mathsf{T} + \lambda \mathbf{I})^{-1}\mathbf{y} = (\mathbf{K} - \lambda \mathbf{I})^{-1}\mathbf{y}$

2) Gradient descent $\mathbf{g}_t = 2\mathbf{K}^\mathsf{T}(\mathbf{K}\mathbf{z} - \mathbf{y}) + 2\lambda \mathbf{K}\mathbf{z}$

Prediction $f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\phi(\mathbf{x}) = \ldots = \sum_{i=1}^n z_i k(\mathbf{x}_i, \mathbf{x})$

Bayesian Interpretation Same as ridge regression, except that the hypothesis class for $\mathcal{H}$ for h (comp. of mean) may be different.

2.3 Sparse Regression: LASSO

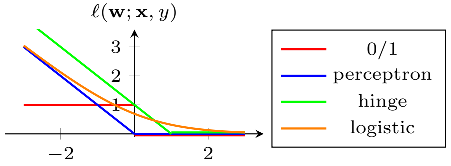
Prior: $w_i \sim p(w_i; 0, b) = \frac{1}{2b} e^{-\frac{|w_i - \mu|}{b}}$ where $\mu = 0$, (connection: $\lambda = \frac{2\sigma^2}{b}$).

2.4 Regression with Outliers

Use losses smaller than squared loss, or distributions with fatter tails.

3 Classification

3.1 Classification Losses



0/1 Loss

$\ell_{0/1}(\mathbf{w}; \mathbf{x}, y) = \mathbb{1}_{\{y_i \neq \text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x})\}} = \begin{cases} 0 & y = \text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x}) \\ 1 & \text{otherwise} \end{cases} = \begin{cases} 0 & y\,\text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x}) = 1 \\ 1 & \text{otherwise} \end{cases}$

3.2 Perceptron

$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^n \max\left\{0, -y_i \mathbf{w}^\mathsf{T}\mathbf{x}_i\right\}$

$\mathbf{g}_t = \sum_{i=1}^n \begin{cases} 0, & -y_i \mathbf{w}_t^\mathsf{T}\mathbf{x}_i < 0, \\ -y_i \mathbf{x}_i, & \text{otherwise.} \end{cases} = -\mathbf{X}^\mathsf{T}(\mathbf{y} \odot [-\mathbf{y} \odot \mathbf{X}\mathbf{w} \geq 0])$

3.3 Kernelized Perceptron

Ansatz $\mathbf{w}^* = \sum_{j=1}^n \alpha_j y_j \phi(\mathbf{x}_j) = \mathbf{X}_\phi^\mathsf{T}(\boldsymbol{\alpha} \odot \mathbf{y})$ gives:

$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \sum_{i=1}^n \max\left(0, -y_i \boldsymbol{\alpha}^\mathsf{T}\mathbf{k}_i\right)$,

where $\mathbf{k}_i = (y_1 k(\mathbf{x}_i, \mathbf{x}_1), \ldots, y_n k(\mathbf{x}_i, \mathbf{x}_n))^\mathsf{T}$.

Gradient Step: Equiv. between updating $\mathbf{w}$ and $\boldsymbol{\alpha}$

if $y_i \mathbf{w}^\mathsf{T}\mathbf{x}_i \geq 0$:
  $\mathbf{w}_t = \mathbf{w}_{t-1}$
else:
  $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} + \eta_t y_i \phi(\mathbf{x}_i)$
  $= \sum_{j=1}^n \alpha_j^{(t-1)} y_j \phi(\mathbf{x}_j) + \eta_t y_i \phi(\mathbf{x}_i)$
  $= \sum_{j=1}^n \begin{cases} \alpha_j^{(t-1)} y_j \phi(\mathbf{x}_j), & i \neq j \\ (\alpha_j^{(t-1)} + \eta_t) y_i \phi(\mathbf{x}_i), & i = j \end{cases}$

if $y_i \sum_{j=1}^n \alpha_j y_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$:
  $\boldsymbol{\alpha}^{(t)} \leftarrow \boldsymbol{\alpha}^{(t-1)}$
else:
  $\boldsymbol{\alpha}_j^{(t)} \leftarrow \boldsymbol{\alpha}_j^{(t-1)}$ for all $j \neq i$
  $\boldsymbol{\alpha}_i^{(t)} \leftarrow \boldsymbol{\alpha}_i^{(t-1)} + \eta_t$ for $i = j$

Pred. $f(\mathbf{x}) = \text{sign}\left(\mathbf{w}^{*\mathsf{T}}\phi(\mathbf{x})\right) = \ldots = \text{sign}\left(\sum_{j=1}^n \alpha_j^* y_j k(\mathbf{x}_j, \mathbf{x})\right)$

3.4 Support Vector Machines (SVMs)

$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^n \max\left\{0, 1 - y_i \mathbf{w}^\mathsf{T}\mathbf{x}_i\right\} + \lambda \|\mathbf{w}\|_2^2$ (reg. by default)

Hinge Loss (ℓ_{SVM}) maximizes margin of separator.

