Upcoming lectures:
The next two lectures include hands-on-practice

- May 2 (today): Classification [Kashima]
- May 13: Model evaluation and selection [Takeuchi]
- May 20: Hands-on practice [Takeuchi]
- May 27: Feature selection [Yamada]
- June 3: Dimensionality reduction [Yamada]
- …

Bring your own laptop (recommended)

Kyoto University

Statistical Learning Theory
- Classification -

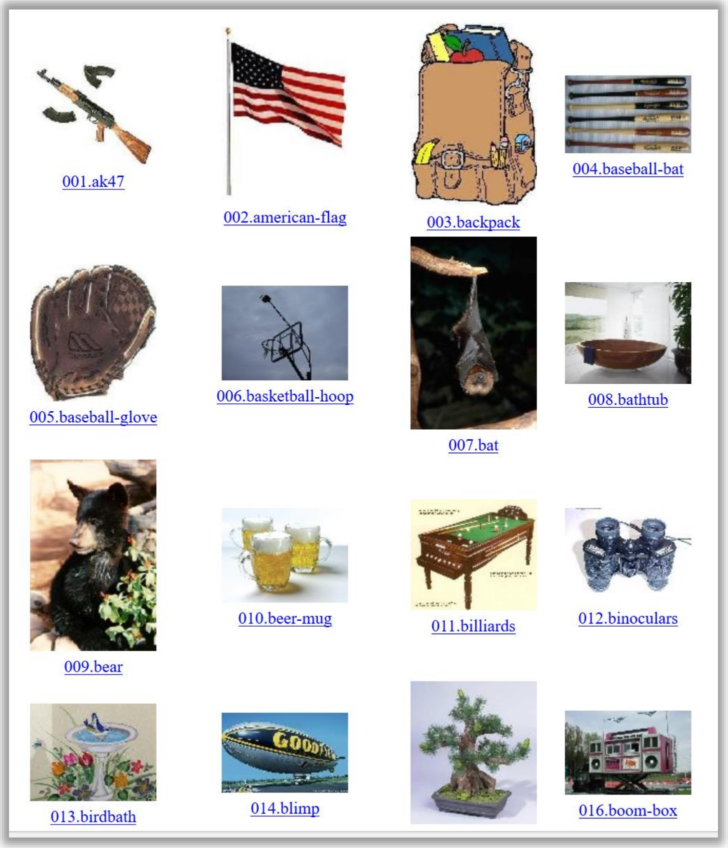
Hisashi Kashima

Department of Intelligence Science and Technology

Classification

Classification: Supervised learning for predicting discrete variable

- Goal: Obtain a function $f: \mathcal{X} \to \mathcal{Y}$ ($\mathcal{Y}$: discrete domain)
  - E.g. $x \in \mathcal{X}$ is an image and $y \in \mathcal{Y}$ is the type of object appearing in the image
  - Two-class classification: $\mathcal{Y} = \{+1, -1\}$
- Training dataset: N pairs of an input and an output $\{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$



http://www.vision.caltech.edu/Image_Datasets/Caltech256/

Some applications of classification: From binary to multi-class classification

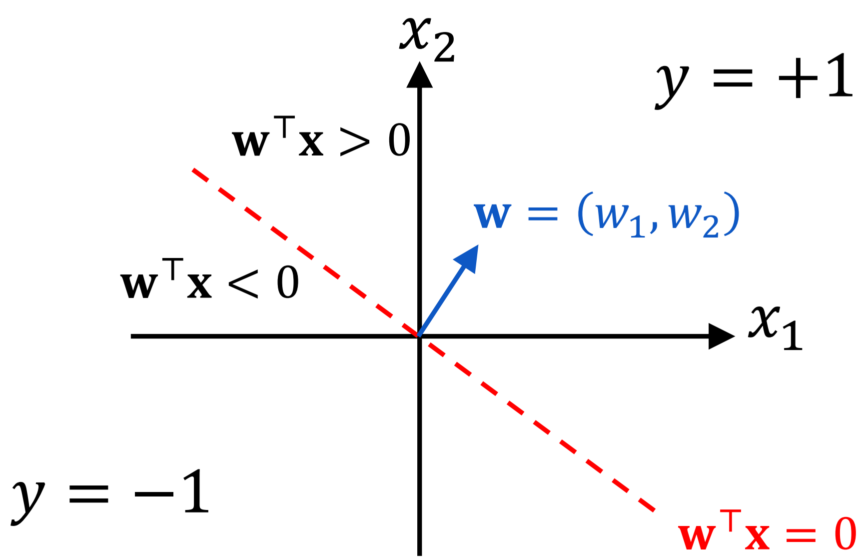
- Binary (two-class)classification:
  - Purchase prediction: Predict if a customer $\mathbf{x}$ will buy a particular product $(+1)$ or not (-1)
  - Credit risk prediction: Predict if a obligor $\mathbf{x}$ will pay back a debt $(+1)$ or not (-1)
- Multi-class classification ($\neq$ Multi-label classification):
  - Text classification: Categorize a document $\mathbf{x}$ into one of several categories, e.g., {politics, economy, sports, ...}
  - Image classification: Categorize the object in an image $\mathbf{x}$ into one of several object names, e.g., {AK5, American flag, backpack, ...}
  - Action recognition: Recognize the action type ({running, walking, sitting, ...}) that a person is taking from sensor data $\mathbf{x}$

