Probabilistyczne podejścia do ML

Maciej Bartczak

CORE 2020 #2
15.04.2020

Modelowanie niepewności

Klasyfikacja

$$F(x) = [w_1, \ldots, w_n] \quad w_j \in \mathbb{R}$$

$$CrossEntropy(F(x), y = k) = -\log\left(\frac{\exp(w_k)}{\sum exp(w_j)}\right)$$

Modelowanie niepewności

Klasyfikacja

$$F(x) = [w_1, \ldots, w_n] \quad w_j \in \mathbb{R}$$

$$CrossEntropy(F(x), y = k) = -\log\left(\frac{\exp(w_k)}{\sum exp(w_j)}\right)$$

Dla zmiennej losowej Y takiej że

$$\mathbb{P}(Y = k) = \frac{\exp(w_k)}{\sum exp(w_j)}$$

zachodzi

$$CrossEntropy(F(x), y = k) = -\log(\mathbb{P}(Y = k))$$

Modelowanie niepewności

Segmentacja

$$F(x) = [w_{pk}]_{p \in piksele,\ k \in klasy} \quad w_{pk} \in \mathbb{R}$$

$$CrossEntropy(F(x), [y_p]) = -\sum_p \log\left(\frac{\exp(w_{py_p})}{\sum_k exp(w_{pk})}\right)$$

Modelowanie niepewności

Segmentacja

$$F(x) = [w_{pk}]_{p \in piksele,\ k \in klasy} \quad w_{pk} \in \mathbb{R}$$

$$CrossEntropy(F(x), [y_p]) = -\sum_p \log\left(\frac{\exp(w_{py_p})}{\sum_k exp(w_{pk})}\right)$$

Dla **niezależnych** zmiennych losowych $[Y_p]_{p \in piksele}$ takich że

$$\mathbb{P}(Y_p = k) = \frac{\exp(w_{pk})}{\sum_j exp(w_{pj})}$$

zachodzi

$$CrossEntropy(F(x), [y_p]) = -\log(\mathbb{P}(\forall_p Y_p = y_p))$$

Modelowanie niepewności

Ogólny setup

- model zwraca rozkład prawdopodobieństwa
- w przypadku skończonej liczby możliwości można zwracać ich prawdopodobieństwa
  - w przypadku klasyfikacji robimy dokładnie to
  - w przypadku segmentacji upraszczamy sytuację założeniem **niezależności**, dzięki czemu zwracany rozkład prawdopodobieństwa opisany jest $P \cdot K$ zamiast P^K parametrami (P - liczba pikseli, K - liczba klas)
- trzeba określić sensowną parametryzację zwracanych rozkładów, a najlepiej adekwatną do problemu
- minimalizacja lossu = maksymalizacja prawdopodobieństw obserwacji

Modelowanie niepewności

Przykład 1: model zwracający predykcję temperatury na jutro

Modelowanie niepewności

Przykład 1: model zwracający predykcję temperatury na jutro

1. $Y \sim p_0 \delta_{0^\circ C} + p_1 \delta_{1^\circ C} + \cdots + p_{30} \delta_{30^\circ C}$

Modelowanie niepewności

Przykład 1: model zwracający predykcję temperatury na jutro

1. $Y \sim p_0 \delta_{0^\circ C} + p_1 \delta_{1^\circ C} + \cdots + p_{30} \delta_{30^\circ C}$
2. $Y \sim N(\mu, \sigma^2)$

Modelowanie niepewności

Przykład 1: model zwracający predykcję temperatury na jutro

1. $Y \sim p_0 \delta_{0^\circ C} + p_1 \delta_{1^\circ C} + \cdots + p_{30} \delta_{30^\circ C}$
2. $Y \sim N(\mu, \sigma^2)$
3. $Y \sim \theta \cdot N(\mu_1, \sigma^2) + (1 - \theta) \cdot N(\mu_2, \sigma^2) \qquad (\mu_1 \leqslant \mu_2)$