$\mathbf{g}_t = \sum_{i=1}^n \begin{cases} 0, & 1 - y_i \mathbf{w}_t^\mathsf{T}\mathbf{x}_i < 0, \\ -y_i \mathbf{x}_i, & \text{otherwise.} \end{cases} + 2\lambda \mathbf{w}_t$

$= -\mathbf{X}^\mathsf{T}(\mathbf{y} \odot [1 - \mathbf{y} \odot \mathbf{X}\mathbf{w} \geq 0]) + 2\lambda \mathbf{w}_t$

3.5 Kernelized Support Vector Machines

$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \sum_{i=1}^n \max\left\{0, 1 - y_i \boldsymbol{\alpha}^\mathsf{T}\mathbf{k}_i\right\} + \lambda \boldsymbol{\alpha}^\mathsf{T}\mathbf{K}\boldsymbol{\alpha}$

where $\mathbf{k}_i = (y_1 k(\mathbf{x}_i, \mathbf{x}_1), \ldots, y_n k(\mathbf{x}_i, \mathbf{x}_n))^\mathsf{T}$.

3.6 Nearest Neighbor Classifiers (k-NN)

$y = \text{sign}\left(\sum_{i=1}^n y_i \mathbb{1}_{\{\mathbf{x}_i \text{ among } k \text{ nearest neighbors of } \mathbf{x}\}}\right)$ (choose k via CV)

3.7 Logistic Regression

$P(Y = y \mid \mathbf{x}, \mathbf{w}) = Ber(y; \sigma(\mathbf{w}^\mathsf{T}\mathbf{x})) = \frac{1}{1 + e^{-y\mathbf{w}^\mathsf{T}\mathbf{x}}} = p_y$

$P(Y = +1 \mid \mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^\mathsf{T}\mathbf{x}) = p_+$

Grad. step with Gaussian Prior: $\mathbf{w} \leftarrow \mathbf{w}(1 - 2\lambda\eta_t) + \eta_t y \mathbf{x} \hat{P}(Y = -y \mid \mathbf{w}, \mathbf{x})$

3.7.1 Multi-Class Logistic Regression

$P(Y = i \mid \mathbf{x}, \mathbf{w}_1, \ldots, \mathbf{w}_c) = \frac{\exp(\mathbf{w}_i^\mathsf{T}\mathbf{x})}{\sum_{j=1}^c \exp(\mathbf{w}_j^\mathsf{T}\mathbf{x})} = p_i$

4 Kernels

4.1 Definition of a Kernel

For a data space $\mathcal{X}$ a *kernel* is a function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ satisfying *i)* and [*ii)* or *iii)*]:

i) *symmetry*: $\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$: $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$

ii) *positive semi-definiteness*: for any n, any set $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq \mathcal{X}$, the kern G. matrix $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{1 \leq i, j \leq n}$ must be p. sem. def.

iii) k is an inner product $\langle \cdot, \cdot \rangle : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ in a suitable space $\mathcal{F}$ (where $\Phi\colon \mathcal{X} \to \mathcal{F}$ is the feature map)

4.2 Common Kernels

- Linear kernel: $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\mathsf{T}\mathbf{x}'$
- Monomials of degree m: $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\mathsf{T}\mathbf{x}')^m$
- Monomials up to degree m: $k(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^\mathsf{T}\mathbf{x}')^m$
- Gaussian (RBF, Sq. exp. kernel) $k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|_2^2 / h^2)$
- Sigmoid (tanh) kernel: $k(\mathbf{x}, \mathbf{x}') = \tanh(\kappa \mathbf{x}^\mathsf{T}\mathbf{x}') - b$
- Laplacian kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|_1)$

4.3 Kernel Composition Rules

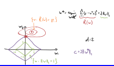
$k_1 + k_2$ / $k_1 \cdot k_2$ / $c \cdot k_1$ $(c > 0)$ / $f(k_1)$, f poly. pos. coeff, exponential.

5 Feature Selection

Greedy FW $s_i = \arg\min_{j \in V \setminus S} \hat{L}(S \cup \{j\})$

Greedy BW $s_i = \arg\min_{j \in S} \hat{L}(S \setminus \{j\})$

Linear Models: Sparsity Trick: $\|\mathbf{w}\|_0 \to \|\mathbf{w}\|_1$



6 Imbalanced Data

Convention: + is the rare class.

Possible Approaches: Upsampling / Downsampling. Or choosing classifier based on trade-offs / evaluation metrics:

Cost-Sensitive Loss $R(\mathbf{w}) = \sum_{i=1}^n c_{y_i} \min(0, -y_i \mathbf{w}^\mathsf{T}\mathbf{x}_i)$.

$c_+ > 0$ (cost for mispredicting positive class), w.l.o.g. set $c_- = 1$

		True Label Pos.	True Label Neg.	
Pred. Label	**Pos.**	**TP**	**FP**	$\sum = p_+$
	Neg.	**FN**	**TN**	$\sum = p_-$
		$\sum = n_+$	$\sum = n_-$	

(n_+ #positive instances, p_+ "#predicted as +")

$n = n_+ + n_- = p_+ + p_- = \text{TP} + \text{FP} + \text{FN} + \text{FP}$

first letter: whether prediction was correct. *second letter*: prediction.