A simple model for classification: Linear classifier

- Linear (binary) classification model:

$$y = \mathrm{sign}(\mathbf{w}^\top \mathbf{x}) = \mathrm{sign}(w_1 x_1 + w_2 x_2 + \cdots + w_D x_D)$$

  - $|\mathbf{w}^\top \mathbf{x}|$ indicates the intensity of belief
  - $\mathbf{w}^\top \mathbf{x} = 0$ gives a separating hyperplane
    - $\mathbf{w}$: normal vector perpendicular to the separating hyperplane



Learning framework: Loss minimization and statistical estimation

- Two learning frameworks
  1. Loss minimization: $L(\mathbf{w}) = \sum_{i=1}^{N} \ell\left(y^{(i)}, \mathbf{w}^{\top}\mathbf{x}^{(i)}\right)$
     - Loss function ℓ: directly handles utility of predictions
     - Regularization term $R(\mathbf{w})$
  2. Statistical estimation (likelihood maximization): $L(\mathbf{w}) = \prod_{i=1}^{N} f_{\mathbf{w}}(y^{(i)} | \mathbf{x}^{(i)})$
     - Probabilistic model: generation process of class labels
     - Prior distribution $P(\mathbf{w})$
- They are often equivalent : $\begin{cases} \text{Loss = Probabilistic model} \\ \text{Regularization = Prior} \end{cases}$

Classification problem in loss minimization framework: Minimize loss function + regularization term

- Minimization problem: $\mathbf{w}^* = \mathrm{argmin}_{\mathbf{w}}\, L(\mathbf{w}) + R(\mathbf{w})$
  - Loss function $L(\mathbf{w})$: Fitness to training data
  - Regularization term $R(\mathbf{w})$: Penalty on the model complexity to avoid overfitting to training data (usually norm of $\mathbf{w}$)
- Loss function should reflect the number of misclassifications on training data
  - Zero-one loss seems reasonable:

Correct classification

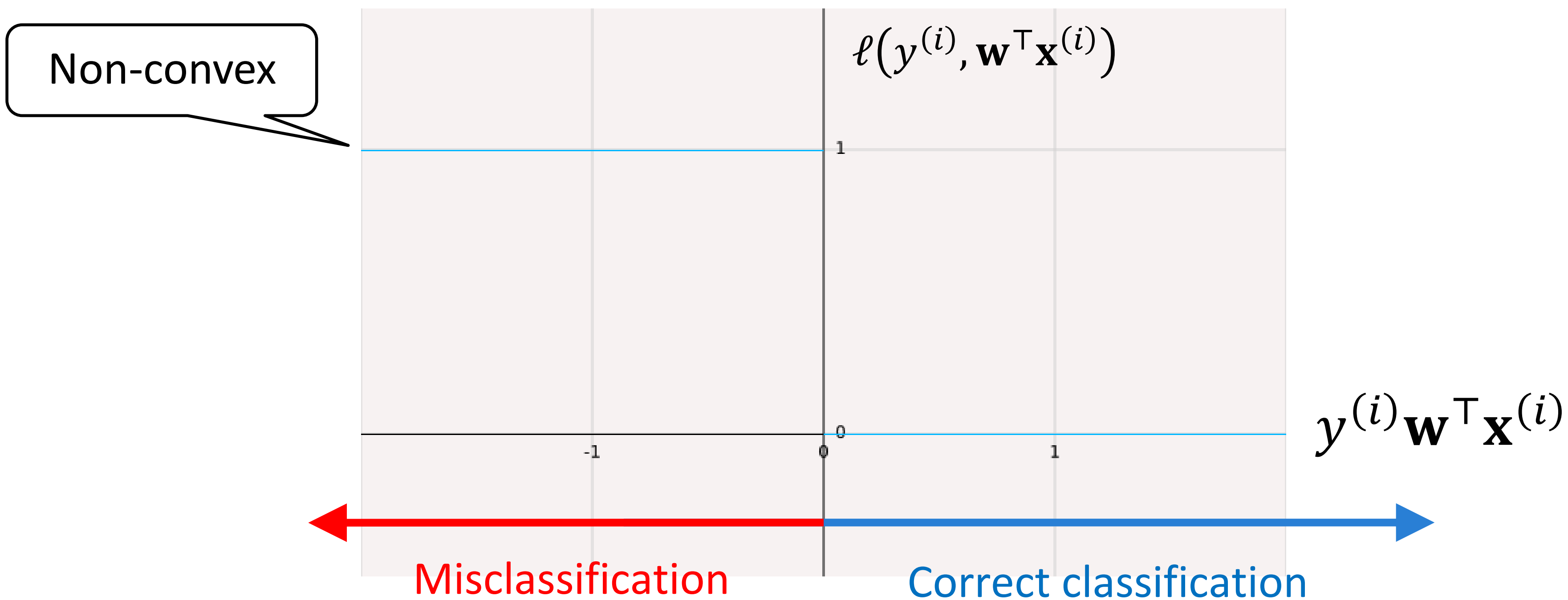
$$\ell^{(i)}\left(y^{(i)}, \mathbf{w}^\top \mathbf{x}^{(i)}\right) = \begin{cases} 0 & \left(y^{(i)} = \mathrm{sign}\left(\mathbf{w}^\top \mathbf{x}^{(i)}\right)\right) \\ 1 & \left(y^{(i)} \neq \mathrm{sign}\left(\mathbf{w}^\top \mathbf{x}^{(i)}\right)\right) \end{cases}$$

Incorrect classification

Zero-one loss:
Number of misclassification is hard to minimize

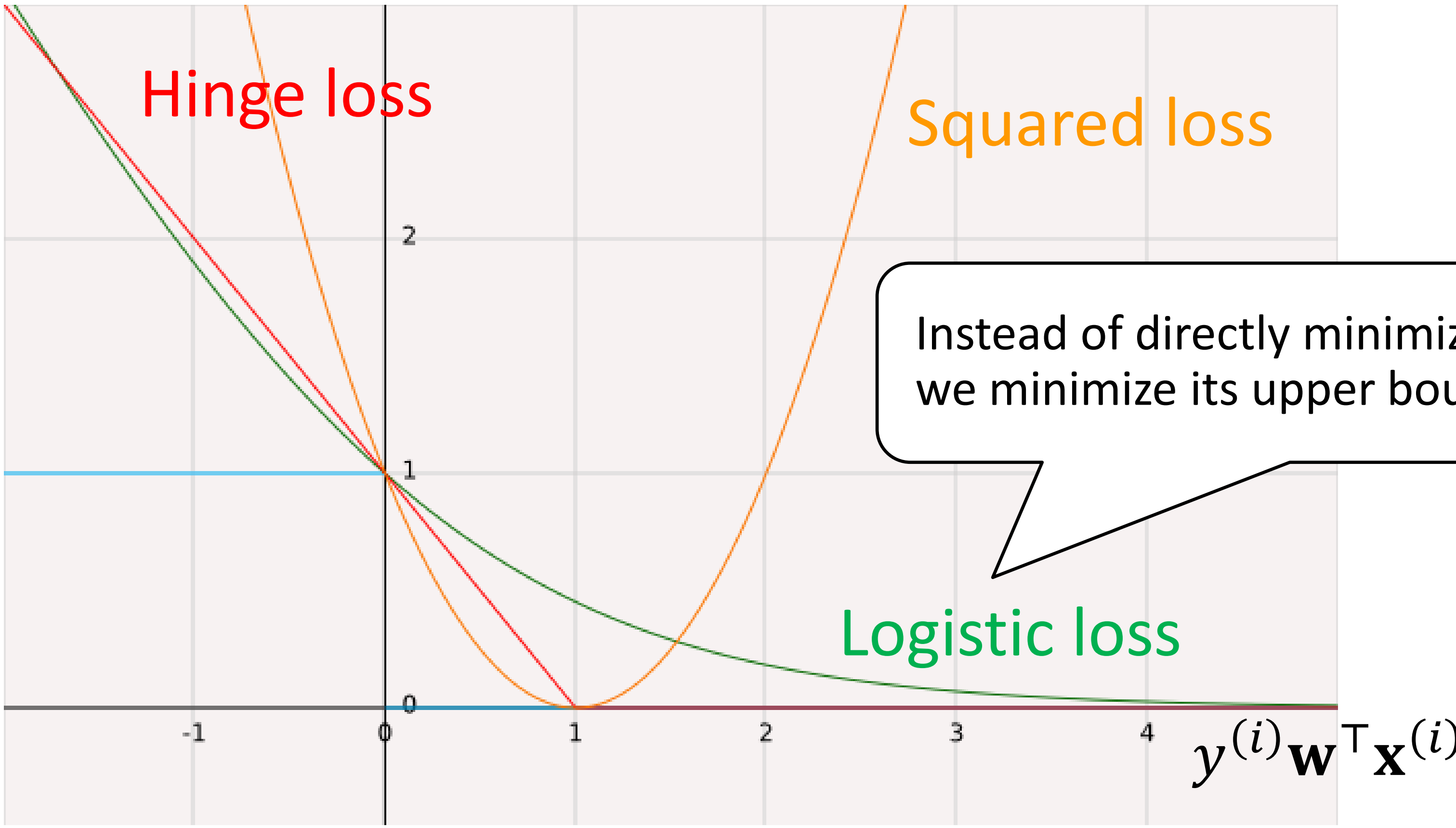
- Zero-one loss: $\ell\left(y^{(i)}, \mathbf{w}^\top \mathbf{x}^{(i)}\right) = \begin{cases} 0 & \left(y^{(i)} \mathbf{w}^\top \mathbf{x}^{(i)} > 0\right) \\ 1 & \left(y^{(i)} \mathbf{w}^\top \mathbf{x}^{(i)} \leq 0\right) \end{cases}$