output/loss

1. jak w klasyfikacji

Modelowanie niepewności

Przykład 1: model zwracający predykcję temperatury na jutro

1. $Y \sim p_0\delta_{0^\circ C} + p_1\delta_{1^\circ C} + \cdots + p_{30}\delta_{30^\circ C}$
2. $Y \sim N(\mu, \sigma^2)$
3. $Y \sim \theta \cdot N(\mu_1, \sigma^2) + (1-\theta) \cdot N(\mu_2, \sigma^2) \qquad (\mu_1 \leqslant \mu_2)$

output/loss

1. jak w klasyfikacji
2. $F(x) = [\mu, \sigma]$, $Loss(F(x), y) = -\log p_{N(\mu,\sigma^2)}(y) =$
   $= -\log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)\right) = const + \log\sigma + \frac{(y-\mu)^2}{2\sigma^2}$

Modelowanie niepewności

Przykład 1: model zwracający predykcję temperatury na jutro

1. $Y \sim p_0\delta_{0^\circ C} + p_1\delta_{1^\circ C} + \cdots + p_{30}\delta_{30^\circ C}$
2. $Y \sim N(\mu, \sigma^2)$
3. $Y \sim \theta \cdot N(\mu_1, \sigma^2) + (1-\theta) \cdot N(\mu_2, \sigma^2) \qquad (\mu_1 \leqslant \mu_2)$

output/loss

1. jak w klasyfikacji
2. $F(x) = [\mu, \sigma]$, $Loss(F(x), y) = -\log p_{N(\mu,\sigma^2)}(y) =$
   $= -\log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)\right) = const + \log\sigma + \frac{(y-\mu)^2}{2\sigma^2}$
3. $F(x) = [\mu_1, \mu_2, \sigma, \theta]$, $Loss(F(x), y) =$
   $= -\log\left(\frac{\theta}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu_1)^2}{2\sigma^2}\right) + \frac{1-\theta}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu_2)^2}{2\sigma^2}\right)\right)$

Modelowanie niepewności

Przykład 2: model zwracający predykcję temperatury w wielu miejscach

Modelowanie niepewności

Przykład 2: model zwracający predykcję temperatury w wielu miejscach

1. $Y_j \sim N(\mu_j, \sigma^2)$ niezależnie, $F(x) = [\mu_1, \ldots, \mu_n, \sigma]$,

$$Loss(F(x), [y_1, \cdots, y_n]) = n \log \sigma + \sum_j \frac{(y_j - \mu_j)^2}{2\sigma^2}$$

Modelowanie niepewności

Przykład 2: model zwracający predykcję temperatury w wielu miejscach

1. $Y_j \sim N(\mu_j, \sigma^2)$ niezależnie, $F(x) = [\mu_1, \ldots, \mu_n, \sigma]$,

$$Loss(F(x), [y_1, \cdots, y_n]) = n \log \sigma + \sum_j \frac{(y_j - \mu_j)^2}{2\sigma^2}$$

2. $Y_{ref} \sim N(\mu, \sigma^2)$, $Y_j | Y_{ref} \overset{ciid}{\sim} Y_{ref} + N(0, \nu^2)$, $F(x) = [\mu, \sigma, \nu]$

Modelowanie niepewności

Przykład 2: model zwracający predykcję temperatury w wielu miejscach

1. $Y_j \sim N(\mu_j, \sigma^2)$ niezależnie, $F(x) = [\mu_1, \ldots, \mu_n, \sigma]$,

$$Loss(F(x), [y_1, \cdots, y_n]) = n \log \sigma + \sum_j \frac{(y_j - \mu_j)^2}{2\sigma^2}$$

2. $Y_{ref} \sim N(\mu, \sigma^2)$, $Y_j | Y_{ref} \overset{ciid}{\sim} Y_{ref} + N(0, \nu^2)$, $F(x) = [\mu, \sigma, \nu]$

$$\begin{aligned} p(y_{ref}, y_1, \ldots, y_n) &= p(y_{ref}) p(y_1, \ldots, y_n | y_{ref}) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_{ref} - \mu)^2}{2\sigma^2}\right) \prod_j \frac{1}{\sqrt{2\pi\nu^2}} \exp\left(-\frac{(y_j - y_{ref})^2}{2\nu^2}\right) \end{aligned}$$

