Regularization

- L0 - Count of the non-zero weights – Difficult minimize
- L1 Lasso - Sum of the attribute weights
  - It set the weights to 0 for irrelevant attributes

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

- L2 Ridge (Squared Euclidean norm) - Sum of the squared attribute weights. Used to avoid Overfiiting.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Scaling:

- Min/Max normalization: $x^{new} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}(x_{\max}^{new} - x_{\min}^{new}) + x_{\min}^{new}$
- Z-Score normalisierung: $x^{new} = \frac{x - \mu_x}{\sigma_x}$
- Decimal scaling: $x^{new} = |x| \cdot 10^a \quad a = \max_x\{i \in \mathbb{Z} \mid |x| \cdot 10^i < 1\}$
- Logarithmic scaling: $x^{new} = \log_a x$

Information Gain of an Attribute

- Reduction of entropy by splitting the data along an attribute
  - $G_L(x_j) = H_L(y) - \sum_{v=1}^{k} p_L(x_j = v) H_L(y|x_j = v)$

Loan	x_1 (Credit report)	x_2 (Employment last 3 months)	x_3 (Collateral > 50% loan)	y (Payed back in full)
1	Positive	Yes	No	Yes
2	Positive	No	Yes	Yes
3	Positive	No	No	No
4	Negative	No	Yes	No
5	Negative	Yes	No	No

- $G_L(x_1) = H_L(y) - p_L(x_1 = p)H_L(y|x_1 = p) - p_L(x_1 = n)H_L(y|x_1 = n)$
  $= 0.97 - \frac{3}{5}0.91 - \frac{2}{5}0 = 0.42bit$
- Splitting along x_1 reduces the uncertainty regarding the class label y by 0.42 bit.

Information Gain

Entropy is the expected information of a message.

- $H(y) = -\sum_{v=1}^{k} p(y = v)\log_2 p(y = v)$

Loan	x_1 (Credit report)	x_2 (Employment last 3 months)	x_3 (Collateral > 50% loan)	y (Payed back in full)
1	Positive	Yes	No	Yes
2	Positive	No	Yes	Yes
3	Positive	No	No	No
4	Negative	No	Yes	No
5	Negative	Yes	No	No

$$H_L(y|x_1 = n) = -\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{2}\log_2\frac{0}{2} = 0bit$$

Idea: factor the information that is contained in the attribute values into the decision

- $H_L(x_j) = -\sum_{v=1}^{k} p_L(x_j = v)\log_2 p_L(x_j = v)$

Information gain ratio:

- $GR_L(x_j) = \frac{G_L(x_j)}{H_L(x_j)}$

ID3

```
ID3(L, X)
1. If all data in L have same class y or X={}, then
   return leaf node with majority class y.
2. Else
   1. For all attributes xj ∈ X, calculate split
      criterion GL(xj) or GRL(xj).
   2. Choose attribute xj ∈ X with highest GL(xj) or
      GRL(xj).
   3. Let Li = {(x,y) ∈ L: xj = i}.
   4. Return test node with attribute xj and children
      ID3(L1, X\xj), …, ID3(Lk, X\xj).
```

C4.5

Learning Decision Trees with C4.5

```
C4.5(L)
1. If all data in L have same class y or are identical, then
   return leaf node with majority class y.
2. Else
   1. For all discrete attributes xj ∈ X: calculate GL(xj).
   2. For all continuous attributes xj ∈ X and all values v
      that occur for xj in L: calculate GL(xj ≤v).
   3. If discrete attribute has highest GL(xj):
      1. Let Li = {(x,y) ∈ L: xj = i}.
      2. Return test node with attribute xj and children
         C4.5(L1), …, C4.5(Lk).
   4. If continuous attribute has highest GL(xj ≤v):
      1. Let L≤ = {(x,y) ∈ L: xj ≤ v}, L> = {(x,y) ∈ L: xj>v}
      2. Return test node with test xj ≤v and children
         C4.5(L≤), C4.5(L>).
```

Regression Trees

- Variance of the target attribute on sample L:
  - $Var(L) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$
- Variance = MSE of predicting the mean value.
- Splitting criterion: variance reduction of $[x_j \le v]$:
  - $R_L[x_j \le v]$
    $= Var(L) - \frac{n_{[x_j \le v]}}{n} Var\left(L_{[x_j \le v]}\right) - \frac{n_{[x_j > v]}}{n} Var\left(L_{[x_j > v]}\right)$
- Stopping criterion:
  - Do not create a new test node if $nVar(L) \le \tau$.