Accuracy $\frac{TP+TN}{TP+TN+FP+FN} = \frac{TP+TN}{n}$

Precision (for class + or P) $\frac{TP}{TP+FP} = \frac{TP}{p_+} \in [0, 1]$

Recall (for class + or P) $\frac{TP}{TP+FN} = \frac{TP}{n_+} \in [0, 1]$

F1 Score, F-Measure $\frac{2}{\frac{1}{\text{Prec.}} + \frac{1}{\text{Rec.}}} = \frac{2TP}{2TP+FP+FN} \in [0, 1]$

TPR, True Pos. Rate $\frac{TP}{TP+FN} = \frac{TP}{n_+}$ (= Recall for +)

FPR, False Pos. Rate $\frac{FP}{TN+FP} = 1 - \frac{TN}{TN+FP} = 1 - \frac{TN}{n_-}$ (1− Recall for−)

Approaches to pick parameter c_+ or vary treshold $y = \text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x} - \tau)$

- **Precision Recall Curve**: x: Precision, y: Recall. Ideal: parameters in upper right corner.
- **Receiver Operator Characteristic (ROC) Curve**: x: FPR, y: TPR, Ideal: Classifier in upper left. Random classifier: Diagonal from lower left to upper right

7 Multiclass Classification

One-VS-All $y = \arg\max_{i \in \{1, \ldots, c\}} f_i(\mathbf{x})$ (e.g., $f_i(\mathbf{x}) = \frac{\mathbf{w}_i^\mathsf{T}\mathbf{x}}{\|\mathbf{w}_i\|_2}$)

One-VS-One Train $\binom{c}{2}$ bin. clf. for each pair $(i, j) \in \{1, \ldots, c\}^2$.

$f_{(i,j)} : \mathcal{X} \to \{-1, +1\}$ $y = \arg\max_{i \in \{1, \ldots, c\}} \sum_{j=1 j \neq i}^c \mathbb{1}_{\{f_{(i,j)}(\mathbf{x}) = +1\}}$.

Alternative Methods

- Encode label binary, build Clf. for each bit, (use err. corr. codes)
- Use multi-class models (Multc. Perceptron, Gen. Models)

8 Neural Networks

8.1 Losses

- One output: usual losses: perceptron, hinge, squard loss, …
- Multiple outputs: then we usually define the loss as a *sum of per-output loss*: $L = \sum_{k=1}^p \ell_k(f_k(\mathbf{W}, \mathbf{x}), \mathbf{y})$ or use the *cross entropy* loss: $\ell(Y = i; f_1, \ldots, f_c) = -\log \frac{\exp(f_i)}{\sum_{j=1}^c \exp(f_j)}$.

8.2 Backward Propagation

$\mathbf{W} \leftarrow \mathbf{W} - \eta_t \nabla_{\mathbf{W}} \ell(\mathbf{W}; \mathbf{y}, \mathbf{x})$

Output Layer Gradient $\ell = L + 1$

$\delta_i^{(L+1)} = \frac{\partial \ell_i}{\partial f_i}$ $\boldsymbol{\delta}^{(L+1)} = \nabla_f L$

Hidden Layer Gradient / Error Gradient $\ell = L : -1 : 1$

$\delta_j^{(\ell)} = \varphi'(z_j^{(\ell)}) \sum_{k=1}^{m_{\ell+1}} w_{k,j}^{(\ell+1)} \delta_k^{(\ell+1)} = \varphi'(z_j^{(\ell)}) \cdot \left(\mathbf{W}^{(\ell+1)} \boldsymbol{\delta}^{(\ell+1)}\right)_j$

$\boldsymbol{\delta}^{(\ell)} = \boldsymbol{\varphi}'(\mathbf{z}^{(\ell)}) \odot \left(\mathbf{W}^{(\ell+1)} \boldsymbol{\delta}^{(\ell+1)}\right)$ $\delta_j^{(\ell)} = \frac{\partial L}{\partial z_j^{(\ell)}} = \frac{\partial L}{\partial b_j^{(\ell)}}$

$\frac{\partial L}{\partial w_{i,j}^{(\ell)}} = v_j^{(\ell-1)} \delta_i^{(\ell)}$ $(v_{\text{in}} \cdot \delta_{\text{out}})$ $\nabla_{\mathbf{W}^{(\ell)}} L = \frac{\partial L}{\partial \mathbf{W}^{(\ell)}} = \boldsymbol{\delta}^{(\ell)} \mathbf{v}^{(\ell-1)\mathsf{T}}$

8.3 Activation Functions

- **Sigmoid** $\varphi(z) = \frac{1}{1 + e^{-z}} \in (0, 1)$, $\varphi'(z) = \varphi(z)(1 - \varphi(z))$
- **Tanh** $\varphi(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \in (-1, 1)$, $\varphi'(z) = 1 - \tanh^2(z)$
- **ReLU** $\varphi(z) = \max(z, 0) \in [0, \infty)$, $\varphi'(z) = \mathbb{1}_{\{z > 0\}}$

9 Clustering (Unsupervised Classification)

9.1 k-Means

$L(\boldsymbol{\mu}) = L(\mu_1, \ldots, \mu_k) = \sum_{i=1}^n \underbrace{\min_{j \in \{1, \ldots, k\}} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_2^2}_{\ell(\mathbf{x}_i; \boldsymbol{\mu})}$ (non convex)

$(\mathbf{W}^*, \mathbf{z}_1^*, \ldots, \mathbf{z}_n^*) = \arg\min_{(\mathbf{W}, \mathbf{z}_1, \ldots, \mathbf{z}_n)} \sum_{i=1}^n \|\mathbf{W}\mathbf{z}_i - \mathbf{x}_i\|_2^2$

where $\mathbf{W} \in \mathbb{R}^{d \times k}$ is arbitrary, $\mathbf{z}_1, \ldots, \mathbf{z}_n \in E_k = \{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$.

Note that $\mathbf{e}_i = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$ denotes the i-th unit vector.

