Energy-Based Learning

Yann LeCun

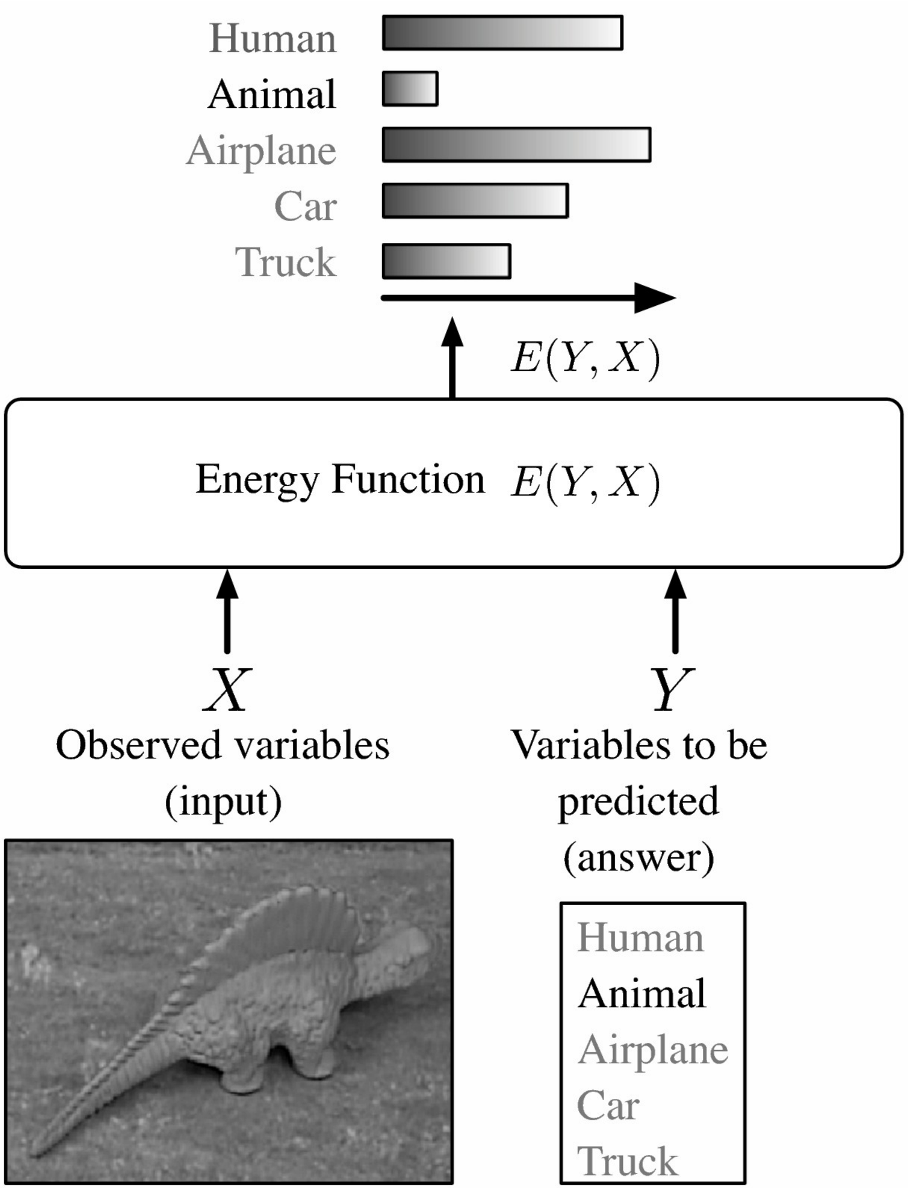
The Courant Institute of Mathematical Sciences

New York University

http://yann.lecun.com

http://www.cs.nyu.edu/~yann

Energy-Based Model for Decision-Making

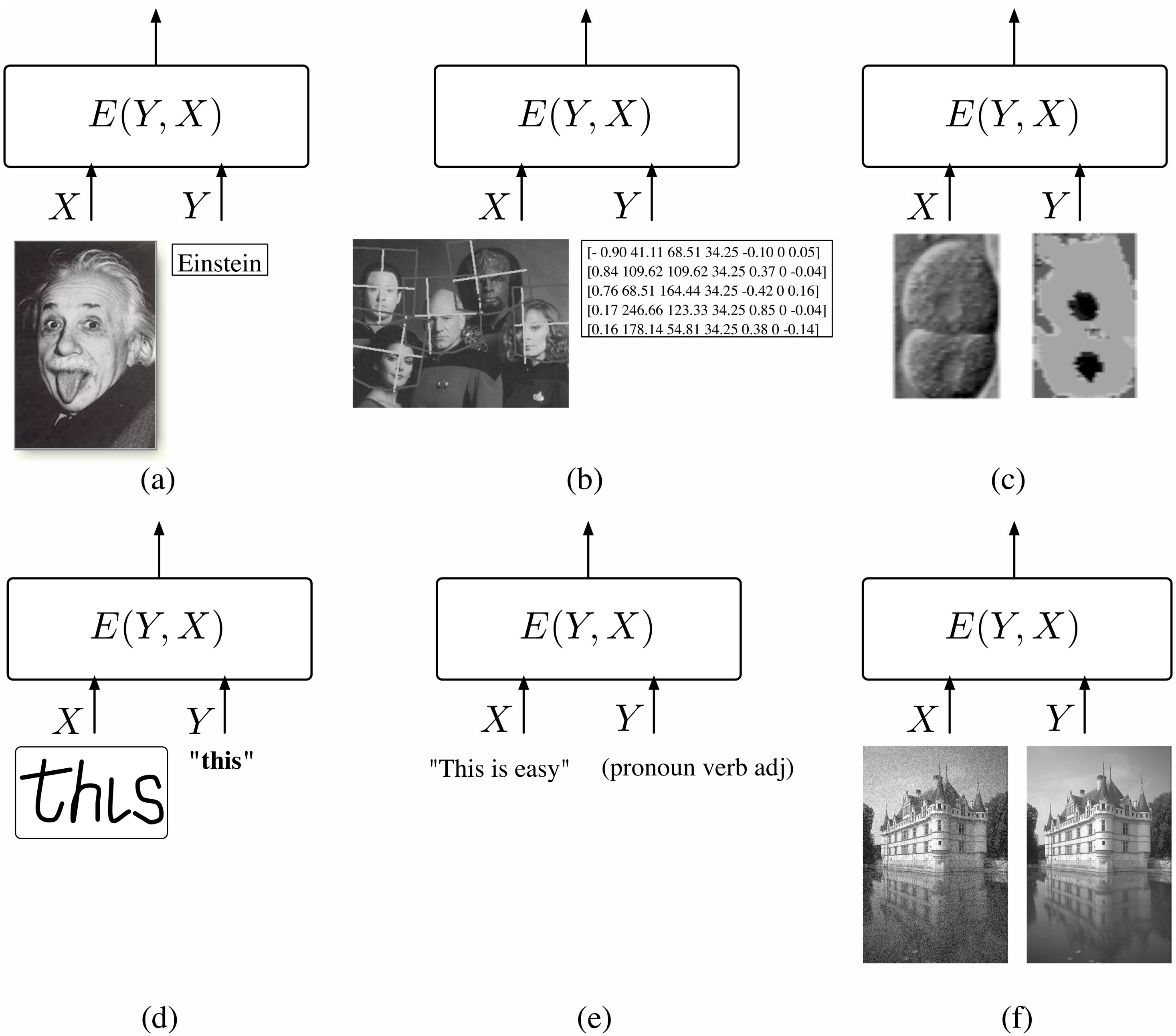


- **Model:** **Measures the compatibility between an observed variable X and a variable to be predicted Y through an energy function E(Y,X).**

$$Y^* = \text{argmin}_{Y \in \mathcal{Y}} E(Y, X).$$

- **Inference:** **Search for the Y that minimizes the energy within a set** $\mathcal{Y}$
- **If the set has low cardinality, we can use exhaustive search.**

Complex Tasks: Inference is non-trivial



- **When the cardinality or dimension of Y is large, exhaustive search is impractical.**
- **We need to use a “smart” inference procedure: min-sum, Viterbi, ...**

What Questions Can a Model Answer?