CART(L)

1. If $\sum_{i=1}^{n}(y_i - \bar{y})^2 < \tau$, then return leaf node with prediction $\bar{y}$.
2. Else
   1. For all discrete attributes $x_j \in X$: calculate $R_L(x_j)$.
   2. For all continuous attributes $x_j \in X$ and all values v that occur for x_j in L: calculate $R_L(x_j \le v)$.
   3. If discrete attribute has highest $R_L(x_j)$:
      1. Let $L_i = \{(x,y) \in L: x_j = i\}$.
      2. Return test node with attribute x_j and children CART(L_1), …, CART(L_k).
   4. If continuous attribute has highest $R_L(x_j \le v)$:
      1. Let $L_\le = \{(x,y) \in L: x_j \le v\}$, $L_> = \{(x,y) \in L: x_j > v\}$
      2. Return test node with test $x_j \le v$ and children CART($L_\le$), CART($L_>$).

Random Forest

- Input: sample L of size n, attributes X.

1. For i=1…k
   1. Draw n instances uniformly with replacement from L into set L_i.
   2. Draw m attributes from all attributes X into X_i.
   3. Learn model f_i on sample L_i using attributes X_i.
2. For classification:
   1. Let $f(\mathbf{x})$ be the majority vote among $(f_1(\mathbf{x}), \dots, f_k(\mathbf{x}))$.
3. For regression:
   1. Let $f(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} f_i(\mathbf{x})$.

Loss in Classification

- Zero-one loss:

$$\ell_{0/1}(f_\theta(\mathbf{x}_i), y_i) = \begin{cases} \overbrace{1 \quad -y_i f_\theta(\mathbf{x}_i) > 0}^{\mathrm{sign}(f_\theta(\mathbf{x}_i)) \ne y_i} \\ \underbrace{0 \quad -y_i f_\theta(\mathbf{x}_i) \le 0}_{\mathrm{sign}(f_\theta(\mathbf{x}_i)) = y_i} \end{cases}$$

- Logistic loss:

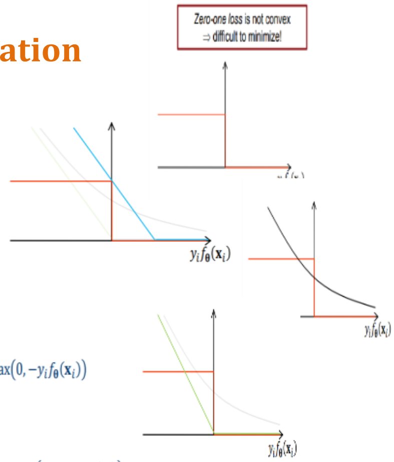
$$\ell_{log}(f_\theta(\mathbf{x}_i), y_i) = \log\left(1 + e^{-y_i f_\theta(\mathbf{x}_i)}\right)$$

- Perceptron loss:

$$\ell_p(f_\theta(\mathbf{x}_i), y_i) = \begin{cases} -y_i f_\theta(\mathbf{x}_i) & -y_i f_\theta(\mathbf{x}_i) > 0 \\ 0 & -y_i f_\theta(\mathbf{x}_i) \le 0 \end{cases} = \max(0, -y_i f_\theta(\mathbf{x}_i))$$

- Hinge loss:

$$\ell_h(f_\theta(\mathbf{x}_i), y_i) = \begin{cases} 1 - y_i f_\theta(\mathbf{x}_i) & 1 - y_i f_\theta(\mathbf{x}_i) > 0 \\ 0 & 1 - y_i f_\theta(\mathbf{x}_i) \le 0 \end{cases} = \max(0, 1 - y_i f_\theta(\mathbf{x}_i))$$



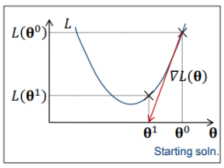
Gradient descent

Linear classification model: minimize

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell(\mathbf{x}^T\boldsymbol{\theta}, y_i) + \lambda\,\Omega(\boldsymbol{\theta})$$