Assign: $z_i^{(t)} \leftarrow \arg\min_{j \in \{1, \ldots, k\}} \|\mathbf{x}_i - \mu_j^{(t-1)}\|_2^2$

Update: $\mu_j^{(t)} \leftarrow \frac{1}{n_j} \sum_{i: z_i^{(t)} = j} \mathbf{x}_i$ (where: $n_j = \left|\left\{\mathbf{x}_i : z_i^{(t)} = j\right\}\right|$

Initialization: Multiple random restarts, k-Means++: select every $\boldsymbol{\mu}_i = \mathbf{x}_j$ with probability proportional to distance of $\mathbf{x}_j$ to closest centroid.

CV doesn't work (Elbow-Heuristic, Regularization $+\lambda k$)

10 Dim. Reduction (Unsup. Regression)

Given $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq \mathbb{R}^k$

Goal obtain "embedding" (low-dim. represent.) $\{\mathbf{z}_1, \ldots, \mathbf{z}_n\} \subseteq \mathbb{R}^k$

10.1 Principal Component Analysis (PCA)

$(\mathbf{W}^*, \mathbf{z}_1^*, \ldots, \mathbf{z}_n^*) = \arg\min_{(\mathbf{W}, \mathbf{z}_1, \ldots, \mathbf{z}_n)} \sum_{i=1}^n \|\mathbf{W}\mathbf{z}_i - \mathbf{x}_i\|_2^2$

where $\mathbf{W} \in \mathbb{R}^{d \times k}$ is orthogonal ($\mathbf{W}^\mathsf{T}\mathbf{W} = \mathbf{I}_k$), and $\mathbf{z}_1, \ldots, \mathbf{z}_n \in \mathbb{R}^k$. (Orthogonality of $\mathbf{W}$ implies that the col. vectors have unit-length.)

Closed Form: Given $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq \mathbb{R}^d$ (w.l.o.g. we assume $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i = \mathbf{o}$), and $1 \leq k \leq d$. Then we build $\boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\mathsf{T} = \frac{1}{n}\mathbf{X}\mathbf{X}^\mathsf{T}$ (where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are the *colums* of $\mathbf{X}$). Then we diagonalize $\boldsymbol{\Sigma} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\mathsf{T} = \sum_{i=1}^d \lambda_i \mathbf{v}_i \mathbf{v}_i^\mathsf{T}$, where $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$. Then the optimal solution is: $\mathbf{W}^* = (\mathbf{v}_1, \ldots, \mathbf{v}_k)$ of $\mathbf{V}$ and the low-dimensional approximation is: $\mathbf{z}_i^* = \mathbf{f}(\mathbf{x}_i) = \mathbf{W}^\mathsf{T}\mathbf{x}_i$.

Com. $\mathbf{W}\mathbf{W}^\mathsf{T}$ is an orthogonal projection onto the col space of $\mathbf{W}$.

10.2 Kernel PCA

$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha},\, \boldsymbol{\alpha}^\intercal \mathbf{K}\boldsymbol{\alpha}=1} \boldsymbol{\alpha}^\intercal \mathbf{K}^\intercal \mathbf{K}\boldsymbol{\alpha}$ (for $k=1$)

Closed Form: For $k \geq 1$: Build $\mathbf{K}$. Center it $\mathbf{K}' = \mathbf{K} - \mathbf{E}_n\mathbf{K} + \mathbf{K}\mathbf{E}_n + \mathbf{E}_n\mathbf{K}\mathbf{E}_n$ $((\mathbf{E}_n)_{ij} = (1))$ diagonalise it $\mathbf{K}' = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\intercal$. Then $\boldsymbol{\alpha}^{(1)}, \dots, \boldsymbol{\alpha}^{(k)} \in \mathbb{R}^n$, where $\boldsymbol{\alpha}^{(i)} = \frac{1}{\lambda_i}\mathbf{v}_i$ $(\lambda_1 \geq \dots \geq \lambda_k)$.

Compression: $\mathbf{x} \mapsto \mathbf{z} = (z_1, \dots, z_k)$, $z_i = \mathbf{w}^\intercal \phi(\mathbf{x}) = \sum_{j=1}^n k(\mathbf{x}, \mathbf{x}_j)\alpha_j^{(i)}$

Disadv.: Non-param. (growth of kernel-gram matrix). Kernel unknown.

10.3 Autoencoders

Key idea: Try to learn the *identity function*!

$\mathbf{x} \approx \mathbf{f}(\mathbf{x};\boldsymbol{\theta})$ where $\mathbf{f}(\mathbf{x};\boldsymbol{\theta}) = \mathbf{f}_2(\mathbf{f}_1(\mathbf{x};\boldsymbol{\theta}_1);\boldsymbol{\theta}_2)$, and so $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$,

$\mathbf{f}_1 : \mathbb{R}^d \to \mathbb{R}^k$ "encoding", $\quad \mathbf{f}_2 : \mathbb{R}^k \to \mathbb{R}^d$ "decoding"

$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{f}(\mathbf{x}_i;\mathbf{W})\|_2^2$

Com. Advantage: Parametric model. NN discovers representation.