- **1. Classification & Decision Making:**
  - "which value of Y is most compatible with X?"
  - Applications: Robot navigation,.....
  - Training: give the lowest energy to the correct answer
- **2. Ranking:**
  - "Is Y1 or Y2 more compatible with X?"
  - Applications: Data-mining....
  - Training: produce energies that rank the answers correctly
- **3. Detection:**
  - "Is this value of Y compatible with X"?
  - Application: face detection....
  - Training: energies that increase as the image looks less like a face.
- **4. Conditional Density Estimation:**
  - "What is the conditional distribution P(Y|X)?"
  - Application: feeding a decision-making system
  - Training: differences of energies must be just so.

Decision-Making versus Probabilistic Modeling

Energies are uncalibrated

- The energies of two separately-trained systems cannot be combined
- The energies are uncalibrated (measured in arbitrary untis)

How do we calibrate energies?

- We turn them into probabilities (positive numbers that sum to 1).
- Simplest way: Gibbs distribution
- Other ways can be reduced to Gibbs by a suitable redefinition of the energy.

$$P(Y|X) = \frac{e^{-\beta E(Y,X)}}{\int_{y \in \mathcal{Y}} e^{-\beta E(y,X)}},$$

Partition function

Inverse temperature

Architecture and Loss Function

- **Family of energy functions** $\mathcal{E} = \{E(W,Y,X) : \ W \in \mathcal{W}\}.$
- **Training set** $\mathcal{S} = \{(X^i, Y^i) : i = 1 \dots P\}$
- **Loss functional / Loss function** $\mathcal{L}(E, \mathcal{S})$ $\mathcal{L}(W, \mathcal{S})$
  - Measures the quality of an energy function
- **Training** $W^* = \min_{W \in \mathcal{W}} \mathcal{L}(W, \mathcal{S}).$
- **Form of the loss functional**
  - invariant under permutations and repetitions of the samples

$$\mathcal{L}(E, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^{P} L(Y^i, E(W, \mathcal{Y}, X^i)) + R(W).$$