Gradient descent method:

```
RegERM(Data: (x_1,y_1),...,(x_n,y_n))
  Set θ^0 = 0 and t = 0
  DO
      Compute gradient ∇L(θ^t)
      Compute step size α^t
      Set θ^{t+1} = θ^t − α^t ∇L(θ^t)
      Set t = t+1
  WHILE ||θ^t − θ^{t+1}|| > ε
RETURN θ^t
```



with Step size (line search)

Determine step size through line search:

```
RegERM-LineSearch(Data: (x_1,y_1),...,(x_n,y_n))
  Set θ^0 = 0 and t = 0
  DO
      Compute gradient ∇L(θ^t)
      Choose step size α^t:
          α^t = argmin_{α>0} L(θ^t − α∇L(θ^t))
      Set θ^{t+1} = θ^t − α^t ∇L(θ^t)
      Set t = t+1
  WHILE ||θ^t − θ^{t+1}|| > ε
RETURN θ^t
```

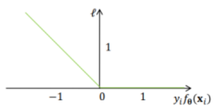
Stochastic Gradient descent

```
RegERM-Stoch(Data: (x_1,y_1),...,(x_n,y_n))
  Set θ^0 = 0 and t = 0
  DO
      Shuffle data randomly
      FOR i = 1,...,n
          Compute subset gradient ∇_{x_i} L(θ^t)
          Compute step size α^t
          Set θ^{t+1} = θ^t − α^t ∇_{x_i} L(θ^t)
          Set t = t+1
      END
  WHILE ||θ^t − θ^{t+1}|| > ε
RETURN θ^t
```

Perceptron

Loss function:

$$\ell_p(f_\theta(\mathbf{x}_i), y_i) = \begin{cases} -y_i f_\theta(\mathbf{x}_i) & -y_i f_\theta(\mathbf{x}_i) > 0 \\ 0 & -y_i f_\theta(\mathbf{x}_i) \le 0 \end{cases} = \max(0, -y_i f_\theta(\mathbf{x}_i))$$



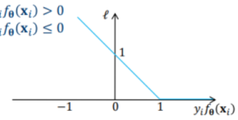

```
Perceptron(Instances {(x_i, y_i)})
  Set θ = 0
  DO
    FOR i = 1,...,n
      IF     y_i f_θ(x_i) ≤ 0
      THEN   θ = θ + y_i x_i
    END
  WHILE θ changes
  RETURN θ
```

Stochastic gradient method:

- $\nabla L_{\mathbf{x}_i}(\boldsymbol{\theta}) = \begin{cases} -y_i \mathbf{x}_i & -y_i f_\theta(\mathbf{x}_i) > 0 \\ 0 & -y_i f_\theta(\mathbf{x}_i) < 0 \end{cases}$

Loss function:

- $\ell_h(f_{\boldsymbol\theta}(\mathbf{x}_i), y_i) = \begin{cases} 1 - y_i f_{\boldsymbol\theta}(\mathbf{x}_i) & \text{if } 1 - y_i f_{\boldsymbol\theta}(\mathbf{x}_i) > 0 \\ 0 & \text{if } 1 - y_i f_{\boldsymbol\theta}(\mathbf{x}_i) \le 0 \end{cases}$
  $= \max(0, 1 - y_i f_{\boldsymbol\theta}(\mathbf{x}_i))$



Regularizer:

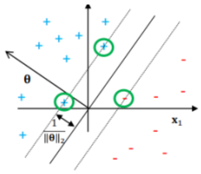
- $\Omega_2(\boldsymbol\theta) = \boldsymbol\theta^{\mathrm{T}}\boldsymbol\theta = \sum_{j=1}^{m} |\theta_j|^2 = \|\boldsymbol\theta\|_2^2$