11 Probabilistic Modeling

11.1 Bayes Optimal Predictor (for Squared Loss)

$R(h) = \iint P(\mathbf{x}, y)\, \ell(y; h(\mathbf{x}))\, d\mathbf{x}\, dy = \mathbb{E}_{\mathbf{x},y}[\ell(y; h(\mathbf{x}))]$

$$\begin{aligned} h^* &= \arg\min_h R(h) = \arg\min_h \mathbb{E}_{\mathbf{x},y}[\ell(y; h(\mathbf{x}))] \\ &= \arg\min_h \mathbb{E}_{\mathbf{x}}[\mathbb{E}_y[\ell(y; h(\mathbf{x})) \mid \mathbf{x}]] \quad \text{optimize indep. for each } \mathbf{x} \\ &= \arg\min_{h(\mathbf{x})} \mathbb{E}_y[\ell(y; h(\mathbf{x})) \mid \mathbf{x}] \end{aligned}$$

$\frac{d}{dh}\mathbb{E}_y[\ell(y; h(\mathbf{x})) \mid \mathbf{x}] = \frac{d}{dh}\int P(y \mid \mathbf{x})\, \ell(y; h(\mathbf{x}))\, dy$

$= \int \frac{d}{dh} P(y \mid \mathbf{x})\, \ell(y; h(\mathbf{x}))\, dy \overset{!}{=} 0 \Longrightarrow h^* = \mathbb{E}_y[Y \mid \mathbf{X} = \mathbf{x}]$.

11.2 Bias Variance Tradeoff

$$\mathbb{E}_D\left[\mathbb{E}_{\mathbf{X},Y}\left[(Y - \hat{h}_D(\mathbf{X}))^2\right]\right] = \mathbb{E}_{\mathbf{X}}\left[\left(\mathbb{E}_D\left[\hat{h}_D(\mathbf{X})\right] - h^*(\mathbf{X})\right)^2\right]$$
$$+ \mathbb{E}_{\mathbf{X}}\left[\mathbb{E}_D\left[\left(\hat{h}_D(\mathbf{X}) - \mathbb{E}_{D'}\left[\hat{h}_{D'}(\mathbf{X})\right]\right)^2\right] + \mathbb{E}_{\mathbf{X},Y}\left[(Y - h^*(\mathbf{X}))^2\right]\right]$$

11.3 Estimating Conditional Distributions

11.3.1 Maximum (Cond.) Likelihood Est., (MLE)

$$\begin{aligned} \boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \hat{P}(y_1, \dots, y_n \mid \mathbf{x}_1, \dots, \mathbf{x}_n, \boldsymbol{\theta}) \\ &\overset{\text{i.i.d}}{=} \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^n \hat{P}(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^n \log \hat{P}(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) \\ &= \arg\min_{\boldsymbol{\theta}} -\sum_{i=1}^n \log \hat{P}(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \dots \text{insert.} \end{aligned}$$

11.3.2 Maximum a Posteriori Estimate, (MAP)

$$\begin{aligned} \boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta} \mid D) = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta} \mid \mathbf{x}_1, \dots, \mathbf{x}_n, y_1, \dots, y_n) \\ &= \arg\max_{\boldsymbol{\theta}} \frac{P(\boldsymbol{\theta}) P(y_1, \dots, y_n \mid \mathbf{x}_1, \dots, \mathbf{x}_n, \boldsymbol{\theta})}{P(y_1, \dots, y_n \mid \mathbf{x}_1, \dots, \mathbf{x}_n)} \\ &= \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta})\, P(y_1, \dots, y_n \mid \mathbf{x}_1, \dots, \mathbf{x}_n, \boldsymbol{\theta}) \\ &= \arg\min_{\boldsymbol{\theta}} -\log P(\boldsymbol{\theta}) - \log P(y_1, \dots, y_n \mid \mathbf{x}_1, \dots, \mathbf{x}_n, \boldsymbol{\theta}) \\ &\overset{\text{i.i.d}}{=} \arg\min_{\boldsymbol{\theta}} -\log P(\boldsymbol{\theta}) - \sum_{i=1}^n \log P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) = \dots \text{insert} \end{aligned}$$

11.4 Introducing Bias trough Bayesian Modeling

- $P(D \mid \boldsymbol{\theta})$ is the *likelihood of the data D given the parameters $\boldsymbol{\theta}$*
- **(Bayesian Prior)** $P(\boldsymbol{\theta})$ is the *prior belief* about $\boldsymbol{\theta}$.
  (Conjugate Prior) if $P(\boldsymbol{\theta} \mid D)$ in same family as $P(\boldsymbol{\theta})$
- $P(\boldsymbol{\theta} \mid D)$ is our *posterior belief*
- The normalization constant $P(D) = \int P(D, \boldsymbol{\theta})\, d\boldsymbol{\theta}$ is called the *marginal likelihood* or *evidence* (model dependent).

12 Decision Theory

Given Cond. distr. $P(y \mid \mathbf{x})$ Set of actions $\mathcal{A}$, cost func. $C : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}$

Goal Bayesian Decision Theory recommends to pick (b. opt. dec.)

$a^* = \arg\min_{a \in \mathcal{A}} \mathbb{E}_y[C(y, a) \mid \mathbf{x}] = \arg\min_{a \in \mathcal{A}} \int_y P(y \mid \mathbf{x})\, C(y, a)\, dy$

12.1 Uncertainty Sampling (Active Learning)

Strategy Always pick the example that we are *most uncertain* about.

$i_t \in \arg\min_i \left|\frac{1}{2} - \hat{P}(Y_i \mid \mathbf{x}_i)\right|$ (where $\mathbf{x}_i \in D_X$)

Com. violates the i.i.d. assumption.