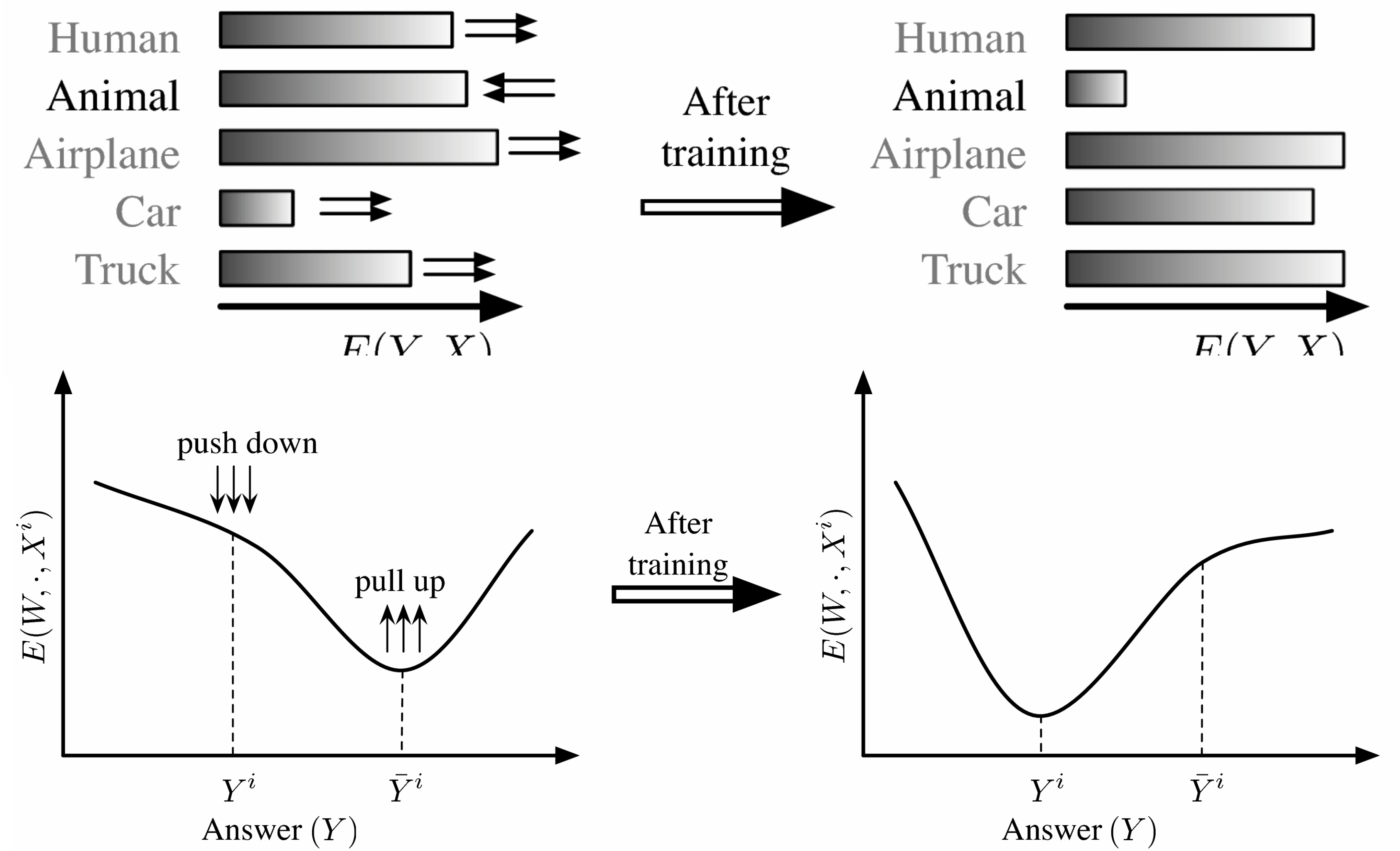
Per-sample loss → L

Desired answer → Y^i

Energy surface for a given Xi as Y varies → $E(W, \mathcal{Y}, X^i)$

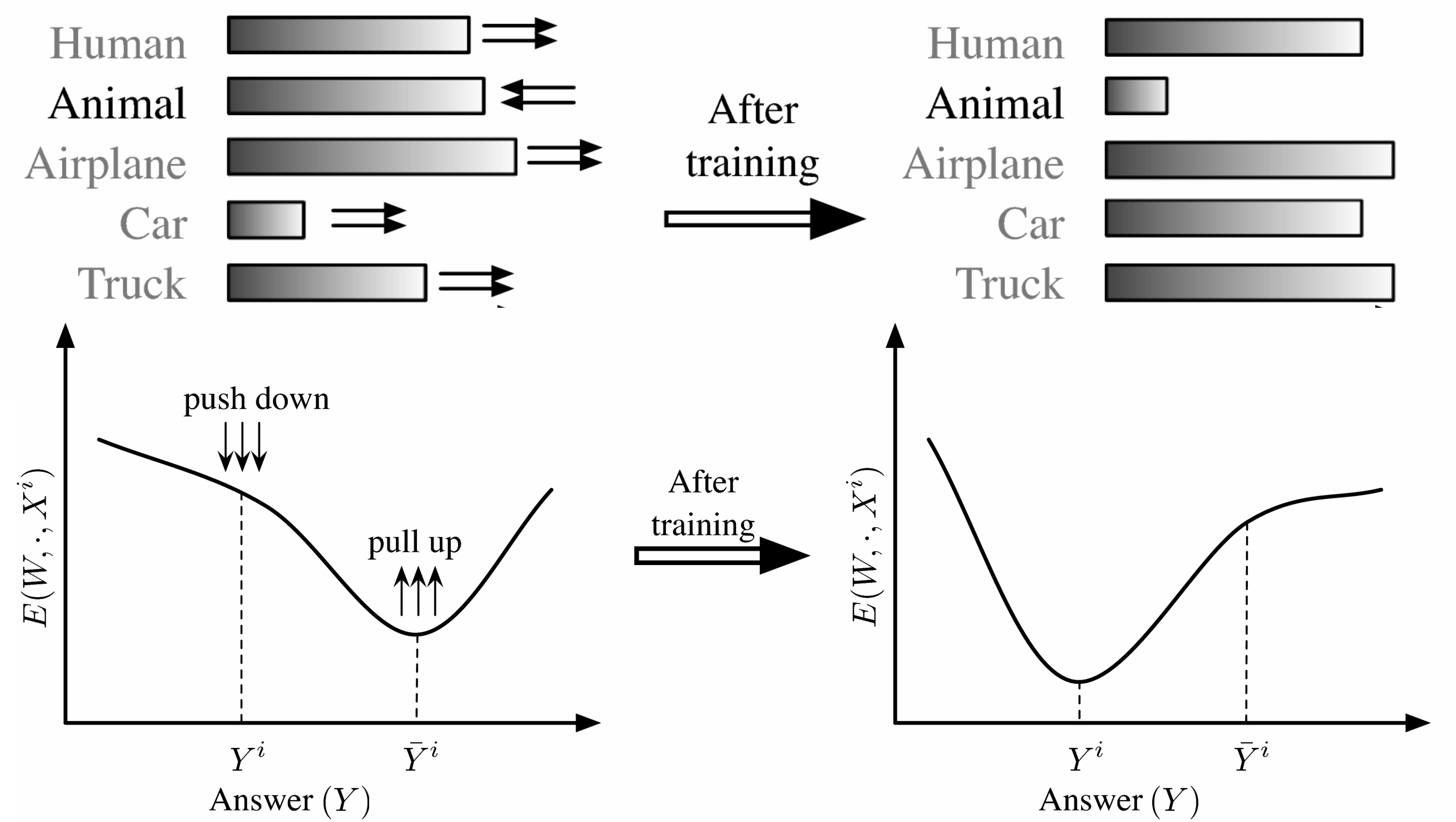
Regularizer → $R(W)$

Designing a Loss Functional



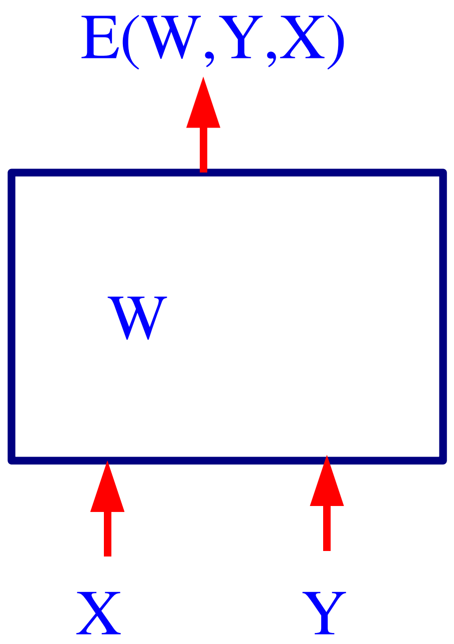
- **Correct answer has the lowest energy -> LOW LOSS**
- **Lowest energy is not for the correct answer -> HIGH LOSS**

Designing a Loss Functional



- **Push down on the energy of the correct answer**
- **Pull up on the energies of the incorrect answers, particularly if they are smaller than the correct one**

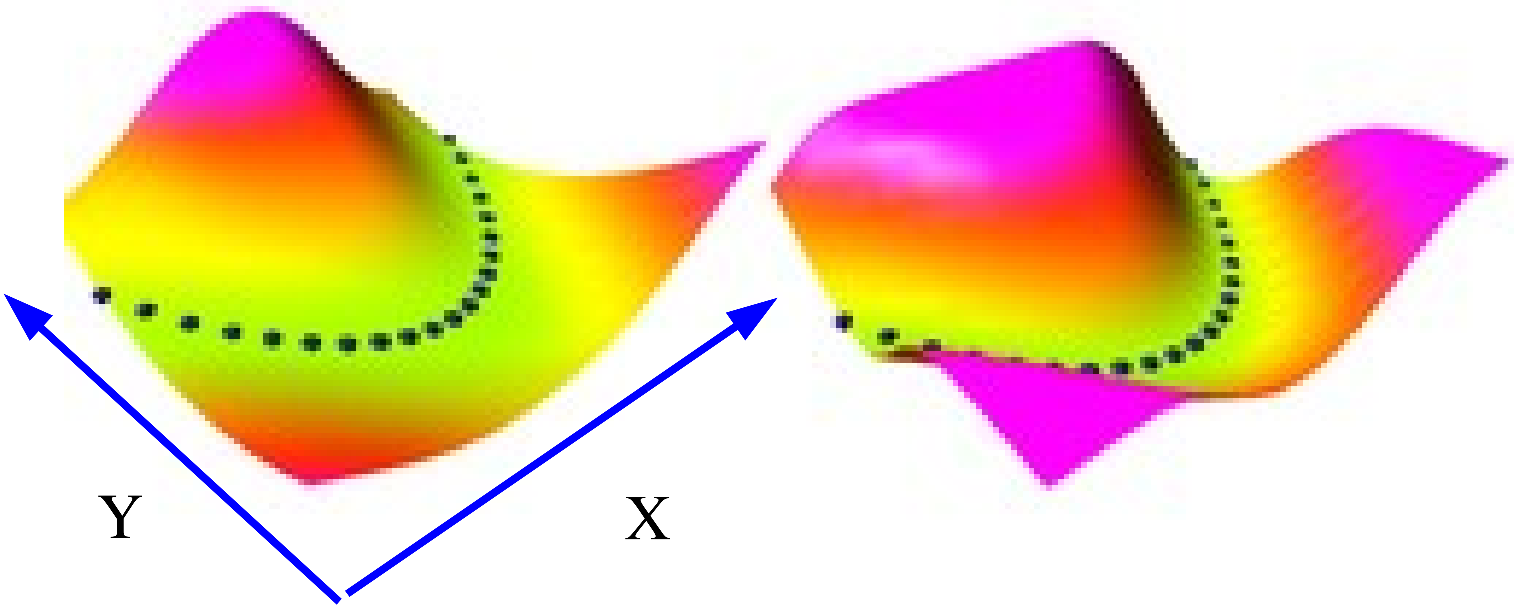
Architecture + Inference Algo + Loss Function = Model