SVM

Loss function is 0, if…

$$\sum_{i=1}^{n} \max(0, 1 - y_i f_{\boldsymbol\theta}(\mathbf{x}_i)) = 0$$
$$\Leftrightarrow \forall_{i=1}^{n}: y_i f_{\boldsymbol\theta}(\mathbf{x}_i) \ge 1$$
$$\Leftrightarrow \forall_{i=1}^{n}: y_i \mathbf{x}_i{}^{\mathrm{T}}\boldsymbol\theta \ge 1$$
$$\Leftrightarrow \forall_{i=1}^{n}: y_i \mathbf{x}_i{}^{\mathrm{T}} \frac{\boldsymbol\theta}{\|\boldsymbol\theta\|_2} \ge \frac{1}{\|\boldsymbol\theta\|_2}$$
$$\Leftrightarrow \forall_{i=1}^{n}: \mathbf{x}_i{}^{\mathrm{T}} \frac{\boldsymbol\theta}{\|\boldsymbol\theta\|_2} \begin{cases} \ge \frac{1}{\|\boldsymbol\theta\|_2} & \text{if } y_i = +1 \\ \le \frac{-1}{\|\boldsymbol\theta\|_2} & \text{if } y_i = -1 \end{cases}$$

Distance of closest point to plane (margin)



- Linear classification model: minimize

$$L(\boldsymbol\theta) = \sum_{i=1}^{n} \left[\max(0, 1 - y_i \mathbf{x}_i{}^{\mathrm{T}}\boldsymbol\theta) + \frac{\lambda}{n} \boldsymbol\theta^{\mathrm{T}}\boldsymbol\theta \right]$$

- Gradient:

$$\nabla L(\boldsymbol\theta) = \sum_{i=1}^{n} \nabla_{\mathbf{x}_i} L(\boldsymbol\theta)$$

- Stochastic gradient for $\mathbf{x}_i$:

$$\nabla_{\mathbf{x}_i} L(\boldsymbol\theta) = \begin{cases} \frac{2\lambda}{n}\boldsymbol\theta & \text{if } y_i \mathbf{x}_i{}^{\mathrm{T}}\boldsymbol\theta > 1 \\ \frac{2\lambda}{n}\boldsymbol\theta - y_i \mathbf{x}_i & \text{if } y_i \mathbf{x}_i{}^{\mathrm{T}}\boldsymbol\theta < 1 \end{cases}$$

Linear Regression

Regularizer for Regression

- L1 regularization:

$$\Omega_1(\boldsymbol\theta) \propto \|\boldsymbol\theta\|_1 = \sum_{j=1}^{m} |\theta_j|$$

- L2 regularization:

$$\Omega_2(\boldsymbol\theta) \propto \|\boldsymbol\theta\|_2^2 = \sum_{j=1}^{m} \theta_j{}^2$$

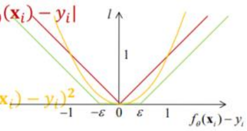
Loss Functions for Regression

- Absolute loss:

$$\ell_{\text{abs}}(f_{\boldsymbol\theta}(\mathbf{x}_i), y_i) = |f_{\boldsymbol\theta}(\mathbf{x}_i) - y_i|$$

- Squared loss:

$$\ell_2(f_{\boldsymbol\theta}(\mathbf{x}_i), y_i) = (f_{\boldsymbol\theta}(\mathbf{x}_i) - y_i)^2$$



- ε-insensitive loss:

$$\ell_\varepsilon(f_{\boldsymbol\theta}(\mathbf{x}_i), y_i) = \begin{cases} |f_{\boldsymbol\theta}(\mathbf{x}_i) - y_i| - \varepsilon & |f_{\boldsymbol\theta}(\mathbf{x}_i) - y_i| - \varepsilon > 0 \\ 0 & |f_{\boldsymbol\theta}(\mathbf{x}_i) - y_i| - \varepsilon \le 0 \end{cases}$$

Special Cases

- Lasso: squared loss + L1 regularization

$$L(\boldsymbol\theta) = \sum_{i=1}^{n} \ell_2(f_{\boldsymbol\theta}(\mathbf{x}_i), y_i) + \lambda\|\boldsymbol\theta\|_1$$

- Ridge regression: squared loss + L2 regularization

$$L(\boldsymbol\theta) = \sum_{i=1}^{n} \ell_2(f_{\boldsymbol\theta}(\mathbf{x}_i), y_i) + \lambda\|\boldsymbol\theta\|_2^2$$

- Elastic net: squared loss, L1 + L2 regularization

$$L(\boldsymbol\theta) = \sum_{i=1}^{n} \ell_2(f_{\boldsymbol\theta}(\mathbf{x}_i), y_i) + \lambda\|\boldsymbol\theta\|_2^2 + \lambda'\|\boldsymbol\theta\|_1$$