- Non-convex function is hard to optimize directly



Convex surrogates of zero-one loss: Different functions lead to different learning machines

- Convex surrogates: Upper bounds of zero-one loss
  - Hinge loss → SVM, Logistic loss → logistic regression, ...



Instead of directly minimizing zero-one loss, we minimize its upper bound

Logistic regression

Logistic regression:
Minimization of logistic loss is a convex optimization

- Logistic loss:

$$\ell\left(y^{(i)}, \mathbf{w}^\top \mathbf{x}^{(i)}\right) = \frac{1}{\ln 2} \ln\left(1 + \exp\left(-y^{(i)} \mathbf{w}^\top \mathbf{x}^{(i)}\right)\right)$$

- (Regularized) Logistic regression:

Convex

$$\mathbf{w}^* = \mathrm{argmin}_{\mathbf{w}} \sum_{i=1}^{N} \ln\left(1 + \exp\left(-y^{(i)} \mathbf{w}^\top \mathbf{x}^{(i)}\right)\right) + \lambda \|\mathbf{w}\|_2^2$$

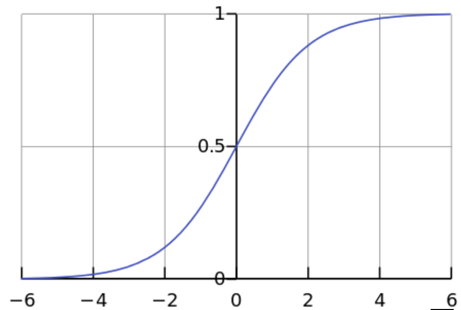
Statistical interpretation: Logistic loss min. as MLE of logistic regression model

- Minimization of logistic loss is equivalent to maximum likelihood estimation of logistic regression model

- Logistic regression model (conditional probability):

$$f_{\mathbf{w}}(y=1|\mathbf{x}) = \sigma(\mathbf{w}^\top\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^\top\mathbf{x})}$$

  - σ: Logistic function ($\sigma: \Re \to (0,1)$)



- Log likelihood:

$$L(\mathbf{w}) = \sum_{i=1}^{N} \log f_{\mathbf{w}}(y^{(i)}|\mathbf{x}^{(i)}) = -\sum_{i=1}^{N} \log\left(1+\exp\left(-y^{(i)}\mathbf{w}^\top\mathbf{x}\right)\right)$$

$$\left(=\sum_{i=1}^{N} \delta\left(y^{(i)}=1\right)\log\frac{1}{1+\exp(-\mathbf{w}^\top\mathbf{x})} + \delta\left(y^{(i)}=-1\right)\log\left(1-\frac{1}{1+\exp(-\mathbf{w}^\top\mathbf{x})}\right)\right)$$