- **1. Design an architecture:** a particular form for E(W,Y,X).
- **2. Pick an inference algorithm for Y:** MAP or conditional distribution, belief prop, min cut, variational methods, gradient descent, MCMC, HMC.....
- **3. Pick a loss function:** in such a way that minimizing it with respect to W over a training set will make the inference algorithm find the correct Y for a given X.
- **4. Pick an optimization method.**

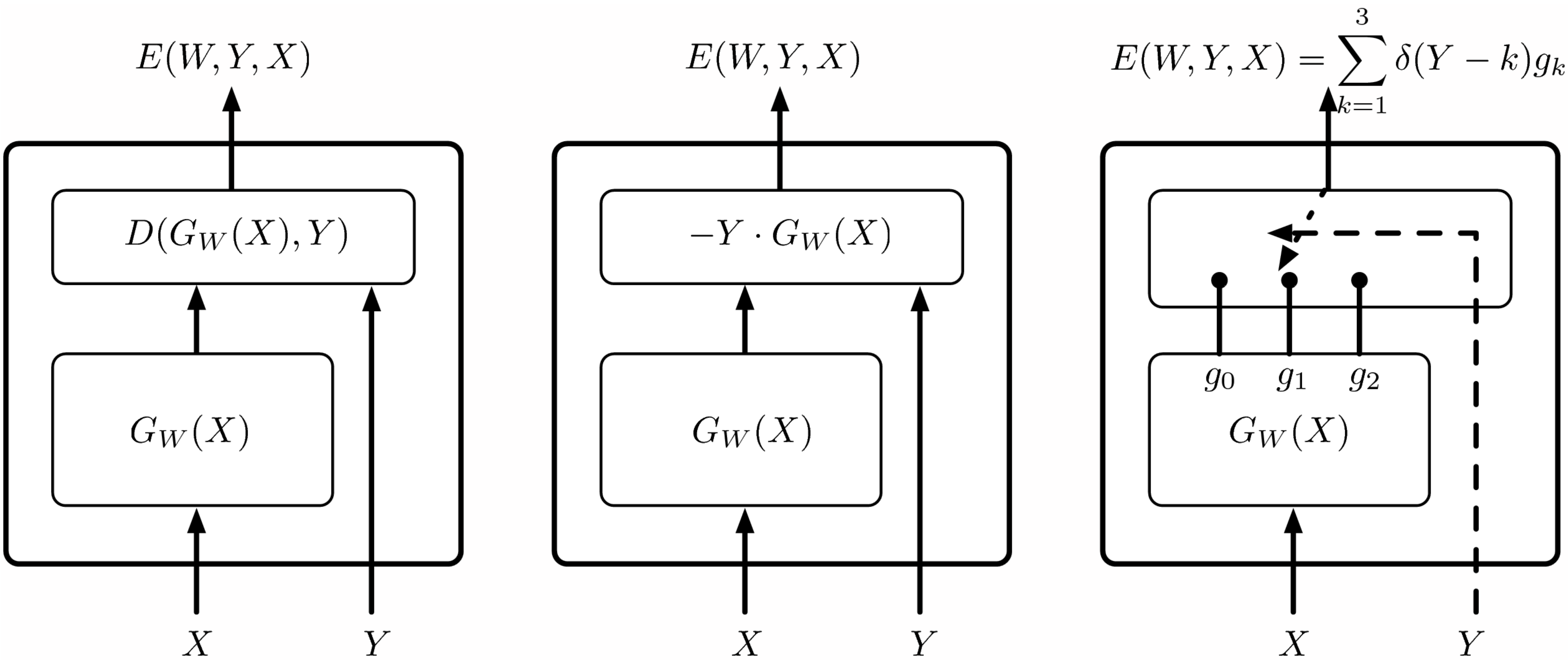
- **PROBLEM: What loss functions will make the machine approach the desired behavior?**

Several Energy Surfaces can give the same answers



- Both surfaces compute Y=X^2
- MINy E(Y,X) = X^2
- Minimum-energy inference gives us the same answer