Ridge

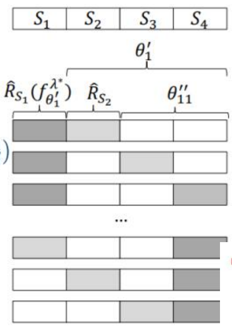
$$\boldsymbol\theta = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

Lasso

$$L(\boldsymbol\theta) = (\mathbf{X}\boldsymbol\theta - \mathbf{y})^{\mathrm{T}}(\mathbf{X}\boldsymbol\theta - \mathbf{y}) + \lambda\|\boldsymbol\theta\|_1$$

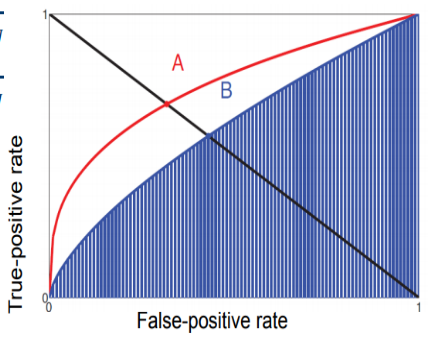
Nested Cross Validation

- For $i = 1 \dots k$
  - Iterate over values λ
    - For $j = 1 \dots k \setminus i$
      - Train $f_{\theta_{ij}}^{\lambda}$ on $S \setminus S_i \setminus S_j$
      - Determine $\hat{R}_{S_j}\left(f_{\theta_{ij}}^{\lambda}\right)$
    - Average $\hat{R}_{S_j}$ to determine $\hat{R}_{S\setminus S_i}\left(f_{\theta_i'}^{\lambda}\right)$
  - Choose λ_i^* that minimizes $\hat{R}_{S\setminus S_i}\left(f_{\theta_i'}^{\lambda}\right)$
  - Train $f_{\theta_i}^{\lambda_i^*}$ on $S \setminus S_i$
  - Determine $\hat{R}_{S_i}\left(f_{\theta_i'}^{\lambda_i^*}\right)$
- Average $\hat{R}_{S_i}\left(f_{\theta_i}^{\lambda_i^*}\right)$ to determine $\hat{R}_S(f_{\theta^*}^{\lambda^*})$
- Determine λ^* by averaging λ_i^*
- Train $f_\theta^{\lambda^*}$ on S
- Return $f_\theta^{\lambda^*}$ and $\hat{R}_S(f_{\theta^*}^{\lambda^*})$



$$r_{TP} = \frac{n_{TP}}{n_{TP} + n_{FN}}$$

$$r_{FP} = \frac{n_{FP}}{n_{FP} + n_{TN}}$$



Precision & Recall

- **TP (True Positive)** - Predicted & Actuals are versicolor
- **TN (True Negative)** - Predicted & Actuals are *not versicolor*
- **FP (False Positive)** - Predicted is verisicolor but acutal is not versicolor
- **FN (False Negative)** - Predicted is not verisicolor but acutal is versicolor

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Precision = TP / (TP + FP)

Predicted & Actuals are versicolor / (Predicted & Actuals are versicolor) + (Predicted is verisicolor but acutal is not versicolor)

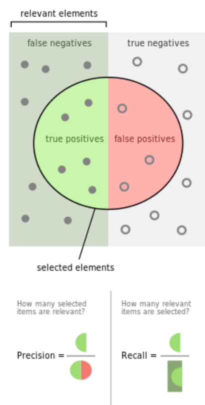
Recall = TP / (TP + FN)

Predicted & Actuals are versicolor / (Predicted & Actuals are versicolor) + (Predicted is not verisicolor but acutal is versicolor)

- F_α measures combine precision and recall values into single value:

$$F_\alpha = \frac{n_{TP}}{\alpha(n_{TP} + n_{FP}) + (1 - \alpha)(n_{TP} + n_{FN})}$$

- $\alpha = 1$: Precision
- $\alpha = 0$: Recall
- $\alpha = 0.5$: "F-measure", harmonic mean of precision and recall.
- Alternative definition: F_β measures.
  - Relationship: $\alpha = \frac{1}{1+\beta}$



Area under the ROC curve (AUC):

- Let $\mathbf{x}_+$ be a randomly drawn positive instance.
- Let $\mathbf{x}_-$ be a randomly drawn negative instance.
- $AUC(\theta) = P(f_\theta(\mathbf{x}_+) > f_\theta(\mathbf{x}_-))$.