Parameter estimation of logistic regression : Numerical nonlinear optimization

- Objective function of (regularized) logistic regression:

$$L(\mathbf{w}) = \sum_{i=1}^{N} \ln\left(1 + \exp\left(-y^{(i)}\mathbf{w}^\top \mathbf{x}^{(i)}\right)\right) + \lambda \|\mathbf{w}\|_2^2$$

- Minimization of logistic loss / MLE of logistic regression model has no closed form solution

- Numerical nonlinear optimization methods are used
  - Iterate parameter updates: $\mathbf{w}^{\mathrm{NEW}} \leftarrow \mathbf{w} + \mathbf{d}$ (until convergence)

Parameter update : Find the best update minimizing the objective function

- By update $\mathbf{w}^{\mathrm{NEW}} \leftarrow \mathbf{w} + \mathbf{d}$, the objective function will be:

$$L_{\mathbf{w}}(\mathbf{d}) = \sum_{i=1}^{N} \ln\left(1 + \exp\left(-y^{(i)}(\mathbf{w} + \mathbf{d})^\top \mathbf{x}^{(i)}\right)\right) + \lambda \|\mathbf{w} + \mathbf{d}\|_2^2$$

- Find $\mathbf{d}^*$ that minimizes $L_{\mathbf{w}}(\mathbf{d})$:
  - $\mathbf{d}^* = \mathrm{argmin}_{\mathbf{d}}\, L_{\mathbf{w}}(\mathbf{d})$
- … but so far, this problem has not been made easier at all … 😳

Finding the best parameter update :
Approximate the objective with Taylor expansion

- Taylor expansion:

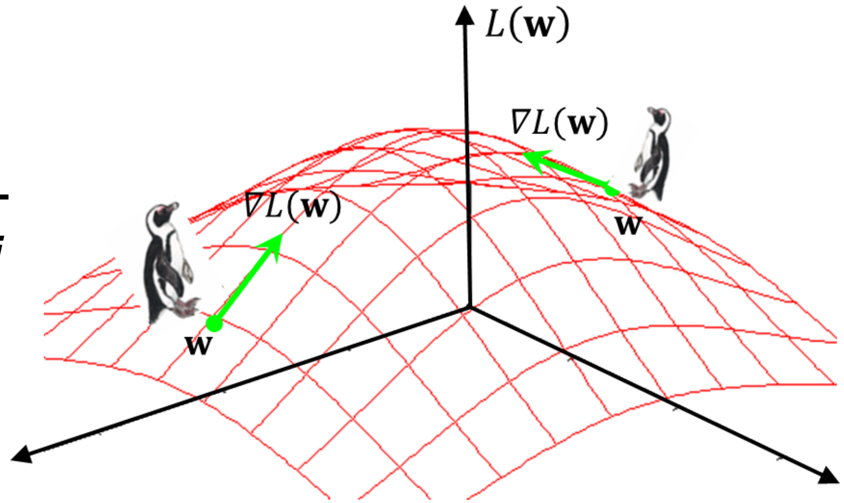
3rd-order term

$$L_{\mathbf{w}}(\mathbf{d}) = L(\mathbf{w}) + \mathbf{d}^\top \nabla L(\mathbf{w}) + \frac{1}{2}\mathbf{d}^\top \boldsymbol{H}(\mathbf{w})\mathbf{d} + O(\mathbf{d}^3)$$

– Gradient vector: $\nabla L(\mathbf{w}) = \left(\frac{\partial L(\mathbf{w})}{\partial w_1}, \frac{\partial L(\mathbf{w})}{\partial w_2}, \dots, \frac{\partial L(\mathbf{w})}{\partial w_D}\right)^\top$

  - Steepest direction

– Hessian matrix: $[H(\mathbf{w})]_{i,j} = \frac{\partial^2 L(\mathbf{w})}{\partial w_i \partial w_j}$



Newton update : Minimizes the second order approximation

- Approximated Taylor expansion (neglecting the 3rd order term):

$$L_{\mathbf{w}}(\mathbf{d}) \approx L(\mathbf{w}) + \mathbf{d}^\top \nabla L(\mathbf{w}) + \frac{1}{2}\mathbf{d}^\top \boldsymbol{H}(\mathbf{w})\mathbf{d} \cancel{+ O(\mathbf{d}^3)}$$