13 Generative Models

13.1 Typical Approach to Generative Modeling

1) Estimate prior on labels $\hat{P}(y)$.
2) For each class y estimate conditional distribution $\hat{P}(\mathbf{x} \mid y)$.
3) Obtain predictive distribution using Bayes' rule:
   $\hat{P}(y \mid \mathbf{x}) = \frac{\hat{P}(y, \mathbf{x})}{\hat{P}(\mathbf{x})} = \frac{\hat{P}(y)\hat{P}(\mathbf{x} \mid y)}{\hat{P}(\mathbf{x})} = \frac{1}{Z}\hat{P}(y)\, \hat{P}(\mathbf{x} \mid y)$, where
   $Z = \hat{P}(\mathbf{x}) = \sum_y \hat{P}(y, \mathbf{x}) = \sum_y \hat{P}(y)\, \hat{P}(\mathbf{x} \mid y)$
4) Predict / decide using Bayesian decision theory with obtained predictive distribution: $a^* = \arg\min_a \mathbb{E}_y[C(y, a) \mid \mathbf{x}]$.

5) Perform outlier detection using $P(\mathbf{x})$ from above. Choose e treshold τ, such that $P(\{\mathbf{x} \mid P(\mathbf{x}) \geq \tau\}) \geq 1 - \delta$ for a small δ.

13.2 Naive Bayes Model (NB)

(i) Model *class label* as generated from a *categorical* variable
   $P(Y = y) = p_y, \quad y \in \mathcal{Y} = \{1, \dots, c\}, \quad p_y \geq 0, \quad \sum_{y \in \mathcal{Y}} p_y = 1$
(ii) Model *features* (for a given class label Y) as *conditionally independent*
   $P(X_1, \dots, X_d \mid Y) = \prod_{i=1}^d P(X_i \mid Y)$

13.2.1 Gaussian Naive Bayes Classifiers (GNBCs)

Here the *features (ii)* are modeled by *(conditionally) independent Gaussians* $P(x_i \mid y) = \mathcal{N}(x_i; \mu_{y,i}, \sigma_{y,i}^2)$

Prediction via Discriminant Function (for $c = 2$)

$$y^* = \arg\max_y P(y \mid \mathbf{x}) = \text{sign}\underbrace{\left(\log\frac{P(Y = +1 \mid \mathbf{x})}{P(Y = -1 \mid \mathbf{x})}\right)}_{f(\mathbf{x})\ (\text{discr. func.})}$$

If we have the discr. func., we can always get back the class probab.:

$f(\mathbf{x}) = \log\frac{P(Y=+1 \mid \mathbf{x})}{1 - P(Y=+1 \mid \mathbf{x})} \iff P(Y = +1 \mid \mathbf{x}) = \frac{1}{1 + e^{-f(\mathbf{x})}} = \sigma(f(\mathbf{x}))$

Special Case: (Equivalence to Log. Reg.)

$c = 2$, class independent variance $P(\mathbf{x} \mid y) = \prod_i^d \mathcal{N}(x_i; \boldsymbol{\mu}_{y,i}, \sigma_i^2)$ (equal diagonal covariance matrices) $\to$ discriminant is linear:

$$f(\mathbf{x}) = \dots \overset{\text{NB}}{=} \sum_{i=1}^d x_i \underbrace{\left(\frac{\mu_{+,i} - \mu_{-,i}}{\sigma_i^2}\right)}_{w_i} + \underbrace{\log\frac{\hat{p}_+}{1 - \hat{p}_+} + \sum_{i=1}^d \frac{\mu_{-,i}^2 + \mu_{+,i}^2}{2\sigma_i^2}}_{w_0} = \mathbf{w}^\intercal\mathbf{x} + w_0$$

So, if the assumption of shared variance is met, GNB = Log. Reg.

$P(Y = +1 \mid \mathbf{x}) = \frac{1}{1 + e^{-f(\mathbf{x})}} = \sigma(\mathbf{w}^\intercal\mathbf{x} + w_0)$

13.2.2 Categorical Naive Bayes Classifiers

Features *(ii)* are modeled by (cond.) independent categorical random variables.

$P(X_i = c \mid Y = y) = \theta_{c|y}^{(i)}, \quad \forall i, y : \sum_c \theta_{c|y}^{(i)}, \quad \theta_{c|y}^{(i)} \geq 0.$

13.3 Bayes Classifiers (BCs)

(i) Model *class label* as generated from *categorical* variable
   $P(Y = y) = p_y, \quad y \in \mathcal{Y} = \{1, \dots, c\}$
(ii) Model *features* (for a given class label Y) trough joint-distribution
   $P(X_1, \dots, X_d \mid Y)$ (features not necessarily cond. indep.)
   Again we may use any distribution here for the joint-distribution.