Modelowanie niepewności

Przykład 2: model zwracający predykcję temperatury w wielu miejscach

1. $Y_j \sim N(\mu_j, \sigma^2)$ niezależnie, $F(x) = [\mu_1, \ldots, \mu_n, \sigma]$,

$$Loss(F(x), [y_1, \cdots, y_n]) = n \log \sigma + \sum_j \frac{(y_j - \mu_j)^2}{2\sigma^2}$$

2. $Y_{ref} \sim N(\mu, \sigma^2)$, $Y_j | Y_{ref} \overset{ciid}{\sim} Y_{ref} + N(0, \nu^2)$, $F(x) = [\mu, \sigma, \nu]$

$$\begin{aligned} p(y_{ref}, y_1, \ldots, y_n) &= p(y_{ref}) p(y_1, \ldots, y_n | y_{ref}) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_{ref} - \mu)^2}{2\sigma^2}\right) \prod_j \frac{1}{\sqrt{2\pi\nu^2}} \exp\left(-\frac{(y_j - y_{ref})^2}{2\nu^2}\right) \end{aligned}$$

$$\begin{aligned} Loss(F(x), [y_{ref}, y_1, \cdots, y_n]) &= -\log p(y_{ref}, y_1, \ldots, y_n) \\ &= const + \log \sigma + n \log \nu + \frac{(y_{ref} - \mu)^2}{2\sigma^2} + \sum_j \frac{(y_j - y_{ref})^2}{2\nu^2} \end{aligned}$$

Bayesian NN

Przykład: mam monetę i wykonuję nią parę rzutów, co można powiedzieć o jej sprawiedliwości?

Bayesian NN

Przykład: mam monetę i wykonuję nią parę rzutów, co można powiedzieć o jej sprawiedliwości?

- podejście częstościowe/klasyczne (*frequentist*)
  - liczenie statystyk opisowych próbki
  - testowanie dopasowania do określonej klasy rozkładów
- podejście bayesowskie
  - określenie początkowego *przekonania* o sprawiedliwości monety
  - update'owanie go wraz z obserwacjami

Bayesian NN

Przykład: mam monetę i wykonuję nią parę rzutów, co można powiedzieć o jej sprawiedliwości?

- podejście częstościowe/klasyczne (*frequentist*)
  - liczenie statystyk opisowych próbki
  - testowanie dopasowania do określonej klasy rozkładów
- podejście bayesowskie
  - określenie początkowego *przekonania* o sprawiedliwości monety
  - update'owanie go wraz z obserwacjami