Simple Architectures



- **Regression**

$$E(W,Y,X) = \frac{1}{2}||G_W(X) - Y||^2.$$

- **Binary Classification**

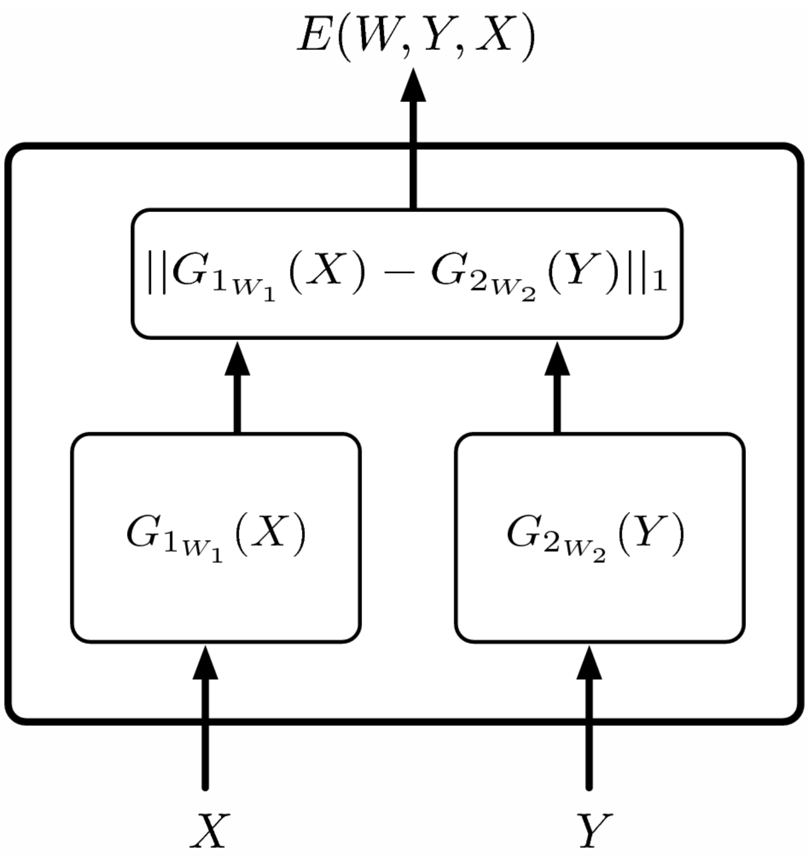
$$E(W,Y,X) = -YG_W(X),$$

- **Multi-class Classification**

Simple Architecture: Implicit Regression

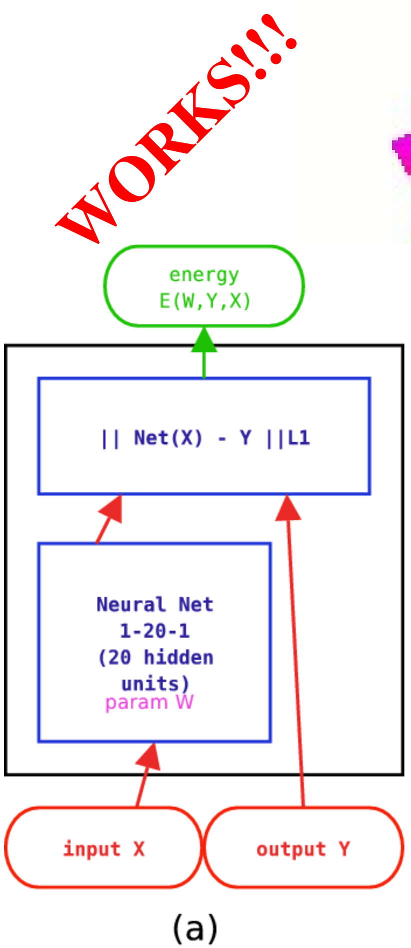
$$E(W, X, Y) = ||G_{1_{W_1}}(X) - G_{2_{W_2}}(Y)||_1,$$

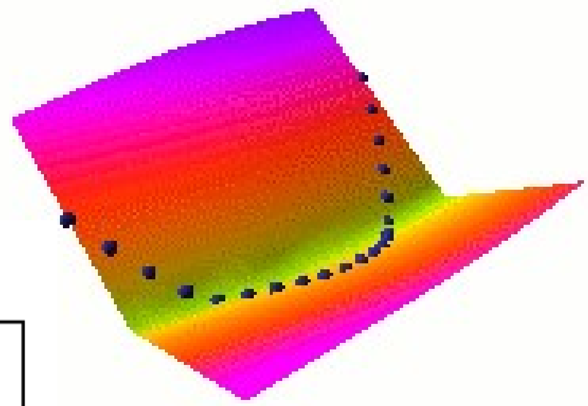
- **The Implicit Regression architecture**
  - allows multiple answers to have low energy.
  - Encodes a constraint between X and Y rather than an explicit functional relationship
  - This is useful for many applications
  - Example: sentence completion: "The cat ate the {mouse,bird,homework,...}"
  - [Bengio et al. 2003]
  - But, inference may be difficult.

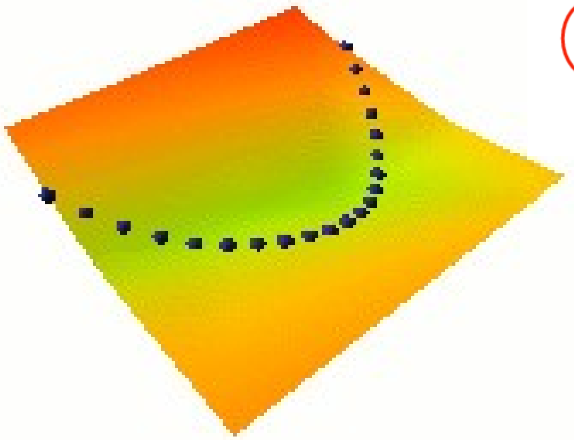


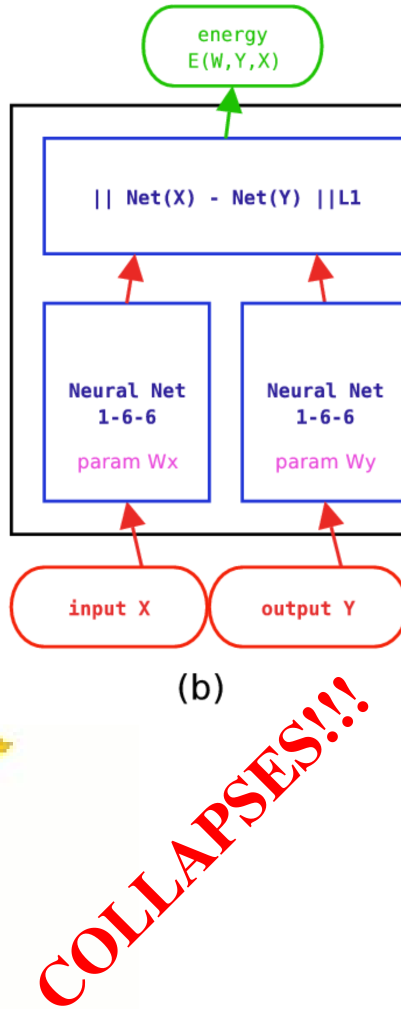
Examples of Loss Functions: Energy Loss

- **Energy Loss** $$L_{energy}(Y^i, E(W, \mathcal{Y}, X^i)) = E(W, Y^i, X^i).$$
  - Simply pushes down on the energy of the correct answer









Examples of Loss Functions:Perceptron Loss

$$L_{perceptron}(Y^i, E(W, \mathcal{Y}, X^i)) = E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i).$$

Perceptron Loss

- Pushes down on the energy of the correct answer
- Pulls up on the energy of the machine's answer
- Always positive. Zero when answer is correct
- No “margin”: technically does not prevent the energy surface from being almost flat.
- Works pretty well in practice, particularly if the energy parameterization does not allow flat surfaces.

Perceptron Loss for Binary Classification

$$L_{perceptron}(Y^i, E(W, \mathcal{Y}, X^i)) = E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i).$$

- **Energy:** $E(W, Y, X) = -Y G_W(X),$
- **Inference:** $Y^* = \text{argmin}_{Y \in \{-1,1\}} - Y G_W(X) = \text{sign}(G_W(X)).$
- **Loss:** $\mathcal{L}_{\text{perceptron}}(W, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^{P} \left(\text{sign}(G_W(X^i)) - Y^i\right) G_W(X^i).$
- **Learning Rule:** $W \leftarrow W + \eta \left(Y^i - \text{sign}(G_W(X^i))\right) \frac{\partial G_W(X^i)}{\partial W},$
- **If Gw(X) is linear in W:** $E(W, Y, X) = -Y W^T \Phi(X)$

$$W \leftarrow W + \eta \left(Y^i - \text{sign}(W^T \Phi(X^i))\right) \Phi(X^i)$$

Examples of Loss Functions: Generalized Margin Losses

- **First, we need to define the Most Offending Incorrect Answer**

- **Most Offending Incorrect Answer: discrete case**

Definition 1 *Let Y be a discrete variable. Then for a training sample (X^i, Y^i), the* ***most offending incorrect answer*** *$\bar{Y}^i$ is the answer that has the lowest energy among all answers that are incorrect:*

$$\bar{Y}^i = \text{argmin}_{Y \in \mathcal{Y} and Y \neq Y^i} E(W, Y, X^i). \tag{8}$$