- Derivative w.r.t. $\mathbf{d}$: $\dfrac{\partial L_{\mathbf{w}}(\mathbf{d})}{\partial \mathbf{d}} \approx \nabla L(\mathbf{w}) + \boldsymbol{H}(\mathbf{w})\mathbf{d}$

- Setting it to be $\mathbf{0}$, we obtain $\mathbf{d} = -\boldsymbol{H}(\mathbf{w})^{-1}\nabla L(\mathbf{w})$

- Newton update formula:

$$\mathbf{w}^{\mathrm{NEW}} \leftarrow \mathbf{w} - \boldsymbol{H}(\mathbf{w})^{-1}\nabla L(\mathbf{w})$$

$$\mathbf{w} \xrightarrow{-\boldsymbol{H}(\mathbf{w})^{-1}\nabla L(\mathbf{w})} \mathbf{w} - \boldsymbol{H}(\mathbf{w})^{-1}\nabla L(\mathbf{w})$$

Modified Newton update:
Second order approximation + linear search

- The correctness of the update $\mathbf{w}^{\mathrm{NEW}} \leftarrow \mathbf{w} - \boldsymbol{H}(\mathbf{w})^{-1}\nabla L(\mathbf{w})$ depends on the second-order approximation:

$$L_{\mathbf{w}}(\mathbf{d}) \approx L(\mathbf{w}) + \mathbf{d}^{\top}\nabla L(\mathbf{w}) + \frac{1}{2}\mathbf{d}^{\top}\boldsymbol{H}(\mathbf{w})\mathbf{d}$$

  - This is not actually true for most cases

- Use only the direction of $\boldsymbol{H}(\mathbf{w})^{-1}\nabla L(\mathbf{w})$ and update with $\mathbf{w}^{\mathrm{NEW}} \leftarrow \mathbf{w} - \eta\boldsymbol{H}(\mathbf{w})^{-1}\nabla L(\mathbf{w})$

- Learning rate $\eta > 0$ is determined by linear search:

$$\eta^* = \mathrm{argmax}_\eta\, L\left(\mathbf{w} - \eta\boldsymbol{H}(\mathbf{w})^{-1}\nabla L(\mathbf{w})\right)$$

(Steepest) gradient descent: Simple update without computing inverse Hessian

- Computing the inverse of Hessian matrix is costly
  - Newton update: $\mathbf{w}^{\mathrm{NEW}} \leftarrow \mathbf{w} - \eta \boldsymbol{H}(\mathbf{w})^{-1} \nabla L(\mathbf{w})$
- (Steepest) gradient descent:
  - Replacing $\boldsymbol{H}(\mathbf{w})^{-1}$ with $\boldsymbol{I}$ gives

Gradient of objective function

$$\mathbf{w}^{\mathrm{NEW}} \leftarrow \mathbf{w} - \eta \nabla L(\mathbf{w})$$

  - $\nabla L(\mathbf{w})$ is the steepest direction
  - Learning rate η is determined by line search

Summary:
Gradient descent

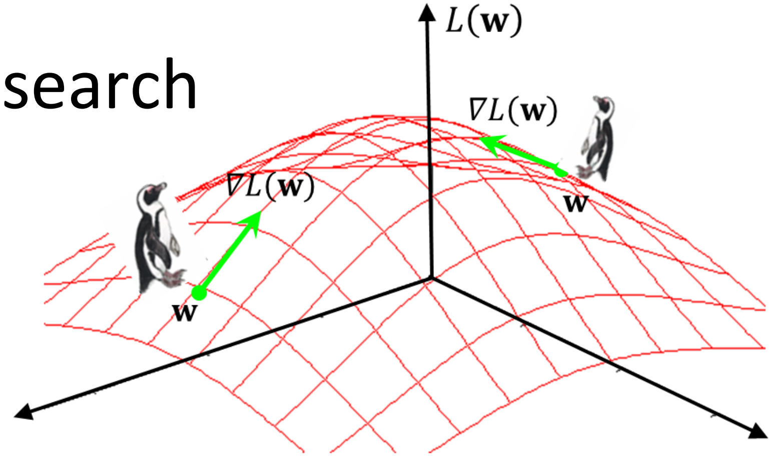
- Steepest gradient descent is the simplest optimization method:
- Update the parameter in the steepest direction of the objective function

$$\mathbf{w}^{\mathrm{NEW}} \leftarrow \mathbf{w} - \eta \nabla L(\mathbf{w})$$

  – Gradient: $\nabla L(\mathbf{w}) = \left(\frac{\partial L(\mathbf{w})}{\partial w_1}, \frac{\partial L(\mathbf{w})}{\partial w_2}, \dots, \frac{\partial L(\mathbf{w})}{\partial w_D} \right)^{\top}$