$$\vartheta \sim Beta(\alpha, \beta),\ X_j|\vartheta \sim Toss(\vartheta)$$

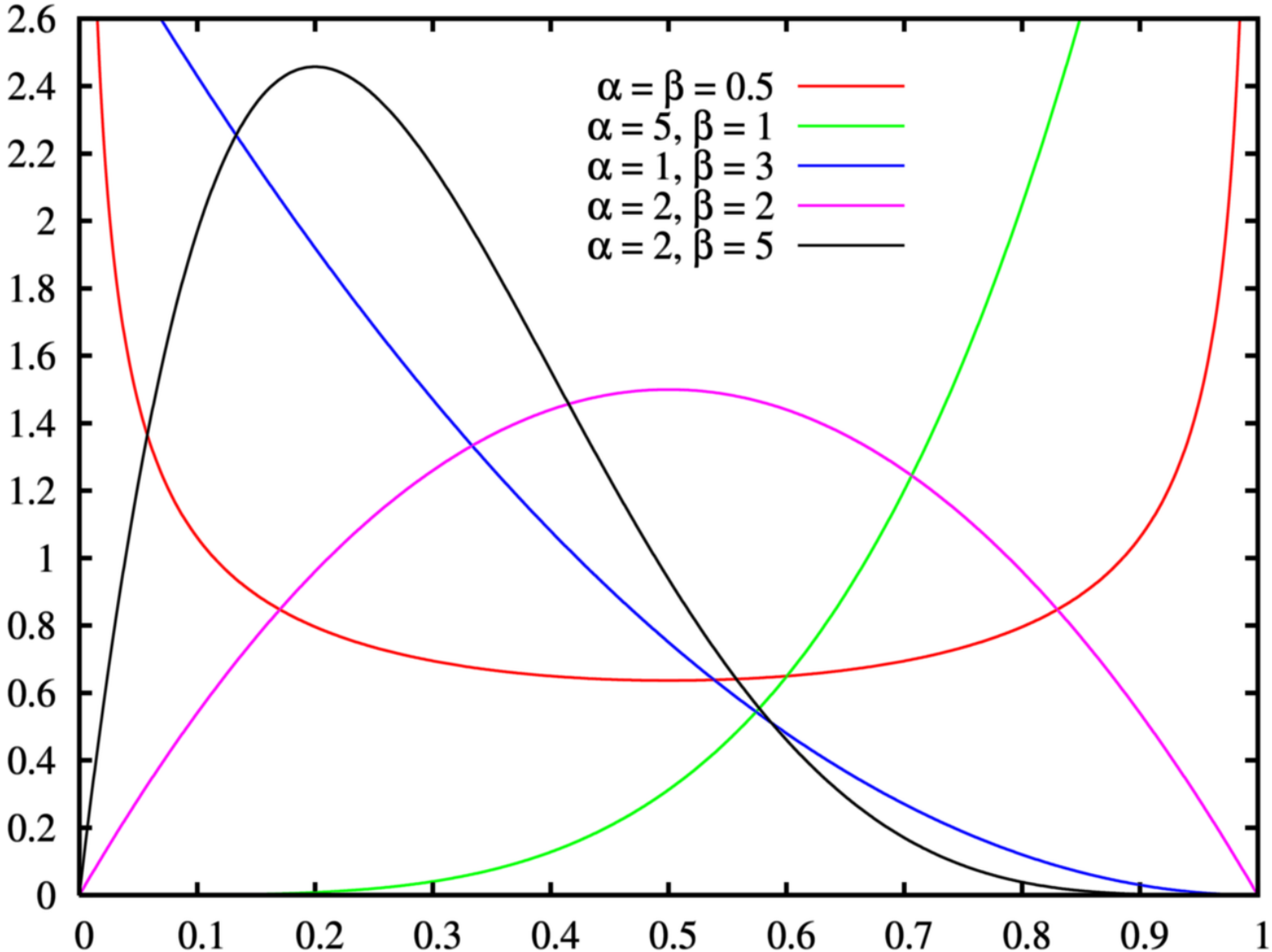
$$p(\theta, x_1, \ldots, x_m) = const \cdot \theta^{\alpha-1}(1-\theta)^{\beta-1} \cdot \prod_j \theta^{x_j}(1-\theta)^{1-x_j} \Rightarrow$$

$$p(\theta|x_1, \ldots, x_m) = const \cdot \theta^{\alpha-1+\sum_j x_j}(1-\theta)^{\beta-1+n-\sum_j x_j} \Rightarrow$$

$$\vartheta|X_1, \ldots X_n \sim Beta(\alpha + \sum_j X_j, \beta + n - \sum_j X_j)$$

$$\mathbb{E}[\vartheta|X_1, \ldots X_n] = \frac{\alpha'}{\alpha' + \beta'} = \frac{\alpha + \sum_j X_j}{\alpha + \beta + n} \approx \frac{\sum_j X_j}{n} \xrightarrow{p.n.} \vartheta$$

Bayesian NN



źródło

Bayesian NN

Przeniesienie na deep learning

- punkt wyjścia: sieć neuronowa + początkowe *przekonanie* o wartości wag, np. $W \sim N(0, I)$
- model rzeczywistości: $Y|X, W \sim F_W(X)$
- trening: update'owanie *przekonania*, $W|(X_1, Y_1), \ldots (X_n, Y_n) \sim W'$
- predykcja: $Y'|X', (X_1, Y_1), \ldots (X_n, Y_n) \sim F_{W'}(X')$