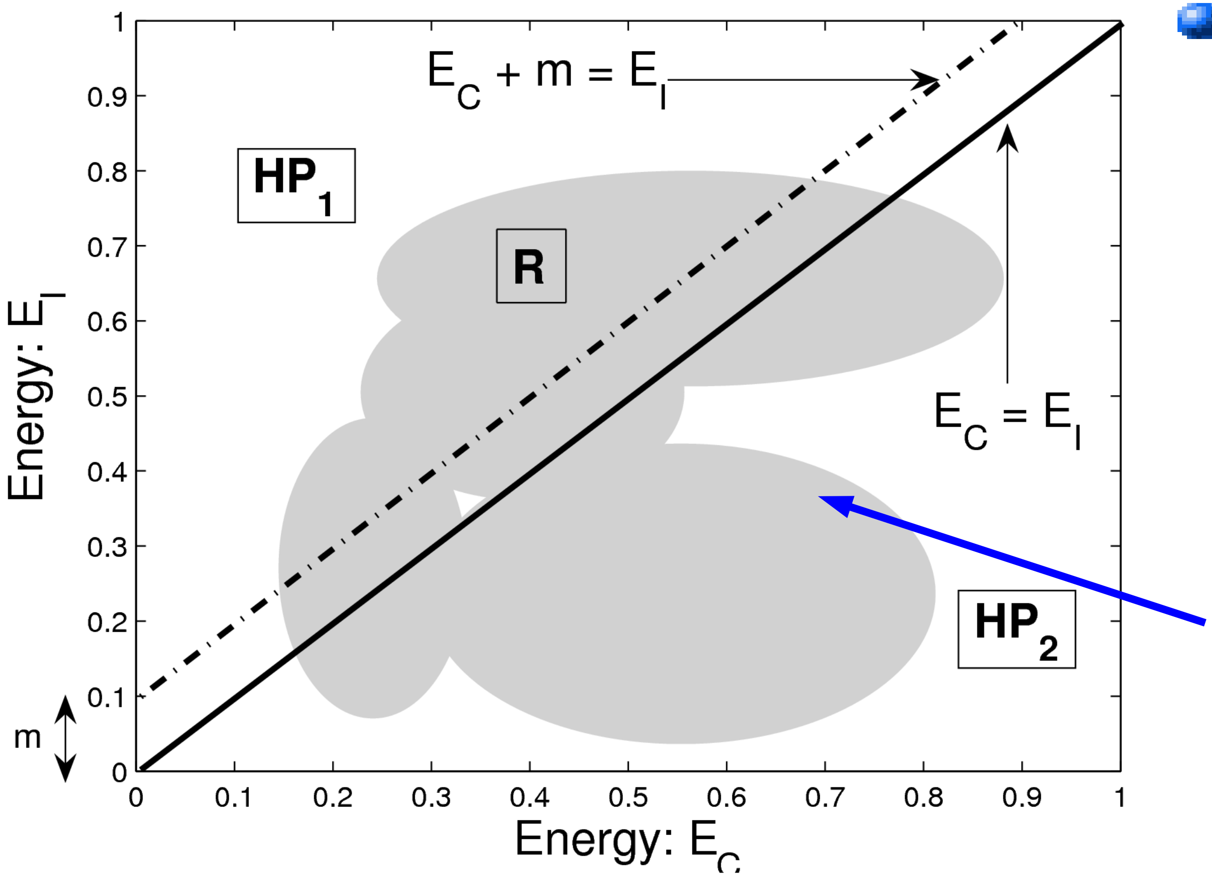
- **Most Offending Incorrect Answer: continuous case**

Definition 2 *Let Y be a continuous variable. Then for a training sample (X^i, Y^i), the* ***most offending incorrect answer*** *$\bar{Y}^i$ is the answer that has the lowest energy among all answers that are at least ϵ away from the correct answer:*

$$\bar{Y}^i = \text{argmin}_{Y \in \mathcal{Y}, \|Y - Y^i\| > \epsilon} E(W, Y, X^i). \tag{9}$$

Examples of Loss Functions: Generalized Margin Losses

$$L_{\text{margin}}(W, Y^i, X^i) = Q_m\left(E(W, Y^i, X^i), E(W, \bar{Y}^i, X^i)\right).$$



Generalized Margin Loss

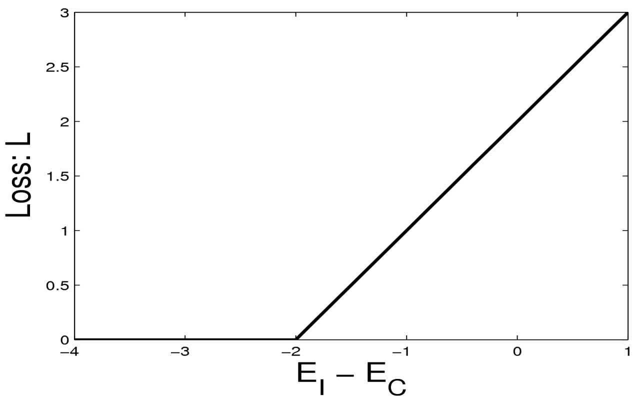
- Qm increases with the energy of the correct answer
- Qm decreases with the energy of the most offending incorrect answer
- whenever it is less than the energy of the correct answer plus a margin m.

Examples of Generalized Margin Losses

$$L_{\text{hinge}}(W, Y^i, X^i) = \max\left(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right),$$

Hinge Loss

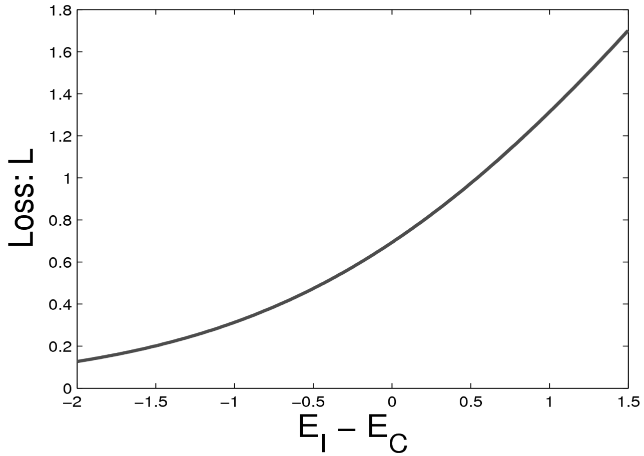
- With the linearly-parameterized binary classifier architecture, we get linear SVM



$$L_{\log}(W, Y^i, X^i) = \log\left(1 + e^{E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)}\right).$$

Log Loss

- "soft hinge" loss
- With the linearly-parameterized binary classifier architecture, we get linear Logistic Regression