Neural Network

Softmax Activation

One output unit per class:

- $x_k^d = \sigma_{sm}(h_k^d) = \frac{e^{h_k^d}}{\sum_{k'} e^{h_{k'}^d}}$
- x_k^d: predicted probability for class k.

Softmax activation function:

- $x_k^d = \sigma_{sm}(h_k^d) = \frac{e^{h_k^d}}{\sum_{k'} e^{h_{k'}^d}}$
- $\frac{\partial \sigma_{sm}(h_k^d)}{\partial h_k^d} = \sigma_{sm}(h_k^d)(1 - \sigma_{sm}(h_k^d))$

Cost function:

- $\ell(\mathbf{y}, \mathbf{x}^d) = \sum_k y_k \log x_k^d$
- $\frac{\partial \ell(\mathbf{y}, \mathbf{x}^d)}{\partial h_k^d} = x_k^d - y_k$

Linear Activation

Linear:

- $x^d = h^d$.
- Output unbounded.

Linear activation function:

- $x_k^d = \sigma_s(h_k^d) = h_k^d$
- $\frac{\partial \sigma_s(h_k^d)}{\partial h_k^d} = 1$

Cost function:

- $\ell(\mathbf{y}, \mathbf{x}^d) = \frac{1}{2}\sum_k (x_k^d - y_k)^2$
- $\frac{\partial \ell(\mathbf{y}, \mathbf{x}^d)}{\partial x_k^d} = x_k^d - y_k$

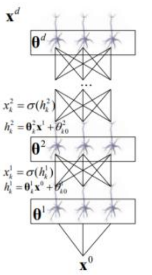
Rectified Linear Units

- $x_k^i = \sigma_{ReLU}(h_k^i) = \max(0, h_k^i)$

Rectified linear activation function:

- $x_k^i = \sigma_{ReLU}(h_k^i) = \max(0, h_k^i)$
- $\frac{\partial \sigma_{ReLU}(h_k^i)}{\partial h_k^i} = \begin{cases} 1 & \text{if } h_k^i > 0 \\ 0 & \text{otherwise} \end{cases}$

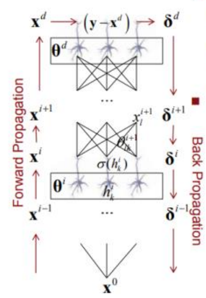
Back Propagation



- Loss function $\hat{R}(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{j=1}^m \ell(\mathbf{y}_j, \mathbf{x}^d)$
- Gradient descent:
  - $\boldsymbol{\theta}' = \boldsymbol{\theta} - \alpha \nabla \hat{R}(\boldsymbol{\theta}) = \boldsymbol{\theta} - \alpha \frac{\partial}{\partial \boldsymbol{\theta}} \hat{R}(\boldsymbol{\theta})$
    $= \boldsymbol{\theta} - \frac{\alpha}{2m} \sum_{j=1}^{m} \frac{\partial}{\partial \boldsymbol{\theta}} \ell(\mathbf{y}_j, \mathbf{x}^d)$
- Stochastic gradient for instance $\mathbf{x}_j$:
  - $\boldsymbol{\theta}' = \boldsymbol{\theta} - \alpha \nabla_{\mathbf{x}_j} \hat{R}(\boldsymbol{\theta})$
    $= \boldsymbol{\theta} - \alpha \frac{\partial}{\partial \boldsymbol{\theta}} \ell(\mathbf{y}_j, \mathbf{x}^d)$

20

- Stochastic gradient for output units for instance $\mathbf{x}_j$:
  $\frac{\partial \ell(\mathbf{y}_j, \mathbf{x}^d)}{\partial \boldsymbol{\theta}_k^d} = \frac{\partial \ell(\mathbf{y}_j, \mathbf{x}^d)}{\partial \mathbf{x}^d} \frac{\partial \mathbf{x}^d}{\partial h_k^d} \frac{\partial h_k^d}{\partial \boldsymbol{\theta}_k^d}$
  $= \frac{\partial \ell(\mathbf{y}_j, \mathbf{x}^d)}{\partial \mathbf{x}^d} \frac{\partial \sigma(h_k^d)}{\partial h_k^d} \mathbf{x}^{d-1} = \delta_k^d \mathbf{x}^{d-1}$
- With
  $\delta_k^d = \frac{\partial \ell(\mathbf{y}_j, \mathbf{x}^d)}{\partial \mathbf{x}^d} \frac{\partial \sigma(h_k^d)}{\partial h_k^d}$