Bayesian NN

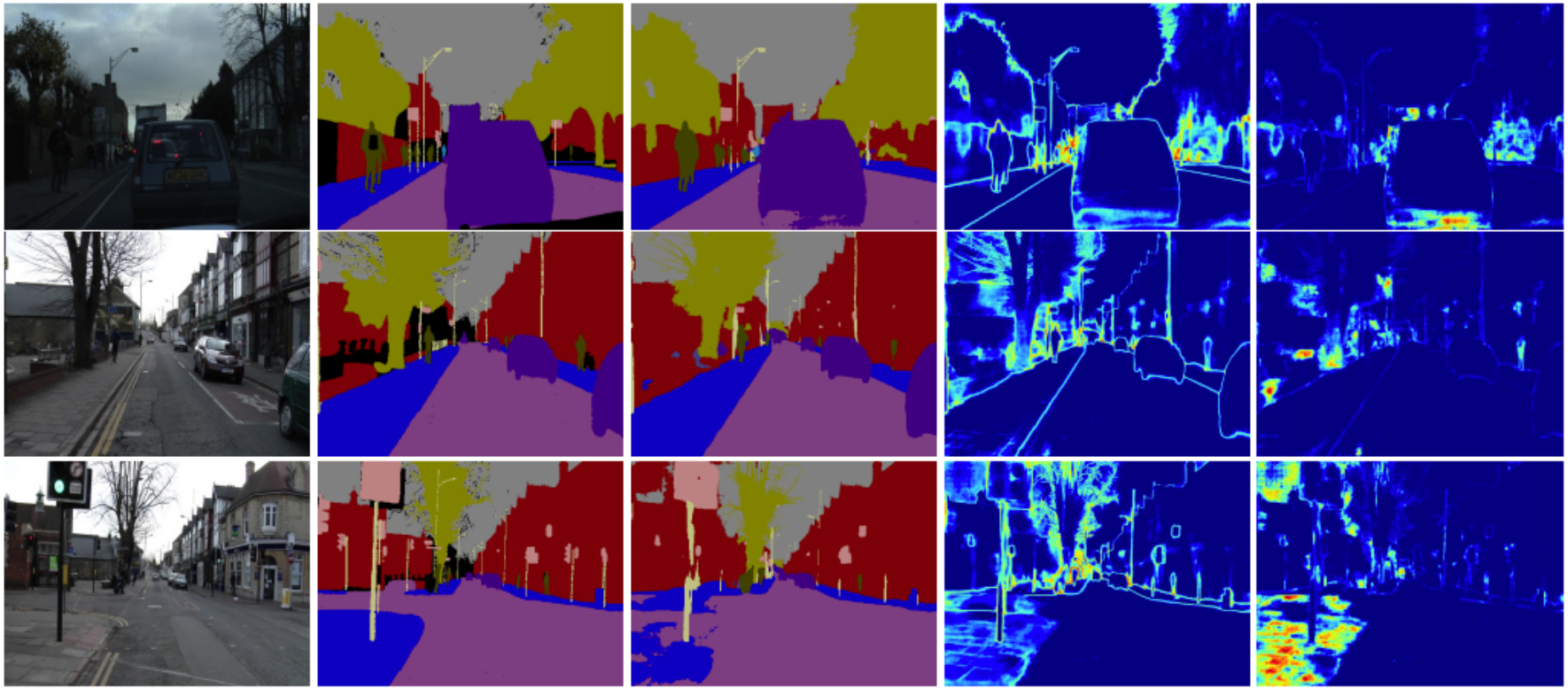
Przeniesienie na deep learning

- punkt wyjścia: sieć neuronowa + początkowe *przekonanie* o wartości wag, np. $W \sim N(0, I)$
- model rzeczywistości: $Y|X, W \sim F_W(X)$
- trening: update'owanie *przekonania*, $W|(X_1, Y_1), \ldots (X_n, Y_n) \sim W'$
- predykcja: $Y'|X', (X_1, Y_1), \ldots (X_n, Y_n) \sim F_{W'}(X')$

Problem

- $? \sim W|(X_1, Y_1), \ldots (X_n, Y_n)$
- w wyjątkowych sytuacjach da się wyznaczyć analitycznie
- rozwiązanie: Variational Inference

Bayesian NN



(a) Input Image (b) Ground Truth (c) Semantic Segmentation (d) Aleatoric Uncertainty (e) Epistemic Uncertainty

Kendall, Alex, and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?." Advances in neural information processing systems. 2017.