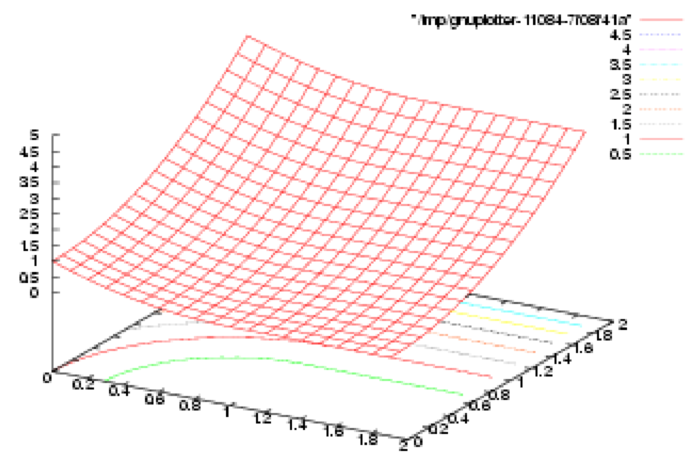


Examples of Margin Losses: Square-Square Loss

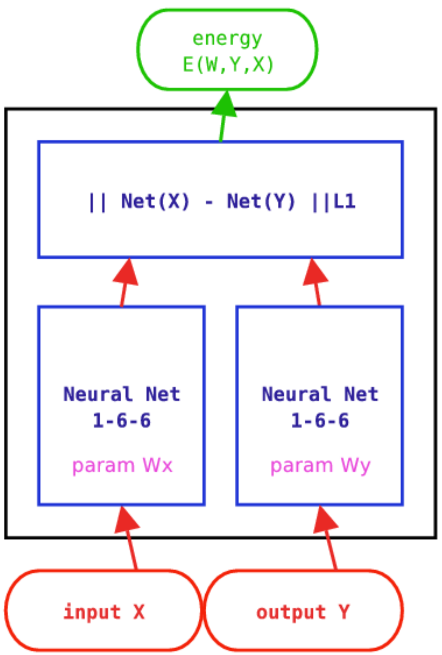
$$L_{\text{sq-sq}}(W, Y^i, X^i) = E(W, Y^i, X^i)^2 + \left(\max(0, m - E(W, \bar{Y}^i, X^i))\right)^2.$$

Square-Square Loss

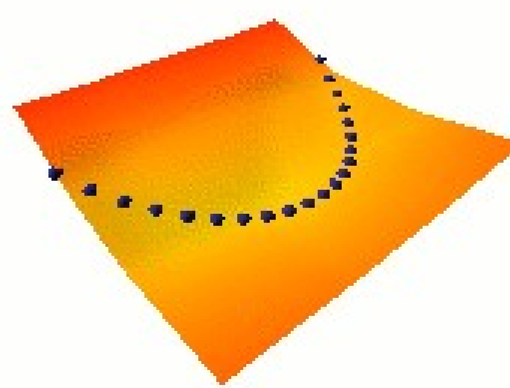
- [LeCun-Huang 2005]
- Appropriate for positive energy functions



Learning Y = X^2



(b)



NO COLLAPSE!!!

Other Margin-Like Losses

- **LVQ2 Loss** [Kohonen, Oja], Driancourt-Bottou 1991]

$$L_{\mathrm{lvq2}}(W, Y^i, X^i) = \min\left(1, \max\left(0, \frac{E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)}{\delta E(W, \bar{Y}^i, X^i)}\right)\right),$$

- **Minimum Classification Error Loss** [Juang, Chou, Lee 1997]

$$L_{\mathrm{mce}}(W, Y^i, X^i) = \sigma\left(E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right),$$

$$\sigma(x) = (1 + e^{-x})^{-1}$$

- **Square-Exponential Loss** [Osadchy, Miller, LeCun 2004]

$$L_{\mathrm{sq-exp}}(W, Y^i, X^i) = E(W, Y^i, X^i)^2 + \gamma e^{-E(W, \bar{Y}^i, X^i)},$$

Negative Log-Likelihood Loss

- **Conditional probability of the samples (assuming independence)**

$$P(Y^1,\dots,Y^P|X^1,\dots,X^P,W) = \prod_{i=1}^{P} P(Y^i|X^i,W).$$

$$-\log \prod_{i=1}^{P} P(Y^i|X^i,W) = \sum_{i=1}^{P} -\log P(Y^i|X^i,W).$$

- **Gibbs distribution:** $P(Y|X^i,W) = \dfrac{e^{-\beta E(W,Y,X^i)}}{\int_{y\in\mathcal{Y}} e^{-\beta E(W,y,X^i)}}.$

$$-\log \prod_{i=1}^{P} P(Y^i|X^i,W) = \sum_{i=1}^{P} \beta E(W,Y^i,X^i) + \log \int_{y\in\mathcal{Y}} e^{-\beta E(W,y,X^i)}.$$

- **We get the NLL loss by dividing by P and Beta:**

$$\mathcal{L}_{\text{nll}}(W,\mathcal{S}) = \frac{1}{P}\sum_{i=1}^{P}\left(E(W,Y^i,X^i) + \frac{1}{\beta}\log \int_{y\in\mathcal{Y}} e^{-\beta E(W,y,X^i)}\right).$$

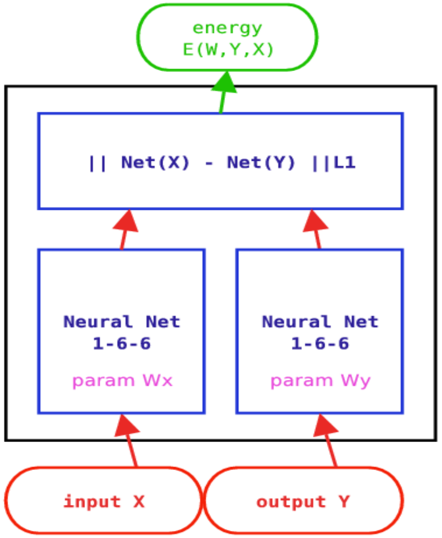
- **Reduces to the perceptron loss when Beta->infinity**

Negative Log-Likelihood Loss

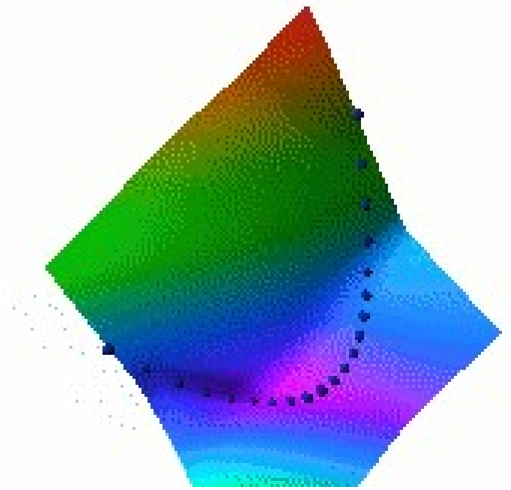
- Pushes down on the energy of the correct answer
- Pulls up on the energies of all answers in proportion to their probability

$$\mathcal{L}_{\rm nll}(W,\mathcal{S}) = \frac{1}{P}\sum_{i=1}^{P}\left(E(W,Y^i,X^i) + \frac{1}{\beta}\log\int_{y\in\mathcal{Y}} e^{-\beta E(W,y,X^i)}\right).$$

$$\frac{\partial L_{\rm nll}(W,Y^i,X^i)}{\partial W} = \frac{\partial E(W,Y^i,X^i)}{\partial W} - \int_{Y\in\mathcal{Y}} \frac{\partial E(W,Y,X^i)}{\partial W} P(Y|X^i,W),$$



(b)



Negative Log-Likelihood Loss: Binary Classification

- **Binary Classifier Architecture:**

$$\mathcal{L}_{\text{nll}}(W,\mathcal{S}) = \frac{1}{P}\sum_{i=1}^{P}\left[-Y^iG_W(X^i) + \log\left(e^{Y^iG_W(X^i)} + e^{-Y^iG_W(X^i)}\right)\right].$$

$$\mathcal{L}_{\text{nll}}(W,\mathcal{S}) = \frac{1}{P}\sum_{i=1}^{P}\log\left(1 + e^{-2Y^iG_W(X^i)}\right),$$