13.3.1 Gaussian Bayes Classifiers (GBCs)

Here the *features (ii)* are modeled by a *multivariate Gaussian*

$P(\mathbf{x} \mid y) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y), \quad \mathbf{x}, \boldsymbol{\mu}_j \in \mathbb{R}^d, \quad \boldsymbol{\Sigma}_y \in \mathbb{R}^{d \times d}$

MLE for class label distribution $\hat{P}(Y = y) = \hat{p}_y = \frac{\text{Count}(Y=y)}{n}$

MLE for feature distribution $\hat{P}(\mathbf{x} \mid y) = \mathcal{N}(\mathbf{x}; \hat{\boldsymbol{\mu}}_y, \hat{\boldsymbol{\Sigma}}_y)$

$$\hat{\boldsymbol{\mu}}_y = \frac{1}{\text{Count}(Y = y)}\sum_{i\,:\, y_i = y}\mathbf{x}_i, \quad \hat{\boldsymbol{\Sigma}}_y = \frac{1}{\text{Count}(Y = y)}\sum_{i\,:\, y_i = y}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_y)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_y)^\intercal$$

Discriminant Function

$$f(\mathbf{x}) = \log\left(\frac{p_+}{1 - p_+}\right) + \frac{1}{2}\left(\log\left(\frac{\det\hat{\boldsymbol{\Sigma}}_-}{\det\hat{\boldsymbol{\Sigma}}_+}\right) + (\mathbf{x} - \hat{\boldsymbol{\mu}}_-)^\intercal\hat{\boldsymbol{\Sigma}}_-^{-1}(\mathbf{x} - \hat{\boldsymbol{\mu}}_-) - (\mathbf{x} - \hat{\boldsymbol{\mu}}_+)^\intercal\hat{\boldsymbol{\Sigma}}_+^{-1}(\mathbf{x} - \hat{\boldsymbol{\mu}}_+)\right)$$

Special Cases

- **Fishers Linear Discriminant Analysis** $c = 2$, $p_+ = p_- = 0.5$, $\boldsymbol{\Sigma}_- = \boldsymbol{\Sigma}_+$, and let $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ $\to$ discr. func. linear:
  $$f(\mathbf{x}) = \mathbf{x}^\intercal\underbrace{\boldsymbol{\Lambda}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)}_{\mathbf{w}} + \underbrace{\frac{1}{2}\boldsymbol{\mu}_-^\intercal\boldsymbol{\Lambda}\boldsymbol{\mu}_- - \frac{1}{2}\boldsymbol{\mu}_+^\intercal\boldsymbol{\Lambda}\boldsymbol{\mu}_+}_{w_0} = \mathbf{w}^\intercal\mathbf{x} + w_0$$
- **Quadratic Analysis** In general $\boldsymbol{\Sigma}_- \neq \boldsymbol{\Sigma}_+ \to f$ quadratic (as above)

14 Latent Variable Modeling

Clustering = Latent Variable Modeling (w. all features + no labels)

14.1 Mixture Modeling

The data is approximated through various clusters. We model each cluster j as a weighted probability distribution $w_j P(\mathbf{x} \mid \boldsymbol{\theta}_j)$. Ass iid $\to$ likh. of. data: $P(D \mid \boldsymbol{\theta}) = \prod_{i=1}^n \sum_{j=1}^k w_j P(\mathbf{x}_i \mid \theta_j)$ where $w_j \geq 0$ and $\sum_{j=1}^k w_j = 1$.

14.1.1 Gaussian Mixture Models (GMMs)

Gaussian Mixtures are a *convex-combination* of *Gaussian distributions*

$\boldsymbol{\theta} = [(w_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), \dots, (w_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$, $w_i \geq 0$, $\sum w_i = 1$

$P(Z = z \mid \boldsymbol{\theta}) = w_z, \quad P(X = \mathbf{x} \mid Z = z, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$

$P(\mathbf{x} \mid \boldsymbol{\theta}) = \sum_{z=1}^k P(\mathbf{x}, z \mid \boldsymbol{\theta}) = \sum_{z=1}^k P(\mathbf{z} \mid \boldsymbol{\theta})\, P(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\theta}) = \sum_{z=1}^k w_z \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$

MLE Estimate (non-convex, hard to solve via gradient desc.)

Basic trick guess z, compute MLE in closed form!

Hard-EM

E-step Predict most likely class for each point $\mathbf{x}_i$

$z_i^{(t)} = \arg\max_z P\left(z \,\middle|\, \mathbf{x}_i, \boldsymbol{\theta}^{(t-1)}\right) = \arg\max_z w_z^{(t-1)}\mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_z^{(t-1)}, \boldsymbol{\Sigma}_z^{(t-1)})$

M-step Given $z_i^{(t)}$s compute MLE of $\boldsymbol{\theta}$ (as in GBC).

Special Case (k**-Means**): Uniform weights $w_1 = \dots = w_k = \frac{1}{k}$ and identical spherical cov. mat. $\boldsymbol{\Sigma}_1 = \dots = \boldsymbol{\Sigma}_k = \sigma^2\mathbf{I}$ for a fixed σ^2. Then the E/M-step are the same as in k-Means.

Soft-EM

E-step: $\gamma_j^{(t)}(\mathbf{x}_i) \leftarrow P(Z = j \mid \mathbf{x}_i, \boldsymbol{\theta}) = \frac{w_j^{(t-1)}\mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_j^{(t-1)}, \boldsymbol{\Sigma}_j^{(t-1)})}{\sum_{\ell=1}^k w_\ell^{(t-1)}\mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_\ell^{(t-1)}, \boldsymbol{\Sigma}_\ell^{(t-1)})}$

M-step: (MLE, MAP, given $\gamma_j(\mathbf{x}_i)$'s)

$$w_j^{(t)} \leftarrow \frac{1}{n}\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i), \quad \boldsymbol{\mu}_j^{(t)} \leftarrow \frac{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i)}, \quad \boldsymbol{\Sigma}_j^{(t)} \leftarrow \frac{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i)(\mathbf{x}_i - \boldsymbol{\mu}_j^{(t)})(\mathbf{x}_i - \boldsymbol{\mu}_j^{(t)})^\intercal}{\sum_{i=1}^n \gamma_j^{(t)}(\mathbf{x}_i)}$$

Special Case (k**-Means**): Uniform weights $w_1 = \dots = w_k = \frac{1}{k}$ and identical spherical cov. mat. $\boldsymbol{\Sigma}_1 = \dots = \boldsymbol{\Sigma}_k = \sigma^2\mathbf{I}$ with $\sigma \to 0$ (since post. prob. $\gamma_j(\mathbf{x}_i)$ become deterministic, converge to 0 or 1).