Bayesian NN

Variational Inference

- $\widehat{W} \sim q_\theta(w)$ - parametryzowany rozkład prawdopodobieństwa
- szukanie θ takiego że $\widehat{W} \overset{approx.}{\sim} W'$
- minimalizacja $KL(\widehat{W} \| W')$, gdzie $KL(X \| Y) = \mathbb{E} \log\left(\frac{p_X(X)}{p_Y(X)}\right)$

$$p(w|x,y) = \frac{p(x,y,w)}{p(x,y)} = \frac{p(y|x,w)p(x,w)}{p(x,y)}$$
$$= \frac{p(y|x,w)p(x)p(w)}{p(x,y)} = \frac{p(y|x,w)p(w)}{p(y|x)}$$

Bayesian NN

Variational Inference

$$
\begin{aligned}
KL(\widehat{W} \| W') &= \int q_\theta(\widehat{w}) \log \left(\frac{q_\theta(\widehat{w})}{p_{w|x,y}(\widehat{w}|x,y)} \right) d\widehat{w} \\
&= \int q_\theta(w) \log \left(\frac{q_\theta(w)}{p(w|x,y)} \right) dw \\
&= \int q_\theta(w) \log \left(\frac{q_\theta(w) p(y|x)}{p(y|x,w) p(w)} \right) dw \\
&= \log p(y|x) + \int q_\theta(w) \log \left(\frac{q_\theta(w)}{p(y|x,w) p(w)} \right) dw
\end{aligned}
$$

Bayesian NN

Variational Inference

$$
\begin{aligned}
KL(\widehat{W} \| W') &= \int q_\theta(\widehat{w}) \log \left(\frac{q_\theta(\widehat{w})}{p_{w|x,y}(\widehat{w}|x,y)} \right) d\widehat{w} \\
&= \int q_\theta(w) \log \left(\frac{q_\theta(w)}{p(w|x,y)} \right) dw \\
&= \int q_\theta(w) \log \left(\frac{q_\theta(w) p(y|x)}{p(y|x,w)p(w)} \right) dw \\
&= \log p(y|x) + \int q_\theta(w) \log \left(\frac{q_\theta(w)}{p(y|x,w)p(w)} \right) dw
\end{aligned}
$$

$$
\begin{aligned}
&\nabla_\theta KL(\widehat{W} \| W') = \\
&= \int \nabla_\theta q_\theta(w) \cdot \log \left(\frac{q_\theta(w)}{p(y|x,w)p(w)} \right) + q_\theta(w) \cdot \frac{\nabla_\theta q_\theta(w)}{q_\theta(w)} dw \\
&= \int q_\theta(w) \cdot \nabla_\theta \log q_\theta(w) \cdot \left(\log \left(\frac{q_\theta(w)}{p(y|x,w)p(w)} \right) + 1 \right) dw
\end{aligned}
$$

Bayesian NN

Stochastic Variational Inference

$$\nabla_\theta KL(\widehat{W}\|W') =$$

$$= \int q_\theta(w) \cdot \nabla_\theta \log q_\theta(w) \cdot \left(\log \left(\frac{q_\theta(w)}{p(y|x,w)p(w)} \right) + 1 \right) dw$$

$$= \mathbb{E}\left[\nabla_\theta \log q_\theta(\widehat{W}) \cdot \left(\log \left(\frac{q_\theta(\widehat{W})}{p(y|x,\widehat{W})p(\widehat{W})} \right) + 1 \right) \right] =: \mathbb{E}[f(\widehat{W})]$$

Rozkład $\widehat{W}$ jest znany, $\mathbb{E}[f(\widehat{W})]$ można policzyć metodami Monte Carlo. Problem tkwi w

$$\log p(y|x,\widehat{W}) = \sum_j \log p(y_j|x_j,\widehat{W})$$

Bayesian NN

Stochastic Variational Inference

$$\nabla_\theta KL(\widehat{W}\|W') =$$
$$= \int q_\theta(w) \cdot \nabla_\theta \log q_\theta(w) \cdot \left(\log\left(\frac{q_\theta(w)}{p(y|x,w)p(w)}\right) + 1\right) dw$$
$$= \mathbb{E}\left[\nabla_\theta \log q_\theta(\widehat{W}) \cdot \left(\log\left(\frac{q_\theta(\widehat{W})}{p(y|x,\widehat{W})p(\widehat{W})}\right) + 1\right)\right] =: \mathbb{E}[f(\widehat{W})]$$