- Stochastic gradient for hidden units for instance $\mathbf{x}_j$:
  $\frac{\partial \ell(\mathbf{y}_j, \mathbf{x}^d)}{\partial \boldsymbol{\theta}_k^i} = \frac{\partial \ell(\mathbf{y}_j, \mathbf{x}^d)}{\partial h_k^i} \frac{\partial h_k^i}{\partial \boldsymbol{\theta}_k^i} = \delta_k^i \mathbf{x}^{i-1}$
- With
  $\delta_k^i = \frac{\partial \ell(\mathbf{y}_j, \mathbf{x}^d)}{\partial h_k^i}$
  $= \frac{\partial \ell(\mathbf{y}_j, \mathbf{x}^d)}{\partial (\mathbf{x}_1^{i+1}, \ldots, \mathbf{x}_{n_{i+1}}^{i+1})} \frac{\partial (\mathbf{x}_1^{i+1}, \ldots, \mathbf{x}_{n_{i+1}}^{i+1})}{\partial h_k^i}$
  $= \sum_{l=1}^{n_i+1} \frac{\partial \ell(\mathbf{y}_j, \mathbf{x}^d)}{\partial h_l^{i+1}} \frac{\partial h_l^{i+1}}{\partial x_k^i} \frac{\partial x_k^i}{\partial h_k^i}$
  $= \sum_{l=1}^{n_i+1} \delta_l^{i+1} \theta_{lk}^{i+1} \frac{\partial \sigma(h_k^i)}{\partial h_k^i}$

Back Propagation: Algorithm

- Iterate over training instances $(\mathbf{x}, \mathbf{y})$:
  - Forward propagation: for i=0…d:
    - For k=1…n_i: $h_k^i = \boldsymbol{\theta}_k^i \mathbf{x}^{i-1} + \theta_{k0}^i$
    - $\mathbf{x}^i = \sigma(\mathbf{h}^i)$
  - Back propagation:
    - For k=1…n_i: $\delta_k^d = \frac{\partial}{\partial h_k^d} \sigma(h_k^d) \frac{\partial}{\partial x_k^d} \ell(y_k, x_k^d)$
      $\boldsymbol{\theta}_k^{d\,\prime} = \boldsymbol{\theta}_k^d - \alpha \delta_k^d \mathbf{x}^{d-1}$
    - For i=d-1…1:
      - For k=1…n_i: $\delta_k^i = \sigma'(h_k^i) \sum_l \delta_l^{i+1} \theta_{lk}^{i+1}$
        $\boldsymbol{\theta}_k^{i\,\prime} = \boldsymbol{\theta}_k^i - \alpha \delta_k^i \mathbf{x}^{i-1}$
- Until concergence

Normal initialization with:

- Draw from $N\left[0, \sqrt{\frac{2}{n_{i+1}+n_i}}\right]$.

n *is number of layers*

Uniform initialization (Glorot initialization):

- Draw from $U\left[-\frac{6}{n_{i+1}+n_i}, \frac{6}{n_{i+1}+n_i}\right]$.

Parallel Inference – weight calculation

$$\mathbf{h}^i = \boldsymbol{\theta}^i \mathbf{x}^{i-1}$$

$$\begin{bmatrix} h_1^i \\ \vdots \\ h_{n_i}^i \end{bmatrix} = \begin{bmatrix} \theta_{11}^i & \ldots & \theta_{1n_{i-1}}^i \\ \vdots & & \\ \theta_{n_i 1}^i & \ldots & \theta_{n_i n_{i-1}}^i \end{bmatrix} \begin{bmatrix} x_1^{i-1} \\ \vdots \\ x_{n_{i-1}}^{i-1} \end{bmatrix}$$

Keep $[x_1^{i-1}, \ldots, x_{n_{i-1}}^{i-1}]$ in cache.