14.1.2 Selecting k for GMMs

Elbow method, or CV works (degeneracy of GMMs), to avoid degeneracy: regularization $\boldsymbol{\Sigma}_j^{(t)} \leftarrow \dots + \nu^2\mathbf{I}$ (wishart prior).

14.2 MMs for Outlier Detection

Use $P(\mathbf{x})$ and if there are examples vary treshold τ, $P(\mathbf{x}) < \tau$.

14.3 MMs in Conjunction with Discr. Models

Use $P(\mathbf{x})$ from GMM (density esimation) and use $P(y \mid \mathbf{x})$ from robust discr. model (prediction). And then: $P(\mathbf{x}, y) = P(\mathbf{x})\, P(y \mid \mathbf{x})$.

14.4 Semi-Superv. Learning with GMMs

$$\gamma_j(\mathbf{x}_i) = \begin{cases} \mathbb{1}_{\{j = y_i\}} & \text{if } \mathbf{x}_i \text{ is labeled} \\ P(Z = j \mid \mathbf{x}_i, \boldsymbol{\Sigma}, \boldsymbol{\mu}, \mathbf{w}) & \text{if } \mathbf{x}_i \text{ is unlabeled} \end{cases}$$

15 Time Series

Given A sequence of observations $y_1, \dots, y_t$ (typically discrete, unit-length time steps, not i.i.d - dependent over time)

Goal Predict y_{t+1}

15.1 Markov Chains

Markov Assumption: (next state only depends on the prev. state)

$\forall t \geq 1 : P(Y_t \mid Y_1, \dots, Y_{t-1}) = P(Y_t \mid Y_{t-1})$

Stationarity Assumption: (trans. prob. remain const. over time)

$\forall t, y, y' : P\left(Y_{t+1} = y \,\middle|\, Y_t = y'\right) = P\left(Y_t = y \,\middle|\, Y_{t-1} = y'\right)$

15.1.1 Prediction

Sum Rule $\to$ Prod. Rule $\to$ Markov Assump. $\to$ Stat. Assump.

$P(Y_{t+\ell} = y \mid y_{1:t}) = \dots = \sum_{y_{t+(\ell-1)}} \cdots \sum_{y_{t+1}} \theta_{y|y_{t+(\ell-1)}} \cdots \theta_{y_{t+1}|y_t}$

Matrix/Vector Notation:

Represent $P(Y_t = y) = p_y^{(t)}$, where $y \in \{1, \dots, c\}$ as a vector

$\mathbf{p}^{(t)} = (p_1^{(t)}, p_2^{(t)}, \dots, p_c^{(t)})^\intercal \in [0, 1]^c$

Represent $P\left(Y_{t+1} = y \,\middle|\, Y_t = y'\right)$ as matrix $\mathbf{T} \in [0, 1]^{c \times c}$ *(transition matrix)*

$T_{y,y'} = P\left(Y_{t+1} = y \,\middle|\, Y_t = y'\right) = \theta_{y|y'}$

Then it holds that $\mathbf{p}^{(t+\ell)} = \mathbf{T}^\ell\mathbf{p}^{(t)}$.

15.1.2 Reduction: k-th Order to 1st Order

Enrich state space. Memory and running time $\mathcal{O}(c^{k+1})$.

15.1.3 Learning a Markov Chain

$\hat{p}_y = \frac{\text{Count}(Y_1 = y)}{m}, \quad \hat{\theta}_{y|y'} = \frac{\text{Count}(Y_{t+1} = y, Y_t = y')}{\text{Count}(Y_t = y')}$

Com. We may also do MAP by adding pseudo-counts.

15.2 Gaussian Linear Time Series

For example, we could approximate $P(Y_{t+1} \mid y_{1:t})$ trough

$P(Y_{t+1} \mid y_{t-k+1}, \dots, y_t) = \mathcal{N}\left(y; \mathbf{w}_0 + \sum_{i=1}^k w_i y_{t-k+i}, \sigma^2\right)$

This is called a *(Gaussian) autoregressive model of order k*.

Key idea: Don't allow arbitrary dependence on previous k values.

Key idea: Y_t are dep., BUT: *transitions* are indep.

15.3 Gaussian Non-Linear Time Series

$P(Y_{t+1} = y \mid y_{t-k+1}, \dots, y_t) = \mathcal{N}\left(y; f(y_{t-k+1}, \dots, y_t; \boldsymbol{\theta}), \sigma^2\right)$

for some (nonlinear, multivariate) function f (e.g., trained NN).

15.4 Bernoulli Non-Linear Time Series

$P(Y_{t+1} = +1 \mid y_{t-k+1}, \dots, y_t) = \frac{1}{1 + e^{-f(y_{t-k+1}, \dots, y_t; \boldsymbol{\theta})}}$

15.5 Predicting Multiple Timesteps ahead

Use forward-sampling algorithm. Do prediction, sample, use prediction, return average to approximate expected value.