Rozkład $\widehat{W}$ jest znany, $\mathbb{E}[f(\widehat{W})]$ można policzyć metodami Monte Carlo. Problem tkwi w

$$\log p(y|x,\widehat{W}) = \sum_j \log p(y_j|x_j,\widehat{W})$$

Rozwiązanie: *mini batches* J

$$\log p(y|x,\widehat{W}) \approx \frac{|J|}{n} \sum_{j\in J} \log p(y_j|x_j,\widehat{W})$$

Bayesian NN

Stochastic Variational Inference

- zadać rozkład W, np $W \sim N(0, I)$
- model rzeczywistości: $Y|X, W \sim F_W(X)$
- obserwacje: $(x_1, y_1), \ldots (x_n, y_n)$
- trening: fitowanie

$$W' \sim W|\{(X_1, Y_1), \ldots (X_n, Y_n) = (x_1, y_1), \ldots (x_n, y_n)\}$$

  - parametryzacja rozkładu $\widehat{W} \sim q_\theta$, wybranie θ_0
  - liczenie $\nabla_\theta KL(\widehat{W} \| W')$, na *mini batchu*, metodami Monte Carlo
  - gradient descent
- predykcja: $Y'|X', (X_1, Y_1), \ldots (X_n, Y_n) \sim F_{W'}(X')$

Bayesian NN

CamVid	IoU
SegNet [28]	46.4
FCN-8 [29]	57.0
DeepLab-LFOV [24]	61.6
Bayesian SegNet [22]	63.1
Dilation8 [30]	65.3
Dilation8 + FSO [31]	66.1
DenseNet [20]	66.9
This work:	
DenseNet (Our Implementation)	67.1
+ Aleatoric Uncertainty	67.4
+ Epistemic Uncertainty	67.2
+ Aleatoric & Epistemic	**67.5**

(a) CamVid dataset for road scene segmentation.

NYUv2 40-class	Accuracy	IoU
SegNet [28]	66.1	23.6
FCN-8 [29]	61.8	31.6
Bayesian SegNet [22]	68.0	32.4
Eigen and Fergus [32]	65.6	34.1
This work:		
DeepLabLargeFOV	70.1	36.5
+ Aleatoric Uncertainty	70.4	37.1
+ Epistemic Uncertainty	70.2	36.7
+ Aleatoric & Epistemic	**70.6**	**37.3**

(b) NYUv2 40-class dataset for indoor scenes.

Table 1: **Semantic segmentation performance.** Modeling both aleatoric and epistemic uncertainty gives a notable improvement in segmentation accuracy over state of the art baselines.

Make3D	rel	rms	$\log_{10}$
Karsch et al. [33]	0.355	9.20	0.127
Liu et al. [34]	0.335	9.49	0.137
Li et al. [35]	0.278	7.19	0.092
Laina et al. [26]	0.176	4.46	0.072
This work:			
DenseNet Baseline	0.167	3.92	0.064
+ Aleatoric Uncertainty	**0.149**	3.93	**0.061**
+ Epistemic Uncertainty	0.162	**3.87**	0.064
+ Aleatoric & Epistemic	**0.149**	4.08	0.063

(a) Make3D depth dataset [25].

NYU v2 Depth	rel	rms	$\log_{10}$	δ_1	δ_2	δ_3
Karsch et al. [33]	0.374	1.12	0.134	-	-	-
Ladicky et al. [36]	-	-	-	54.2%	82.9%	91.4%
Liu et al. [34]	0.335	1.06	0.127	-	-	-
Li et al. [35]	0.232	0.821	0.094	62.1%	88.6%	96.8%
Eigen et al. [27]	0.215	0.907	-	61.1%	88.7%	97.1%
Eigen and Fergus [32]	0.158	0.641	-	76.9%	95.0%	98.8%
Laina et al. [26]	0.127	0.573	0.055	81.1%	95.3%	98.8%
This work:						
DenseNet Baseline	0.117	0.517	0.051	80.2%	95.1%	98.8%
+ Aleatoric Uncertainty	0.112	0.508	0.046	81.6%	95.8%	98.8%
+ Epistemic Uncertainty	0.114	0.512	0.049	81.1%	95.4%	98.8%
+ Aleatoric & Epistemic	**0.110**	**0.506**	**0.045**	**81.7%**	**95.9%**	**98.9%**

(b) NYUv2 depth dataset [23].