  – Learning rate η is determined by line search

$$\mathbf{w} \xrightarrow{-\eta \nabla L(\mathbf{w})} \mathbf{w} - \eta \nabla L(\mathbf{w})$$



Example of gradient descent: Gradient of logistic regression

- $L(\mathbf{w}) = \sum_{i=1}^{N} \ln\left(1 + \exp\left(-y^{(i)}\mathbf{w}^\top\mathbf{x}^{(i)}\right)\right)$

- $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^{N} \frac{1}{1+\exp\left(-y^{(i)}\mathbf{w}^\top\mathbf{x}^{(i)}\right)} \frac{\partial\left(1+\exp\left(-y^{(i)}\mathbf{w}^\top\mathbf{x}^{(i)}\right)\right)}{\partial \mathbf{w}}$

$$= -\sum_{i=1}^{N} \frac{1}{1 + \exp\left(-y^{(i)}\mathbf{w}^\top\mathbf{x}^{(i)}\right)} \exp\left(-y^{(i)}\mathbf{w}^\top\mathbf{x}^{(i)}\right) y^{(i)}\mathbf{x}^{(i)}$$

$$= -\sum_{i=1}^{N} \left(1 - f_{\mathbf{w}}\left(y^{(i)} | \mathbf{x}^{(i)}\right)\right) y^{(i)}\mathbf{x}^{(i)}$$

Can be easily computed with the current prediction probabilities

Mini batch optimization:
Efficient training using data subsets

- Objective function for N instances:
  $L(\mathbf{w}) = \sum_{i=1}^{N} \ell\left(\mathbf{w}^\top \mathbf{x}^{(i)}\right) + \lambda R(\mathbf{w})$

- Its derivative $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^{N} \frac{\partial \ell\left(\mathbf{w}^\top \mathbf{x}^{(i)}\right)}{\partial \mathbf{w}} + \lambda \frac{\partial R(\mathbf{w})}{\partial \mathbf{w}}$ needs $O(N)$ computation

- Approximate this with only one instance:
  $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} \approx N \frac{\partial \ell\left(\mathbf{w}^\top \mathbf{x}^{(j)}\right)}{\partial \mathbf{w}} + \lambda \frac{\partial R(\mathbf{w})}{\partial \mathbf{w}}$ (Stochastic approximation)

- Also we can do this with $1 < M < N$ instances:
  $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} \approx \frac{N}{M} \sum_{j \in \mathrm{MiniBatch}} \frac{\partial \ell\left(\mathbf{w}^\top \mathbf{x}^{(j)}\right)}{\partial \mathbf{w}} + \lambda \frac{\partial R(\mathbf{w})}{\partial \mathbf{w}}$ (Mini batch)

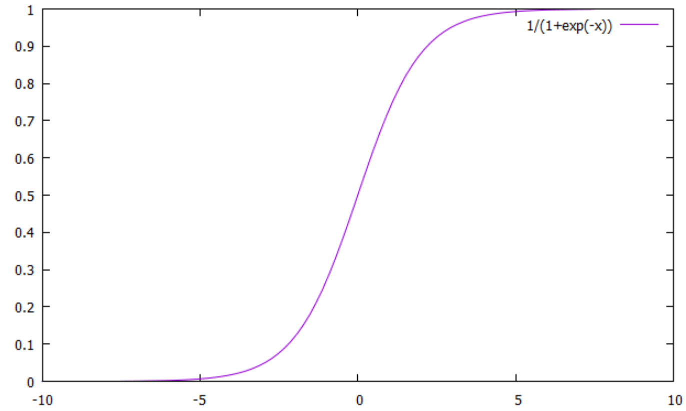
Multi-class Classification

Multi-class classification:
Generalization of supervised two-class classification