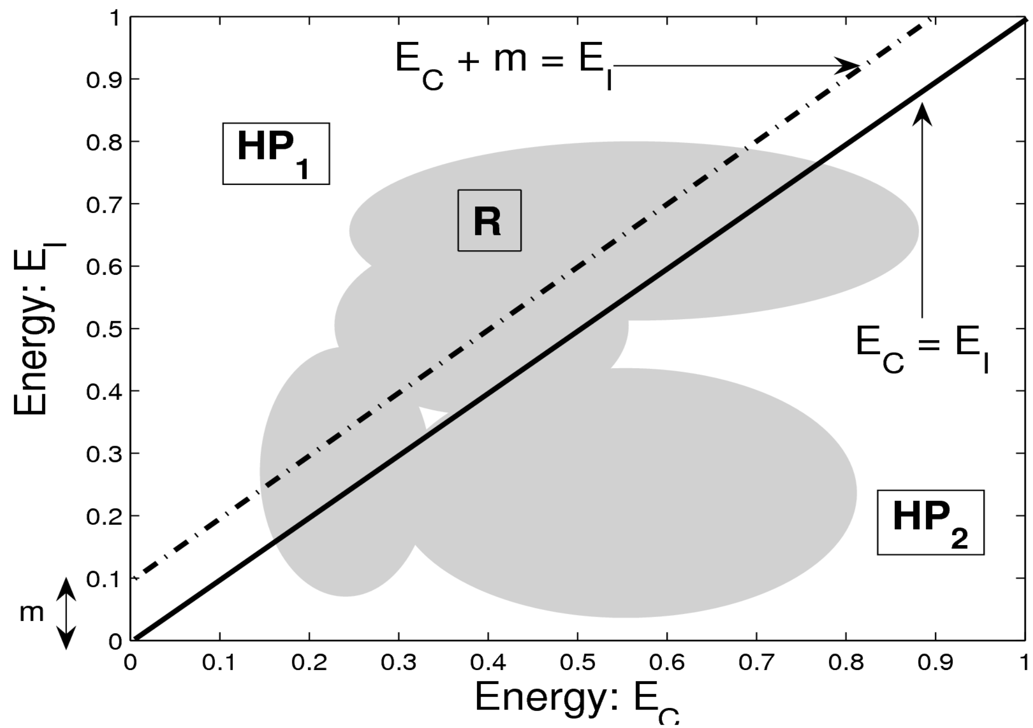
- **Linear Binary Classifier Architecture:**

$$\mathcal{L}_{\text{nll}}(W,\mathcal{S}) = \frac{1}{P}\sum_{i=1}^{P}\log\left(1 + e^{-2Y^iW^T\Phi(X^i)}\right).$$

- **Learning Rule: logistic regression**

What Makes a “Good” Loss Function

- **Good loss functions make the machine produce the correct answer**
  - Avoid collapses and flat energy surfaces



Sufficient Condition on the Loss

Let (X^i, Y^i) be the i^{th} training example and m be a positive margin. Minimizing the loss function L will cause the machine to satisfy $E(W, Y^i, X^i) < E(W, Y, X^i) - m$ for all $Y \neq Y^i$, if there exists at least one point (e_1, e_2) with $e_1 + m < e_2$ such that for all points (e'_1, e'_2) with $e'_1 + m \geq e'_2$, we have

$$Q_{[E_y]}(e_1, e_2) < Q_{[E_y]}(e'_1, e'_2),$$

where $Q_{[E_y]}$ is given by

$$L(W, Y^i, X^i) = Q_{[E_y]}(E(W, Y^i, X^i), E(W, \bar{Y}^i, X^i)).$$

What Make a “Good” Loss Function

Good and bad loss functions

Loss (equation #)	**Formula**	**Margin**
energy loss	$E(W, Y^i, X^i)$	none
perceptron	$E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i)$	0
hinge	$\max\left(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$	m
log	$\log\left(1 + e^{E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)}\right)$	> 0
LVQ2	$\min\left(M, \max(0, E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$	0
MCE	$\left(1 + e^{-\left(E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)\right)}\right)^{-1}$	> 0
square-square	$E(W, Y^i, X^i)^2 - \left(\max(0, m - E(W, \bar{Y}^i, X^i))\right)^2$	m
square-exp	$E(W, Y^i, X^i)^2 + \beta e^{-E(W,\bar{Y}^i,X^i)}$	> 0
NLL/MMI	$E(W, Y^i, X^i) + \frac{1}{\beta} \log \int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)}$	> 0
MEE	$1 - e^{-\beta E(W,Y^i,X^i)} / \int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)}$	> 0

Advantages/Disadvantages of various losses

- **Loss functions differ in how they pick the point(s) whose energy is pulled up, and how much they pull them up**
- **Losses with a log partition function in the contrastive term pull up all the bad answers simultaneously.**
  - This may be good if the gradient of the contrastive term can be computed efficiently
  - This may be bad if it cannot, in which case we might as well use a loss with a single point in the contrastive term
- **Variational methods pull up many points, but not as many as with the full log partition function.**
- **Efficiency of a loss/architecture: how many energies are pulled up for a given amount of computation?**
  - The theory for this is to be developed

Linear Machines: Regression with Mean Square

Linear Regression, Mean Square Loss:

- decision rule: $y = W'X$
- loss function: $L(W, y^i, X^i) = \frac{1}{2}(y^i - W'X^i)^2$
- gradient of loss: $\frac{\partial L(W, y^i, X^i)}{\partial W}' = -(y^i - W(t)'X^i)X^i$
- update rule: $W(t+1) = W(t) + \eta(t)(y^i - W(t)'X^i)X^i$
- direct solution: solve linear system $[\sum_{i=1}^{P} X^i X^{i\prime}]W = \sum_{i=1}^{P} y^i X^i$

Linear Machines: Perceptron

Perceptron:

- decision rule: $y = F(W'X)$ (F is the threshold function)
- loss function: $L(W, y^i, X^i) = (F(W'X^i) - y^i)W'X^i$
- gradient of loss: $\frac{\partial L(W,y^i,X^i)}{\partial W}' = -(y^i - F(W(t)'X^i))X^i$
- update rule: $W(t+1) = W(t) + \eta(t)(y^i - F(W(t)'X^i))X^i$
- direct solution: find W such that $-y^i F(W'X^i) < 0 \quad \forall i$

Linear Machines: Logistic Regression

Logistic Regression, Negative Log-Likelihood Loss function:

- decision rule: $y = F(W'X)$, with $F(a) = \tanh(a) = \frac{1-\exp(a)}{1+\exp(a)}$ (sigmoid function).
- loss function: $L(W, y^i, X^i) = 2\log(1 + \exp(-y^i W' X^i))$
- gradient of loss: $\frac{\partial L(W,y^i,X^i)}{\partial W}' = -\left(Y^i - F(W'X)\right) X^i$
- update rule: $W(t+1) = W(t) + \eta(t)(y^i - F(W(t)'X^i))X^i$