For all rows $j = 1..n_i$ (in parallel):

- Load $[\theta_{j1}^i, \ldots, \theta_{jn_{i-1}}^i]$ into cache.
- For all $k = 1..n_{i-1}$ (in parallel): multiply and sum $\theta_{jk}^i x_k^{i-1}$.

CNN

Convolutional Layers

- Convolution, $k \times k \times d$, stride >1.
  - Input size: $x \times y \times d'$, stride s.
  - output size: $\frac{(x-k+1)}{s} \times \frac{(y-k+1)}{s} \times d$.
- Convolutional layer has
  - $k \times k \times d'$ parameters.
- Decreases the spatial resolution.

Primal, Dual, Duality

- Primal decision function:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^{\mathrm{T}}\phi(\mathbf{x})$$

- Dual decision function:

$$f_{\boldsymbol{\alpha}}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)^{\mathrm{T}}\phi(\mathbf{x}) = \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Phi}\phi(\mathbf{x})$$

- Duality between parameters:

$$\boldsymbol{\theta} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i) = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$$

Bayesian Model

Bayesian Model of Learning

System parameter θ^* ← Nature randomly determines θ^* according to $P(\theta)$

↓

Observations $\mathbf{y}$ for input $\mathbf{X}$

Likelihood of observing $\mathbf{y}|\mathbf{X}$ when model parameter is θ.

A priori ("prior") probability of nature choosing θ

- Bayes' equation:

$$P(\theta|\mathbf{X},\mathbf{y}) = \frac{P(\mathbf{y}|\mathbf{X},\theta)P(\theta)}{P(\mathbf{y}|\mathbf{X})}$$

Probability of observing $\mathbf{y}|\mathbf{X}$; independent of θ.

A posteriori ("posterior") probability that θ is the correct parameter given observations $\mathbf{y}|\mathbf{X}$.

11

Time series:

Time Series: DTW Kernel

- Efficient calculation using dynamic programming
- Let $\gamma(k,l)$ be the minimum squared distance of corresponding points up to time k and l.
- Recursive update:

$$\gamma(k,l) = (\mathbf{x}_k - \mathbf{x}_l)^2 + \min\{\gamma(k-1,l-1), \gamma(k-1,l), \gamma(k,l-1)\}$$

- Algorithm:

DTW(*Sequences* **x** *and* **x'**)

  Let $\gamma(0,0) = 0; \gamma(k,0) = \infty; \gamma(0,l) = \infty$

  FOR $k = 1 \ldots T_x$

    FOR $l = 1 \ldots T_y$

    $\gamma(k,l) = (\mathbf{x}_k - \mathbf{x}_l)^2 + \min\{\gamma(k-1,l-1), \gamma(k-1,l), \gamma(k,l-1)\}$

  RETURN $\gamma(T_x, T_y)$

- Maximum-likelihood (ML) model:
  - $\theta_{ML} = \arg\max_{\theta} P(\mathbf{y}|\mathbf{X},\theta)$. — Likelihood
- Maximum-a-posteriori (MAP) model:
  - $\theta_{MAP} = \arg\max_{\theta} P(\theta|\mathbf{y},\mathbf{X})$ — A posteriori ("posterior") distribution

$$\theta_{MAP} = \arg\max_{\theta} P(\theta|\mathbf{y},\mathbf{X}) = \arg\max_{\theta} \frac{P(\mathbf{y}|\mathbf{X},\theta)P(\theta)}{P(\mathbf{y}|\mathbf{X})}$$
$$= \arg\max_{\theta} P(\mathbf{y}|\mathbf{X},\theta)P(\theta)$$

Posterior ∝ likelihood x prior

- Most likely value $\mathbf{y}^*$ for new input $\mathbf{x}^*$ (Bayes-optimal decision):
  - $\mathbf{y}^* = \arg\max_y P(y|\mathbf{x}^*,\mathbf{y},\mathbf{X})$
  - $P(y^*|\mathbf{x}^*,\mathbf{y},\mathbf{X}) = \int P(y^*,\theta|\mathbf{x}^*,\mathbf{y},\mathbf{X})d\theta$
    $= \int P(y^*|\mathbf{x}^*,\theta)P(\theta|\mathbf{y},\mathbf{X})d\theta$

Predictive distribution

"Bayesian model averaging". Often computationally infeasible, but has a closed-form solution in some cases.

15