- Training dataset: $\{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(i)}, y^{(i)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$
  - input $\mathbf{x}^{(i)} \in \mathcal{X} = \mathbb{R}^D$: D-dimensional real vector
  - output $y^{(i)} \in \mathcal{Y}$: one-dimensional scalar
- Estimate a *deterministic mapping* $f: \mathcal{X} \rightarrow \mathcal{Y}$ (often with a confidence value) or a *conditional probability* $P(y|\boldsymbol{x})$
- Classification
  - $\mathcal{Y} = \{+1, -1\}$: Two-class classification
  - $\mathcal{Y} = \{1, 2, \ldots, K\}$: K-class multi-class classification
    - hand-written digit recognition, text classification, …

Two-class classification model: One model with one parameter vector

- Two-class classification model
  - Linear classifier: $f(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^\top \mathbf{x}) \in \{+1, -1\}$
  - Logistic regression: $P(y|\mathbf{x}) = \dfrac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$
  - The model is specified by a parameter vector $\mathbf{w} = (w_1, w_2, \ldots, w_D)^\top$



Simple approaches to multi-class classification: Reduction to two-class classification

- Reduction to a set of two-class classification problems
- Approach 1: One-versus-rest
  - Construct K two-class classifiers; each classifier $\mathrm{sign}(\mathbf{w}^{(k)\top}\mathbf{x})$ discriminates class k from the others
  - Prediction: the most probable class with the largest $\mathbf{w}^{(k)\top}\mathbf{x}$

confidence

- Approach 2: One-versus-one
  - Construct $K(K-1)/2$ two-class classifiers, each of which discriminates between a pair of two classes
  - Prediction by voting

Error Correcting Output Code (ECOC) : An approach inspired by error correcting coding

- Approach 3: Error correcting output code (ECOC)
  - Construct a set of two-class classifiers, each of which discriminates between two groups of classes, e.g. AB vs. CD
  - Prediction by finding the nearest code in terms of Hamming distance

codes

class	two-class classification problems					
	1	2	3	4	5	6
A	1	1	1	1	1	1
B	1	-1	1	-1	-1	-1
C	-1	-1	-1	1	-1	1
D	-1	1	1	-1	-1	1
prediction	**1**	**1**	**1**	**1**	**1**	**-1**

code for class A

Design of ECOC :
Code design is the key for good classification

- Codes (row) should be apart from each other in terms of Hamming distance

codes

class	two-class classification problems					
	1	2	3	4	5	6
A	1	1	1	1	1	1
B	1	-1	1	-1	-1	-1
C	-1	-1	-1	1	-1	1
D	-1	1	1	-1	-1	1

Hamming distances between codes

class	A	B	C	D
A	0	4	4	3
B		0	4	3
C			0	3
D				0

Multi-class logistic regression model:
One model parameter vector for each class

- More direct modeling of multi-class classification
  - One parameter vector $\mathbf{w}^{(k)}$ for each class k
  - Multi-class linear classifier: $f(\boldsymbol{x}) = \underset{k \in \mathcal{Y}}{\operatorname{argmax}} \, \mathbf{w}^{(k)\top}\mathbf{x}$
  - Multi-class logistic regression: $P(k|\boldsymbol{x}) = \dfrac{\exp(\mathbf{w}^{(k)\top}\mathbf{x})}{\sum_{k' \in \mathcal{Y}} \exp(\mathbf{w}^{(k')\top}\mathbf{x})}$
    - converts real values into positive values, and then normalizes them to obtain a probability value $\in [0,1]$

Training multi-class logistic regression: (Regularized) maximum likelihood estimation

- Find the parameters that minimizes the negative log-likelihood

$$J\left(\{\mathbf{w}^{(y)}\}_y\right) = -\sum_{i=1,\dots,N} \log p(y^{(i)}|\mathbf{x}^{(i)}) + \gamma \sum_{y\in\mathcal{Y}} \|\mathbf{w}^{(y)}\|_2^2$$

  - $\|\mathbf{w}^{(y)}\|_2^2$: a regularizer to avoid overfitting

- For multi-class logistic regression $P(k|\boldsymbol{x}) = \frac{\exp(\mathbf{w}^{(k)\top}\mathbf{x})}{\sum_{k'\in\mathcal{Y}}\exp(\mathbf{w}^{(k')\top}\mathbf{x})}$

$$J = -\sum_i \mathbf{w}^{(k)\top}\mathbf{x}^{(i)} + \sum_i \log \sum_{k'\in\mathcal{Y}} \exp\left(\mathbf{w}^{(k)\top}\mathbf{x}^{(i)}\right) + \text{reg.}$$

  – Minimization using gradient-based optimization methods