Table 2: **Monocular depth regression performance.** Comparison to previous approaches on depth regression dataset NYUv2 Depth. Modeling the combination of uncertainties improves accuracy.

Kendall, Alex, and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?." Advances in neural information processing systems. 2017.

Monte Carlo Dropout

Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." international conference on machine learning. 2016.

- standardowe modele trenowane z dropoutem
- taki trening jest zgrubsza równoważny Variational Inference dla W gaussowskich i q_θ produktowych mieszanek gaussowskich (chyba)
- predykcja także jest wykonywana z dropoutem, co daje losowy output
- losowy output traktowany jest jak z $N(\mu, \sigma^2)$

Monte Carlo Dropout

			Avg. Test RMSE and Std. Errors			Avg. Test LL and Std. Errors		
Dataset	N	Q	**VI**	**PBP**	**Dropout**	**VI**	**PBP**	**Dropout**
Boston Housing	506	13	4.32 ±0.29	3.01 ±0.18	**2.97 ±0.19**	-2.90 ±0.07	-2.57 ±0.09	**-2.46 ±0.06**
Concrete Strength	1,030	8	7.19 ±0.12	5.67 ±0.09	**5.23 ±0.12**	-3.39 ±0.02	-3.16 ±0.02	**-3.04 ±0.02**
Energy Efficiency	768	8	2.65 ±0.08	1.80 ±0.05	**1.66 ±0.04**	-2.39 ±0.03	-2.04 ±0.02	**-1.99 ±0.02**
Kin8nm	8,192	8	**0.10 ±0.00**	**0.10 ±0.00**	**0.10 ±0.00**	0.90 ±0.01	0.90 ±0.01	**0.95 ±0.01**
Naval Propulsion	11,934	16	**0.01 ±0.00**	**0.01 ±0.00**	**0.01 ±0.00**	3.73 ±0.12	3.73 ±0.01	**3.80 ±0.01**
Power Plant	9,568	4	4.33 ±0.04	4.12 ±0.03	**4.02 ±0.04**	-2.89 ±0.01	-2.84 ±0.01	**-2.80 ±0.01**
Protein Structure	45,730	9	4.84 ±0.03	4.73 ±0.01	**4.36 ±0.01**	-2.99 ±0.01	-2.97 ±0.00	**-2.89 ±0.00**
Wine Quality Red	1,599	11	0.65 ±0.01	0.64 ±0.01	**0.62 ±0.01**	-0.98 ±0.01	-0.97 ±0.01	**-0.93 ±0.01**
Yacht Hydrodynamics	308	6	6.89 ±0.67	**1.02 ±0.05**	1.11 ±0.09	-3.43 ±0.16	-1.63 ±0.02	**-1.55 ±0.03**
Year Prediction MSD	515,345	90	9.034 ±NA	8.879 ±NA	**8.849 ±NA**	-3.622 ±NA	-3.603 ±NA	**-3.588 ±NA**

Table 1. **Average test performance in RMSE and predictive log likelihood** for a popular variational inference method (VI, Graves (2011)), Probabilistic back-propagation (PBP, Hernández-Lobato & Adams (2015)), and dropout uncertainty (Dropout). Dataset size (N) and input dimensionality (Q) are also given.

Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." international conference on machine learning. 2016.

Materiały

- Gal, Yarin, and Zoubin Ghahramani. *Dropout as a bayesian approximation: Representing model uncertainty in deep learning.* international conference on machine learning. 2016.
- Kendall, Alex, and Yarin Gal. *What uncertainties do we need in bayesian deep learning for computer vision?.* Advances in neural information processing systems. 2017.
- Probabilistic modelling: ermongroup.github.io/cs228-notes
- SVI package integrated with PyTorch: http://pyro.ai/